Geometric Tools Engine Update History

Contents

1 Updates to Version 7.0 7

2 Updates to Version 6.8 9

3 Updates to Version 6.7 19

4 Updates to Version 6.6 23

5 Updates to Version 6.5 25

6 Updates to Version 6.4 31

7 Updates to Version 6.3 34

8 Updates to Version 6.2 36

9 Updates to Version 6.1 40

10 Updates to Version 6.0 45

11 Updates to Version 5.15 51

12 Updates to Version 5.14 54

13 Updates to Version 5.13 56

14 Updates to Version 5.12 66

15 Updates to Version 5.11 68

16 Updates to Version 5.10 72

17 Updates to Version 5.9 73

18 Updates to Version 5.8 75

19 Updates to Version 5.7 77

20 Updates to Version 5.6 78

21 Updates to Version 5.5 80

22 Updates to Version 5.4 81

23 Updates to Version 5.3 81

24 Updates to Version 5.2 83

25 Updates to Version 5.1 85

26 Updates to Version 5.0 86

27 Updates to Version 4.9 88

28 Updates to Version 4.8 89

29 Updates to Version 4.7 91

30 Updates to Version 4.6 91

31 Updates to Version 4.5 92

32 Updates to Version 4.4 96

33 Updates to Version 4.3 99

34 Updates to Version 4.2 100

35 Updates to Version 4.1 101

36 Updates to Version 4.0 105

37 Updates to Version 3.30 106

38 Updates to Version 3.29 106

39 Updates to Version 3.28 106

40 Updates to Version 3.27 108

41 Updates to Version 3.26 108

42 Updates to Version 3.25 108

43 Updates to Version 3.24 127

44 Updates to Version 3.23 127

45 Updates to Version 3.22 132

46 Updates to Version 3.21 137

47 Updates to Version 3.20 140

48 Updates to Version 3.19 141

49 Updates to Version 3.18 142

50 Updates to Version 3.17 143

51 Updates to Version 3.16 145

52 Updates to Version 3.15 151

53 Updates to Version 3.14 153

54 Updates to Version 3.13 154

55 Updates to Version 3.12 156

56 Updates to Version 3.11 158

57 Updates to Version 3.10 160

58 Updates to Version 3.9 161

59 Updates to Version 3.8 164

60 Updates to Version 3.7 165

61 Updates to Version 3.6 166

62 Updates to Version 3.5 166
63 Updates to Version 3.4 167
64 Updates to Version 3.3 169
65 Updates to Version 3.2 171
66 Updates to Version 3.1 177
67 Updates to Version 3.0 179
68 Updates to Version 2.5 181
69 Updates to Version 2.4 186
70 Updates to Version 2.3 191
71 Updates to Version 2.2 197
72 Updates to Version 2.1 208
73 Updates to Version 2.0 209
74 Updates to Version 1.14 232
75 Updates to Version 1.13 235
76 Updates to Version 1.12 236
77 Updates to Version 1.11 240
78 Updates to Version 1.10 240
79 Updates to Version 1.9 242
80 Updates to Version 1.8 245
81 Updates to Version 1.7 247
82 Updates to Version 1.6 250
83 Updates to Version 1.5 264
84 Updates to Version 1.4 266
85 Updates to Version 1.3 272
86 Updates to Version 1.2 276
87 Updates to Version 1.1 280
88 Updates to Version 1.0 282

The version release dates are listed here. Versions released before the current version may be obtained by email request.

- Version 7.0 posted January 7, 2024.
- Version 6.8 posted August 9, 2023.
- Version 6.7 posted July 17, 2023.
- Version 6.6 posted May 8, 2023.
- Version 6.5 posted December 14, 2022.
- Version 6.4 posted June 8, 2022.
- Version 6.3 posted April 3, 2022.
- Version 6.2 posted March 7, 2022.
- Version 6.1 posted February 7, 2022.
- Version 6.0 posted January 6, 2022.
- Version 5.15 posted December 21, 2021.
- Version 5.14 posted November 19, 2021.
- Version 5.13 posted September 25, 2021.
- Version 5.12 posted August 1, 2021.
- Version 5.11 posted June 16, 2021.
- Version 5.10 posted May 23, 2021.
- Version 5.9 posted April 26, 2021.
- Version 5.8 posted March 26, 2021.
- Version 5.7 posted March 8, 2021.
- Version 5.6 posted January 14, 2021.
- Version 5.5 posted December 18, 2020.
- Version 5.4 posted November 17, 2020.
- Version 5.3 posted November 5, 2020.
- Version 5.2 posted October 11, 2020.
- Version 5.1 posted September 30, 2020.
- Version 5.0 posted September 15, 2020.
- Version 4.9 posted September 5, 2020.

- Version 4.8 posted August 12, 2020.
- Version 4.7 posted June 24, 2020.
- Version 4.6 posted April 13, 2020.
- Version 4.5 posted January 12, 2020.
- Version 4.4 posted January 4, 2020.
- Version 4.3 posted December 29, 2019.
- Version 4.2 posted December 5, 2019.
- Version 4.1 posted August 29, 2019.
- Version 4.0 posted August 14, 2019.
- Version 3.30 posted December 29, 2019.
- Version 3.29 posted December 5, 2019.
- Version 3.28 posted August 29, 2019.
- Version 3.27 posted August 14, 2019.
- Version 3.26 posted August 1, 2019.
- Version 3.25 posted April 29, 2019.
- Version 3.24 posted April 12, 2019.
- Version 3.23 posted April 4, 2019.
- Version 3.22 posted March 4, 2019.
- Version 3.21 posted January 21, 2019.
- Version 3.20 posted January 9, 2019.
- Version 3.19 posted November 30, 2018.
- Version 3.18 posted October 30, 2018.
- Version 3.17 posted October 19, 2018.
- Version 3.16 posted October 3, 2018.
- Version 3.15 posted September 8, 2018.
- Version 3.14 posted July 17, 2018.
- Version 3.13 posted June 7, 2018.
- Version 3.12 posted February 19, 2018.
- Version 3.11 posted February 8, 2018.

- Version 3.10 posted September 16, 2017.
- Version 3.9 posted June 18, 2017.
- Version 3.8 posted April 2, 2017.
- Version 3.7 posted February 6, 2017.
- Version 3.6 posted January 28, 2017.
- Version 3.5 posted November 28, 2016.
- Version 3.4 posted November 14, 2016.
- Version 3.3 posted September 24, 2016.
- Version 3.2 posted July 6, 2016.
- Version 3.1 posted June 28, 2016.
- Version 3.0 posted June 19, 2016.
- Version 2.5 posted May 26, 2016.
- Version 2.4 posted April 2, 2016.
- Version 2.3 posted March 10, 2016.
- Version 2.2 posted January 30, 2016.
- Version 2.1 posted January 25, 2016.
- Version 2.0 posted September 23, 2015.
- Version 1.14 posted June 7, 2015.
- Version 1.13 posted May 31, 2015.
- Version 1.12 posted April 20, 2015.
- Version 1.11 posted April 5, 2015.
- Version 1.10 posted March 10, 2015.
- Version 1.9 posted February 1, 2015.
- Version 1.8 posted January 5, 2015.
- Version 1.7 posted December 12, 2014.
- Version 1.6 posted November 25, 2014.
- Version 1.5 posted October 25, 2014.
- Version 1.4 posted September 25, 2014.
- Version 1.3 posted September 13, 2014.

- Version 1.2 posted August 29, 2014.
- Version 1.1 posted August 19, 2014.
- Version 1.0 posted August 11, 2014.

The updated files and related notes are provided for the versions in each of the ensuing sections. Each section has a list of changes that occurred to the version number mentioned in that section. Those changes were rolled up into the zip file that was posted for the next version. Files in the `Include` or `Source` folder are abbreviated (file prefix `Gte` removed) for ease of reading. Modified files are colored gold, new files are colored green, and deleted files are colored red. Source code is colored Violet.

1 Updates to Version 7.0

March 25, 2024. Modified the FindNeighbors member function to use a priority queue to improve the speed of queries. Thanks to CodeReclaimers for providing the speed-up.

GTE/Mathematics/NearestNeighborQuery.h

March 24, 2024. The classes IntpQuadraticNonuniform2 for GTE and GTL pass the same unit tests, and those results were verified by Mathematica for the GTL class. The rendering in the interpolation sample application clearly showed the derivatives did not match at the center triangle of the 4 triangles. It turns out that that sample code itself had a bug. The SampleMesh class initializes mAdjacencies to have the triangle indices for neighbors. For the triangle $\langle \boldsymbol{V}_1, \boldsymbol{V}_4, \boldsymbol{V}_3 \rangle$, the adjacencies were assigned as $\langle 0, 1, 2 \rangle$. The assignment needed to be $\langle 1, 2, 0 \rangle$. After this modification, the rendered surface is visually smooth. The window client size was 512×512, but I made it larger for a better view of the renderings. It is now 768×768.

GTE/Samples/Mathematics/Interpolation2D/Interpolation2DMain.cpp
GTE/Samples/Mathematics/Interpolation2D/Interpolation2DWindow2.cpp

March 23, 2024. My template instantiation system needed serious modification for when I deprecated a class and provided a new implementation. The system failed to instantiate the new implementations. I discovered this when attempting to backport the IntpQuadraticNonuniform2 class (the Cendes–Wong algorithm) from GTL to GTE. The GTL code has been thoroughly unit tested and verified with Mathematica. There are no sample applications that exercise those new implementations, which was clear when the code would not compile—a serious oversight on my part. I have made changes to various classes to get the template instantiation to compile the new code. I will post the backported GTL code for IntpQuadraticNonuniform2 after I unit test the derived classes (IntpSphere2 and IntpVectorField2).

GTE/Mathematics/Delaunay2Mesh.h
GTE/Mathematics/Delaunay3Mesh.h
GTE/Mathematics/IntpLinearNonuniform2.h
GTE/Mathematics/IntpLinearNonuniform3.h
GTE/Mathematics/IntpSphere2.h
GTE/Mathematics/IntpVectorField2.h

March 13, 2024. Ported the Hermite interpolations from GTE to GTL. Verified the systems of equations and solutions that produce the coefficients of the polynomials, both using code that generates code for the system and by using Mathematica to solve symbolically the numerous cases. Unit tests for the GTL code are more rigorous than those for GTE, so the GTL code and unit tests were back-ported to GTE. The SmoothLatticeInterpolation.pdf document has been

GTE/Mathematics/HermiteCubic.h
GTE/Mathematics/HermiteQuintic.h
GTE/Mathematics/HermiteBicubic.h
GTE/Mathematics/HermiteBiquintic.h
GTE/Mathematics/HermiteTricubic.h
GTE/Mathematics/HermiteTriquintic.h

February 25, 2024. Added a new file for fitting a parabola to points $(x, f(x))$. The document Least Squares Fitting of Data by Linear or Quadratic Structures has been updated with a section describing the algorithm.

GTE/Mathematics/ApprParabola2.h

Fixed minor issues after porting the code and unit tests to GTL: Removed test code. The isUnique value was set to false when the intersection of intervals is a nondegenerate interval. Instead, isUnique must be set to true when the w-values are the same at the interval endpoints, false otherwise.

GTE/Mathematics/InscribedFixedAspectRectInQuad.h

February 24, 2024. Implemented the maximum-area, fixed-aspect-ratio, axis-aligned rectangle inscribed in a convex quadrilateral. See the document Largest Fixed-Aspect, Axis-Aligned Rectangle

GTE/Mathematics/InscribedFixedAspectRectInQuad.h

February 19, 2024. Modified the code based on Lev A. Melnikovsky's mathematical formulation to avoid computing eigenvectors until needed and to avoid numerical discontinuities when computing eigenvectors.

GTE/Samples/Imagics/ExtractRidges/ExtractRidgesConsole.cpp

February 1, 2024. I had added the new files BVTree.h, BVTreeOfTriangles.h, and AABBBVTreeOfTriangles.h to the MSVS 2022 project and filter files. The MSVS 2019 project and filter files needed the new files added to them.

GTE/GTMathematics.v16.{vcproj,vcsproj.filters}

January 31, 2024. The GTE LInfinityNorm function was missing a std::fabs applied to M[0]. The GTL version of this function is correct.

GTE/Mathematics/Matrix.h

January 29, 2024. The function operator() was missing the test for zOrder being 5 or smaller. The inner-most loop had k bounded by 4 when it should have been 6.

GTE/Mathematics/HermiteTriquintic.h

January 20, 2024. Added abstract classes and derived class for constructing a bounding volume tree for geometric primitives. These will eventually replace the current OBB tree classes. The class ABBBVTreeOf-Triangles was needed for a contracting project.

GTE/Mathematics/BVTree.h
GTE/Mathematics/BVTreeOfTriangles.h
GTE/Mathematics/AABBBVTreeOfTriangles.h

January 18, 2024. The finite-difference approximation to the second-order derivative $\partial^2 f(x,y)/\partial y^2$ had a typographical error. Also, to avoid sign changes artificially generated because the numerical eigensolver can be inconsistent in returning eigenvectors: At one point an eigenvector U is returned but at a nearby point an eigenvector is return that is approximately $-U$. Thanks to Lev A. Melnikovsky for reporting the problems and fixing them.

GTE/Samples/Imagics/ExtractRidges/ExtractRidgesConsole.cpp

January 7, 2024. Posted Geometric Tools Engine 7.0. No updates yet.

2 Updates to Version 6.8

January 7, 2024. Removed the update PDFs from github. Posting links from the github repository page to the update PDFs at the Geometric Tools website.

Gte6UpdateHistory.pdf
GteFullUpdateHistory.pdf

Added support for vertical synchronization on Linux distributions by querying for the glXSwapIntervalEXT extension. This allows per-application specification of the synchonization interval via the DisplayColor-Buffer(uint32_t) graphics interface call. For an NVIDIA graphics card, the X server is installed with the default not to sync to vertical retrace. You can turn it on globally using the X server tool.

GTE/Graphics/GL45/GLX/GLXEngine.cpp
GTE/Graphics/GL45/GLX/GLXExtensions.cpp

The positions structured buffer was declared as float3 in the vertex shader. However, it is constructed as a float4 in the application. Modified the shader to use float4.

GTE/Samples/Physics/MassSprings3D/Shaders/DrawUsingVertexID.vs.hlsl

January 6, 2024. The application class member mContinueSolving was uninitialized. The debug build worked for me but not the release build. Added class member initialization to the application constructor. When SINGLE_STEP is enabled, the code does not compile because of a porting error. It had a Wild Magic Vector2d exposed but needed to be a GTE Vector2<double>.

GTE/Samples/Physics/RoughPlaneParticle1/RoughPlaneParticle1Window2.cpp

January 6, 2024. The application class member mContinueSolving was uninitialized. The debug build worked for me but not the release build. Added class member initialization to the application constructor. When SINGLE_STEP is enabled, the code does not compile because of a porting error. It had a Wild Magic Vector2d exposed but needed to be a GTE Vector2<double>.

GTE/Samples/Physics/RoughPlaneParticle1/RoughPlaneParticle1Window2.cpp

The preprocessor symbol declaration SINGLE_STEP was missing from the files.

GTE/Samples/Physics/RoughPlaneFlatBoard/RoughPlaneFlatBoardWindow2.cpp
GTE/Samples/Physics/RoughPlaneThinRod2/RoughPlaneThinRod2Window2.cpp

The sample was missing the Visual Code workspace.

GTE/Samples/Distance/DistanceLine2Box2.code-workspace
GTE/Samples/Distance/DistanceOrientedBoxConeFrustum.code-workspace
GTE/Samples/Intersection/IntersectTriangleCylinder/IntersectTriangleCylinder.code-workspace

The sample Visual Code workspace had the wrong name.

GTE/Samples/Mathematics/GeodesicEllipsoid.code-workspace
GTE/Samples/Mathematics/GeodesicHeightField.code-workspace

Visual Code on Linux distributions warned that cmake_minimum_required statements needed to occur before project statements.

CMakeLists.txt (in all occurrences)

January 5, 2024. Modifications based on Microsoft Visual Studio 2022 code analysis warnings about not initializing native class members in the constructor initializer list. (The code analyzer needs improvement, checking whether those members are set appropriately in the body of the constructor.)

GTE/Mathematics/ApprTorus3.h
GTE/Samples/Physics/MassSprings3D/GpuMassSpringVolume.{h,cpp}
GTE/Samples/SceneGraphs/BlendedAnimations/BipedManager.{h,cpp}

January 4, 2024. On Fedora 38, gcc 13.12.1 warned about unused stophere which I had added for debugging. I had added (void)stophere after assignment to avoid the warning on Visual Studio and earlier versions of gcc. I replaced this variable by a LogAssert.

GTE/Samples/Intersection/IntersectLineRectangle/IntersectLineRectangleWindow3.cpp

January 3, 2024. The new implementation had multiple problems. The code was complicated enough that more care was needed to get it right. My previous attempts were a poorly handed response to a request for a new feature. I stepped through the code for the latest version using a debugger, specifically for SimpleGraph0.txt of the sample application. The code coverage hits all the relevant blocks. The unit tests for the simpler configurations pass.

GTE/Mathematics/MinimalCycleBasis.h
GTE/Samples/Mathematics/Geometrics/MinimalCycleBasis/MinimalCycleBasisWindow2.{h,cpp}

January 2, 2024. Running my instantiation tool on Linux distributions, there were various issues.

GTE/Mathematics/MinimalCycleBasis.h
GTE/Mathematics/ApprParaboloid3.h

January 1, 2024. Restored the old paraboloid fitting algorithm so that you can choose between that one and the new robust one. I removed the mathematical comments and provided a web link to the least-squares fitting PDF at the Geometric Tools website. The PDF has been modified to describe both algorithms.

GTE/Mathematics/ApprParaboloid3.h

December 30, 2023. Subtracting out the mean from the inputs for a robust floating-point algorithm.

GTE/Mathematics/ApprParaboloid3.h

Revised the implementation of MinimalCycleBasis. The code now computes isolated vertices, filaments, and basis cycles. It also uses exact arithmetic internally to guarantee correctness of the computations. I added more unit tests for the revision than for the previous code version.

GTE/Mathematics/MinimalCycleBasis.h

Fixed bugs that were discovered when triangulating a polygon tree. The Vertex objects of the VertexList had state information from the previous triangulation that affected DoEarClipping. The state is now reset to the constructor values in the Vertex constructor. The polygon-tree triangulation also was not accumulating the triangles for the multiple DoEarClipping calls.

GTE/Mathematics/TriangulateEC.h

December 24, 2023. The test in Extract for adding a tree to the forest needed another Boolean condition. I also realized that the filament extraction will not work for general subgraphs of filaments. I am working to resolve that. The extraction output uses a set of pointers to a Vertex structure, so the output can vary between runs because of the sorting order implied by the set of structures. I am also working to resolve that. See issue #80 at my online github repository.

GTE/Mathematics/MinimalCycleBasis.h

December 22, 2023. The comments indicated that the filaments are extracted by the code. They are removed from the graph but not made available to the user. I added the ability to extract the filaments in case the caller wants them. The file is relatively old, so I took the opportunity to upgrade it to my current design goals. The constructor is a simple one, no longer mimicking a function object. The Extract function does the cycle and filament extraction. I also added a new template parameter which is the type of the vertex indices. The type was int32_t but now can be any signed or unsigned integer type of size at least 2 bytes. Thanks to Scott Johnson for (once again) reporting problems such as these.

GTE/Mathematics/MinimalCycleBasis.h
GTE/Samples/Geometrics/MinimalCycleBasis/MinimalCycleBasisWindow2.{h,cpp}

December 16, 2023 This is a major rewrite of the previous version. The code is easier to read. I addressed a several technical support items. The duplicate-vertex map has been removed. The PDF Triangulation by Ear Clipping is a major rewrite and explains why the algorithm is self-sorting regarding bridges that share the same outer-polygon vertex. Various unit tests appear to show that the algorithm is performing correctly.

GTE/Mathematics/TriangulateEC.h

December 4, 2023. The nonmanifold test for a shared edge having opposite orderings for the two triangles sharing the edge was flawed. The edge object comes from an EdgeKey<false> object so that the edge is $\langle v_0, v_1 \rangle$ with $v_0 < v_1$. The actual ordering depends on the specified triangle vertex order. Instead, the comparison has to be made for the shared edge of the already existing triangle edge−>T[0] because the vertices for that triangle are ordered.

GTE/Mathematics/ETManifoldMesh.h

December 3, 2023. In operator(), the array reflect is declared to have 3 Boolean elements but only the first 2 are explicitly initialized. According to the C++ standard, the unspecified element is zero-initialized, set to false in this case. I added the third initializer to make it clear that is what was intended.

GTE/Mathematics/DistPlane3CanonicalBox3.h

December 2, 2023. Added new intersection queries for rectangle and linear components (line, ray, segment). Added a sample application to illustrate and for testing.

GTE/Mathematics/IntrLine3Rectangle3.h
GTE/Mathematics/IntrRay3Rectangle3.h
GTE/Mathematics/IntrSegment3Rectangle3.h
GTE/Samples/Intersection/IntersectLineRectangle/*
GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}
GTE/BuildAll*.*

Fixed the minor version numbers to show current version is 6.8. Added the new sample to the intersection CMakeLists.txt.

GTE/CMakeLists.txt
GTE/Samples/CMakeLists.txt

GTE/Samples/Distance/CMakeLists.txt
GTE/Samples/Geometrics/CMakeLists.txt
GTE/Samples/Graphics/CMakeLists.txt
GTE/Samples/Imagics/CMakeLists.txt
GTE/Samples/Intersection/CMakeLists.txt
GTE/Samples/Mathematics/CMakeLists.txt
GTE/Samples/Physics/CMakeLists.txt
GTE/Samples/SceneGraphs/CMakeLists.txt

November 27, 2023. Reimplemented the query for distance between two circles in 3D. The previous version used only floating-point arithmetic. The rounding errors when computing derivatives of the degree-8 polynomial were causing extreme inaccuracy in the results. The new code now uses a mixture of floating-point arithmetic and rational arithmetic. The degree-8 polynomial and its derivatives are created using rational arithmetic. Bisection is performed using floating-point arithmetic, but with some initial rational arithmetic to determine sign changes in a polynomial at the endpoints of a potential root-bounding interval.

GTE/Mathematics/DistCircle3Circle3.h

Added multithreading to the general-degree polynomial root estimator to improve the performance.

GTE/Mathematics/RootsGeneralPolynomial.h

November 26, 2023. Based on reader feedback for FastOrthogonalComplement.pdf, I reordered the vectors in the signatures of the functions listed in the pseudocode and in the actual code (in Vector3.h). This is the usual convention I have for writing code (inputs first, outputs last). I also modified the actual code function name to FastComputeOrthogonalComplement.

GTE/Mathematics/Vector3.h

November 21, 2023. After thinking harder about the new Vector3 function ComputeOrthogonalComplement, I was able to avoid the square root operation. Given a unit-length vector, you can compute the orthogonal complement with only arithmetic operations. A new PDF has been posted to the website: FastOrthogonal-Complement.pdf.

GTE/Mathematics/Vector3.h

November 20, 2023. The RootsPolynomial<T>::Find function suffered from significant floating-point rounding errors in the computation of the coefficients of the polynomial derivatives. The class allows for type T to be rational, but the bisection effectively does not converge because the number of bits per rational number is excessively large. This class is now deprecated. The low-degree root finders are now in RootsLinear.h, RootsQuadratic.h, RootsCubic.h, and RootsQuartic.h. Three of these files are listed here because a comment mentioned using Cauchy's bound when in fact the code uses Lagrange's bound.

GTE/Mathematics/RootsQuadratic.h
GTE/Mathematics/RootsCubic.h
GTE/Mathematics/RootsQuartic.h
GTE/Mathematics/RootsPolynomial.h

The replacement for RootsPolynomial<T>::Find is RootsGeneralPolynomial<T>::Solve. The algorithm is effectively the same as the deprecated Find, but it uses a mixture of floating-point arithmetic and rational arithmetic. The coefficients of the polynomial derivatives are computed using rational arithmetic to avoid the rounding errors that plagued the deprecated code. When the bisection code consumes these polynomials, rational evaluation is used to determine sign information at the endpoints of an interval. If bisection must occur, the rational coefficients are converted to floating-point, and the resulting rounding errors are significantly smaller than that of the deprecated code. The floating-point-based bisection is fast, so the performance is reasonable in order to get increased accuracy of the root estimates.

GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}
GTE/Mathematics/RootsGeneralPolynomial.h

Added a new function ComputeOrthogonalComplement with a signature different from the other same-named function. The new function computes the basis by computing a unit-length quaternion for a rotation matrix whose last column is a user-specified unit-length vector. The other two columns are extracted from the rotation matrix.

GTE/Mathematics/Vector3.h

The Rotate function for a 4-tuple vector did not compile because it uses a Vector3-based Cross product function. The Vector3 quantities were replaced by Vector4 quantities, but the header file Vector3.h still needs to be included to access the cross product of two 4-tuple vectors.

GTE/Mathematics/Quaternion.h

Added a member accessor to get a const reference to the polynomial coefficients.

GTE/Mathematics/Polynomial1.h

Added an exception to the Insert function that detects whether the vertex ordering for two edges is opposite for the two triangles sharing the edge. If the ordering is not opposite, the mesh is not manifold. The test is exposed by default (mThrowOnNonmanifoldInsertion is initially true), but you can disable the test by setting the aforementioned Boolean to false.

GTE/Mathematics/ETManifoldMesh.h

The GetGradient function had a typecast of a float to a size_t and then assigned it to an int32_t. The typecast was changed to a int32_t. Fixed Ubuntu gcc compiler warnings/errors: missing typename when declaring topology, 3 loop variables were size_t but needed to be IndexType.

GTE/Mathematics/SurfaceExtractorMC.h

Fedora gcc complained about undefined int32_t (but Ubuntu gcc did not). Added an include of cstdint.h.

GTE/Mathematics/HermiteBiquintic.h
GTE/Mathematics/HermiteTricubic.h
GTE/Mathematics/HermiteTriquintic.h

The following warnings/errors occur using LVM - clang-cl in Microsoft Visual Studio 2022 version 17.8.0. After these modifications, the Intel C++ Compiler 2023 had no warnings/errors.

An include of Functions.h is required; otherwise, the block of functions starting with atandivpi become infinitely recursive. The definition of atandivpi whose input is IEEEBinary16 makes a call to atandivpi whose argument is typecast to float. This works as long as the definition of atandivpi in Functions.h is accessible. Without it, an implicit IEEEBinary16 constructor call is made from the float input, thus the infinite recursion. The constructor with input of type double has variable temp that is tagged as not initialized. It is on the next line, but I moved that code into the union initialization.

GTE/Mathematics/IEEEBinary16.h

Two declarations of Vector3<T> delta occurred in the code, but the variables are unused. They have been removed.

GTE/Mathematics/DistLine3Circle3.h

Function ConstrainedTriangulate had a declaration of counter that was used for debugging. Even though it is incremented in the code, the value is not used anywhere else and is flagged as such. It has been removed.

GTE/Mathematics/TriangulateCDT.h

The operator() function declared and initialized numZero. Even though it is incremented in the code, the value is not used anywhere else and is flagged as such. They have been removed.

GTE/Mathematics/IntrHalfspace3Segment3.h
GTE/Mathematics/IntrHalfspace3Triangle3.h

The LLVM compiler also generates warnings about Intel intrinsics files emmintrin.h and xmmintrin.h. The projects do not include these headers explicitly, so they must be exposed somewhere in Microsoft's header files.

October 21, 2023. Added a code to save binary STL files.

GTE/Mathematics/STLBinaryFile.h

October 20, 2023. Added a file loader for binary STL files.

GTE/Mathematics/STLBinaryFile.h

October 11, 2023. Reimplemented the Quaternion<T>::Rotate functions to use fewer arithmetic operations. Thanks to Robert Eisele for sending me a link to his webpage about the algorithm; the link is provided in the header file.

GTE/Mathematics/Quaternion.h

September 30, 2023. Toggling between double and BSRational<UIntegerAP32> required commenting and uncommenting some code. I added a preprocessor define so that you can toggle by exposing or hiding this define.

GTE/Samples/Geometrics/PolygonBooleanOperations/PolygonBooleanOperationsWindow2.{h,cpp}

September 28, 2023. Modified the sample application to use exact rational arithmetic if so desired. BSPPolygon2<T> already compiles when T is BSRational<UIntegerAP32>.

GTE/Samples/Geometrics/PolygonBooleanOperations/PolygonBooleanOperationsWindow2.{h,cpp}

Fixed a problem with segment-arc and ray-arc find-intersection queries that were caused by the line-circle find-intersection query setting only one of the result.parameter values when the line and circle are tangent. It is necessary to set both parameter values to the same number, effectively a degenerate interval that is used in interval-interval find-intersection queries used in segment-arc and ray-arc. Thanks to owai on github for reporting the problem, diagnosing it, and fixing it.

GTE/Mathematics/IntrLine2Circle2.h

Removed the deprecation comment about the Contains function without the epsilon argument. This function is still useful when you know the point is on the arc so that you do not need the epsilon value.

GTE/Mathematics/Arc2.h

September 25, 2023. The operator< comparison had an incorrect comparison in it, using origin instead of the correct normal.

GTE/Mathematics/Hyperplane.h

September 25, 2023. Yet more modifications to surface extraction. The index type for the indices array were hard-coded as int32_t. Some applications have different types for the indices. The MarchingCubes and SurfaceExtractionMC classes now have template parameters IndexType to allow the user to choose whatever type is natural to the application. I back-ported changes made to the GTL version of MarchingCubes; for example, int32_t index[8] is now std::array<IndexType, 8> index.

GTE/Mathematics/MarchingCubes.h
GTE/Mathematics/SurfaceExtractorMC.h
GTE/Samples/Physics/BouncingBall/DeformableBall.cpp
GTE/Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow3.h

September 24, 2023. In SurfaceExtractorMC.h, the test code used only a positive perturbation and the output is correct. When the perturbation is negative, the order of sign testing needed to be modified to ensure the output is correct. The BouncingBall physics sample uses SurfaceExtractorMC.h, so it needed a modification to add the perturbation parameter.

GTE/Mathematics/SurfaceExtractorMC.h
GTE/Samples/Physics/BouncingBall/DeformableBall.cpp

September 23, 2023. When the single-voxel Extract returns false, the full-image extraction was supposed to ignore the voxel. Instead it deleted the entire mesh, which is a disastrous bug. The unit tests verified only that the Marching Cubes 256 triangle-lookup table entries are correct, and none of these provided code coverage that would have found the bug. Also, floating-point rounding errors in mesh vertex construction led to small sets of vertices, each set containing vertices that are nearly the same (but should have been the same without the rounding errors). MakeUnique is unable to identify these as duplicates. I modified the single-voxel Extract function to generate the same vertex on an edge shared by multiple voxels, using a consistent ordering of the voxel corners to ensure that the all constructions of the vertex have the same rounding errors. Finally, the interfaces were modified to have a new parameter perturb that allows you to avoid the ignoring of voxels for which at least one corner of the voxel is exactly the input level value. Thanks to Scott Johnson for reporting these problems.

GTE/Mathematics/SurfaceExtractorMC.h

September 17, 2023. If the mesh is not orientable, GetBoundaryPolygons had an infinite loop when trying to traverse a triangle strip the connects two consecutive boundary edges. When not orientable, two adjacent triangles are repeatedly visited. I added a trap for this and throw an exception.

GTE/Mathematics/ETManifoldMesh.h

September 10, 2023. Added new file for distance queries between 2 circles in 2D.

GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}
GTE/Mathematics/DistCircle2Circle2.h

Added includes of C++ Standard Library headers to ensure they are always exposed to the file.

GTE/Mathematics/DistLine2Arc2.h
GTE/Mathematics/DistRay2Arc2.h
GTE/Mathematics/DistSegment2Arc2.h

August 28, 2023. Revised comments. Fixed bugs that were exposed when unit testing ray-arc and segment-arc distance queries. Feature request: Added new distance queries. Sorry for the delay. I struggle with 2-dimensional mathematics. I know I have problems with 3-dimensional mathematics because I get lost walking around my neighborhood. I seem to be most comfortable in 7-dimensional space.

GTE/Mathematics/DistLine2Circle2.h
GTE/Mathematics/DistRay2Circle2.h
GTE/Mathematics/DistSegment2Circle2.h
GTE/Mathematics/DistLine2Arc2.h
GTE/Mathematics/DistRay2Arc2.h
GTE/Mathematics/DistSegment2Arc2.h

August 27, 2023. The comments in these files indicate the direction vectors do not have to be unit length. This was not true in earlier versions of GTE; that is, the directions had to be unit length. I had modified the comments at one time because I was removing the unit-length constraint. Unfortunately, the code was not modified, so only unit-length vectors worked. My unit tests passed anyway because they all used unit-length directions. The constraint is now removed. The direction vectors do not have to be unit length, the code was corrected, and the unit tests were updated to handle non-unit-length vectors. The line-segment distance query in DistPointSegment.h is correct.

GTE/Mathematics/DistPointLine.h
GTE/Mathematics/DistPointRay.h

August 26, 2023. When the line intersects the circle in two distinct points, the parameter members of result must be increasingly ordered. They were not always, so a test-and-swap had to be added. I also fixed old comments about the Result struct.

GTE/Mathematics/Dist2Line2Circle2.h

Added a distance query for segment-circle in 2D.

GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}
GTE/Mathematics/Dist2Segment2Circle2.h

August 24, 2023. The GTMathematicsGPU solutions and projects duplicated the GUID in MSVS 2019 and MSVS 2022. The MSVS 2019 GUID was modified. The GenerateMesh v17 projects were missing references to the GTMathematicsGPU projects (which did not affect the compilation).

GTE/BuildAll.v16.sln
GTE/BuildAllDX11.v16.sln
GTE/BuildAllGL45.v16.sln
GTE/GTMathematicsGPU.v16.sln
GTE/GTMathematicsGPU.v16.vcxproj
GTE/Samples/Geometrics/GenerateMeshUVs/GenerateMeshUVsDX11.v16.sln
GTE/Samples/Geometrics/GenerateMeshUVs/GenerateMeshUVsDX11.v16.vcxproj
GTE/Samples/Geometrics/GenerateMeshUVs/GenerateMeshUVsGL45.v16.sln
GTE/Samples/Geometrics/GenerateMeshUVs/GenerateMeshUVsGL45.v16.vcxproj
GTE/Samples/Physics/BlownGlass/BlownGlassDX11.v16.sln
GTE/Samples/Physics/BlownGlass/BlownGlassDX11.v16.vcxproj
GTE/Samples/Physics/BlownGlass/BlownGlassGL45.v16.sln
GTE/Samples/Physics/BlownGlass/BlownGlassGL45.v16.vcxproj
GTE/Samples/Physics/Fluids2D/Fluids2DDX11.v16.sln
GTE/Samples/Physics/Fluids2D/Fluids2DDX11.v16.vcxproj
GTE/Samples/Physics/Fluids2D/Fluids2DGL45.v16.sln
GTE/Samples/Physics/Fluids2D/Fluids2DGL45.v16.vcxproj
GTE/Samples/Physics/Fluids3D/Fluids3DDX11.v16.sln
GTE/Samples/Physics/Fluids3D/Fluids3DDX11.v16.vcxproj
GTE/Samples/Physics/Fluids3D/Fluids3DGL45.v16.sln
GTE/Samples/Physics/Fluids3D/Fluids3DGL45.v16.vcxproj

Removed MSVS 2022 warnings about uninitialized arrays (which clearly are initialized in the code).

GTE/Mathematics/Image2.h
GTE/Mathematics/Image3.h

August 20, 2023. This is a new file that accompanies the revised version of the document MovingAlongCurveSpecifiedSpeed.pdf. It contains an implementation of the algorithms described in the PDF.

GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}
GTE/Mathematics/ReparameterizeByArclength.h

3 Updates to Version 6.7

August 8, 2023. After refactoring Math.h, a significant number of GTE files no longer compiled. The files required headers from the C++ Standard Library, but some of those headers were accessed indirectly by including other GTE files. I fixed those, but then discovered that my instantiation tool had a significant number of source files that included ArbitraryPrecision.h in order to instantiate templates based on numeric types float, double, BSNumber, and BSRational. The arbitrary precision header itself exposes standard library files that allowed some of the float-based and double-based templates, but when the arbitrary precision header is hidden, several of these templates failed to compile.

To avoid this mess, I modified nearly every GTE mathematics header file to explicitly include standard library headers for symbols that the header file references. This should protect the distribution from major failure if I refactor headers in the future. I also modified the instantiation tool to allow me to compile with or without arbitrary precision headers. I had already done this in my GTL development track. The number of modified files is so large that I am not listing them here.

Because the number of modified files was so large, I also unified my top-level commenting style. General comments at the top of the header about the file now occur after the #pragma statement but before #include-s of the headers that the file requires. I had already done this in my GTL development track.

Items modified during the refactor with additional needs are listed below.

New files for computing distance between a line or ray and a circle. The other cases for segment-circle, line-arc, ray-arc, and segment-arc will be posted soon.

GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}
GTE/Mathematics/DistLine2Circle2.h
GTE/Mathematics/DistRay2Circle2.h

Added initialization of some local variables to avoid MSVS code analysis complaint.

GTE/Mathematics/BSplineSurfaceFit.h
GTE/Mathematics/BSplineVolume.h
GTE/Mathematics/ContEllipse2.h
GTE/Mathematics/ContEllipsoid3.h
GTE/Mathematics/Matrix.h
GTE/Mathematics/FPInterval.h
GTE/Mathematics/GMatrix.h

GTE/Mathematics/IntpThinPlateSpline2.h
GTE/Mathematics/IntpVectorField2.h
GTE/Mathematics/Matrix.h
GTE/Mathematics/NURBSVolume.h
GTE/Mathematics/RootsBisection1.h
GTE/Mathematics/SurfaceExtractorMC.h
GTE/Mathematics/SWInterval.h

Replaced a move operation by a copy operation to avoid MSVS code analysis complaint about the moved object now being invalid.

GTE/Mathematics/Polynomial1.h

The class Quaternion is a client of Chebyshev ratios. The interface for ChebyshevRatio changed, so the quaternion code needed updating.

GTE/Mathematics/Quaternion.h

Commented out debug preprocessor symbols. If you want any of these, expose them again or add them to your global preprocessor defines.

GTE/Mathematics/APInterval.h
GTE/Mathematics/ApprQuery.h
GTE/Mathematics/BitHacks.h
GTE/Mathematics/BSNumber.h

Deleted the GTE_NO_LOGGER macro, exposing the LogAssert statements. This were the only occurrences of the no-logger control.

GTE/Mathematics/ApprPolynomialSpecial2.h
GTE/Mathematics/ApprPolynomialSpecial3.h
GTE/Mathematics/ApprPolynomialSpecial4.h

Added pragma block for MSVS 2022 in the function Convert in the case of the rounding mode FE_TONEAREST. The pragma disables a code analysis warning. This code had been tested and analyzed previously and works fine. I do not understand the code analysis message, which I have added to the pragma block as a comment.

GTE/Mathematics/BSNumber.h

Moved the comparison operators from external scope to class scope to be consistent with the other primitives having class-scope operators.

GTE/Mathematics/CanonicalBox.h

The gcc compilers on Linux complained about a potential uninitialized array (MSVS 2022 did not). The complaint is correct. The GetVertices functions return false if the Delaunay dimension is not 2. The GetBarycentrics functions return false if the triangle index t is invalid or if the ComputeBarycentrics function fails.

In all these cases, the output arrays are uninitialized. First, I removed the test for dimension 2 and added an exception in the constructor if the input Delaunay object is not 2-dimensional. Second, I added initialization of the arrays if failure occurs, setting the components all to zero and then returning false.

GTE/Mathematics/Delaunay2Mesh.h
GTE/Mathematics/Delaunay3Mesh.h

Modified std::abs calls to std::fabs calls.

GTE/Mathematics/IntrAlignedBox3Cone3.h

Converted some loop counters from int32_t to size_t to avoid MSVS code analysis complaints about 32-bit arithmetic performed on index lookups possibly causing overflow (it does not).

GTE/Mathematics/NaturalCubicSpline.h
GTE/Mathematics/NaturalQuarticSpline.h
GTE/Mathematics/UIntegerAP32.h

Replace an auto statement by the explicit type for a std::function object to avoid a MSVS 2022 code analysis complaint about not being able to determine whether a copy or const reference is asked for.

GTE/Mathematics/RiemannianGeodesic.h

Replaced C-style arrays by std::array.

GTE/Mathematics/SurfaceExtractorMC.h

Added precision counter for fused-multiply-add.

GTE/Tools/PrecisionCalculator/PrecisionCalculator.cpp

August 3, 2023. Factored Math.h into Constants.h, Functions.h, and TypeTraits.h. The Math.h file has been removed from the distribution. The constants header contains a small number of commonly used constants. The functions header contains implementations for IEEE recommended functions for convenience. These are for float and double. The type traits header contains traits for has_division and is_arbitrary_precision. Some other traits were added, but they are currently only in use in the GTL version. They are useful for eliminating the verbose nature of std::enable_if for conditionally selecting implementations based on type (floatting-point or rational). The removal of Math.h led to many file modifications. Most of the clients of this header simply needed <cmath>.

GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}
GTE/Mathematics/Math.h
GTE/Mathematics/Constants.h
GTE/Mathematics/Functions.h
GTE/Mathematics/TypeTraits.h

I added support for std::remainder in Functions.h and in the rational arithmetic classes. As it turns out, the factoring of Math.h led to discovery of some bugs. BSNumber.h and BSRational.h implemented the arbitrary precision type trait, setting the value to true. These were both missing a leading underscore, so the compiler thought they were new primary templates rather than implementations of the primary templates defined in TypeTraits.h.

GTE/Mathematics/BSNumber.h
GTE/Mathematics/BSRational.h

The majority of Math.h consisted of special constants for the function estimators such as ACosEstimate.h among others. The constants were moved into the estimators files themselves in GTL together with a reduction in code. I back-ported the files from GTL to GTE. The interfaces have changed for the estimators. They were template classes but are now template functions. A couple of new files were added, one for computing SLERP and one for computing Chebyshev coefficients that occur in the SLERP formula, $\sin((1-t)\theta)/\sin(\theta)$ and $\sin(t\theta)/\sin(\theta)$.

GTE/Mathematics/ACosEstimate.h
GTE/Mathematics/ASinEstimate.h
GTE/Mathematics/ATanEstimate.h
GTE/Mathematics/ChebyshevRatio.h
GTE/Mathematics/ChebyshevRatioEstimate.h
GTE/Mathematics/CosEstimate.h
GTE/Mathematics/Exp2Estimate.h
GTE/Mathematics/ExpEstimate.h
GTE/Mathematics/InvSqrtEstimate.h
GTE/Mathematics/Log2Estimate.h
GTE/Mathematics/LogEstimate.h
GTE/Mathematics/RemezAlgorithm.h
GTE/Mathematics/RotationEstimate.h
GTE/Mathematics/SinEstimate.h
GTE/Mathematics/Slerp.h
GTE/Mathematics/SlerpEstimate.h
GTE/Mathematics/SqrtEstimate.h
GTE/Mathematics/TanEstimate.h

August 2, 2023. Added the ability to discard code for fused-multiply-add (std::fma) if you know your target machine does not have floating-point hardware support for that instruction. The default is to compile std::fma. To discard this and fall back to x*y+z using 2 floating-point operations, define the preprocessor symbol GTE_DISCARD_FMA. The mechanism is simply to provide a function gte::FMA whose body is controlled by the preprocessor symbol.

GTE/Mathematics/Math.h
GTE/Mathematics/BSNumber.h
GTE/Mathematics/BSRational.h
GTE/Mathematics/RootsQuadratic.h
GTE/Mathematics/RootsCubic.h
GTE/Mathematics/IntrEllipse2Ellipse2.h

July 29, 2023. The index i1 was compared to the upper-bound loop counter for contiguous-open segment meshes. It needed to be i0.

GTE/Mathematics/SegmentMesh.h

July 28, 2023. Modified the interfaces to allow selection between using bisection or using the closed-form equations for root estimation. Updated the unit tests for code coverage of either selection.

GTE/Mathematics/RootsQuadratic.h
GTE/Mathematics/RootsCubic.h
GTE/Mathematics/RootsQuartic.h

Minor changes to avoid Visual Studio 2022 code analysis warnings.

GTE/Mathematics/UIntegerAP32.h

This file is cursed. Last check-in, I merged a line from GTL to GTE. The query result is stored in a struct Output in GTL but the GTE query result is stored in a struct Result. I then failed to run my instantiation tools which would have trapped the error.

GTE/Mathematics/DistLine3Circle3.h

July 23, 2023. The code ported to GTL had a unit-test failure for Robust function in the case that the circle center was not the origin. The GTE unit tests did not contain such a case. One block of Robust code was missing the translation of the closest line point to the original coordinate system.

GTE/Mathematics/DistLine3Circle3.h

4 Updates to Version 6.6

July 17, 2023. Added new files for robust root estimation for polynomials of degree 1, 2, 3, and 4. These use exact rational arithmetic for theoretically correct root classification, just as RootsPolynomial does. Rather than call standard mathematics library functions for the closed-form root equations, bisection is used to avoid magnification of rounding errors inherent in power functions. The document LowDegreePolynomialRoots.pdf has a major revision to be more clear about the root estimation. This includes avoiding some of the logic tests mentioned at the Wikipedia page for quartic polynomials. The code has been unit tested for 100% coverage.

GTE/Mathematics/PolynomialRoots.h
GTE/Mathematics/RootsLinear.h
GTE/Mathematics/RootsQuadratic.h (re-release)
GTE/Mathematics/RootsCubic.h
GTE/Mathematics/RootsQuartic.h

July 10, 2023. Running a ConsoleApplication-derived object on a laptop with Intel Integrated Graphics (Iris Xe/UHD) and an NVIDIA GeForce RTX 3050, the Intel graphics is identified as the most powerful adapter because that adapter is associated with the primary display. It is not the most powerful. I discovered this when the Intel DLL igc64.dll threw two exceptions,

```
MONZA::DdiThreadingContext<MONZA::AdapterTraits_Gen12LP>::msg_end
MONZA::IgcThreadingContext<MONZA::AdapterTraits_Gen12LP>::msg_end
```

during shutdown of the DX11 graphics. I had added code to WindowApplication-derived objects some time ago to choose a discrete adapter (if available) before choosing the Intel driver. That same code has now been added to the console-based application layer.

GTE/Applications/MSW/ConsoleSystem.cpp
GTE/Applications/MSW/WindowSystem.cpp

Fixed a bug. The clamping of the infinite-cylinder closest point to the finite-cylinder closest point used height h instead of $h/2$. The unit tests associated with these cases were also fixed.

GTE/Mathematics/DistPoint3Cylinder3.h

June 17, 2023. Fixed a bug in DoQueryOneZero where sMin was set incorrectly in the case tHat > +h/2.

GTE/Mathematics/IntrCanonicalBox3Cylinder3.h

June 16, 2023. Modified the ellipse-ellipse find-intersection query code to be more robust when using floating-point arithmetic. The code still executes with rational arithmetic. Added a sample program for testing the code. The interface for FIQuery<T,Ellipse2<T>,Ellipse2<T> changed, so the area-of-intersection code had an obsolete interface removed. Added support for fused-multiply-add instructions.

GTE/BuildAll*.sln
GTE/Mathematics/Math.h
GTE/Mathematics/BSNumber.h
GTE/Mathematics/BSRational.h
GTE/Mathematics/IntrEllipse2Ellipse2.h
GTE/Mathematics/IntrAreaEllipse2Ellipse2.h
GTE/Samples/Intersection/CMakeLists.txt
GTE/Samples/Intersection/IntersectEllipses/*

June 11, 2023. Modified the code to initial cone parameters when the caller requests initialization. The previous initialization was an implementation of an algorithm describe in the LeastSquaresFitting.pdf file, but the mathematics of the file was incorrect. Fixing the mathematics leads to having to compute roots of a degree 10 polynomial. Instead, a simpler method is provided that uses a line-fitting algorithm to determine the relationship between height h and radial distance r. The PDF itself has been modified.

GTE/Mathematics/ApprCone3.h

May 25, 2023. Added a new class RootsQuadratic for robust estimation of roots to quadratic polynomials. The robust versions for cubic and quartic polynomials use this; the work is in progress. The older code in RootsPolynomial will remain in GTE for now but will be deprecated in GTL.

GTE/Mathematics/RootsQuadratic.h

Added number.Negate() member functions to avoid the verbose number.SetSign(-number.GetSign()). BSRational.SetSign had an mValue block with GTL_BINARY_SCIENTIFIC_SHOW_DOUBLE instead of GTE_BINARY_SCIENTIFIC_SHOW_DOUBLE. This affected only the visualizer display of mValue; the rational number itself is correct.

GTE/Mathematics/BSNumber.h
GTE/Mathematics/BSRational.h

May 22, 2023. The line-circle distance code had flawed logic for the case of a line not perpendicular to the plane of the circle and containing the origin. However, the block is entered when the line origin is on the normal line to the plane containing the circle center.

GTE/Mathematics/DistLine3Circle3.h

Two std::sqrt calls have arguments that are theoretically nonnegative, but floating-point rounding errors can lead to negative arguments. I added guards against this (clamp to nonnegative). The other std::sqrt calls already have the guard. The errant calls were in SolveDepressedQuartic. The problem was discovered during the line-circle distance query investigation.

GTE/Mathematics/RootsPolynomial.h

5 Updates to Version 6.5

May 6, 2023. Ethan Thornburg's modifications to allow disabling exceptions, calling std::terminate() instead.

GTE/Mathematics/Logger.h

Fedora 38 with gcc 13.1.1 compiler complained about std::move whose argument is a temporary object.

GTE/Mathematics/DisjointIntervals.h

Fedora 38 with gcc 13.1.1 compiler complained about potentially uninitialized variables. Add initialization even though in fact they are initialized before use (metaknowledge about the algorithm).

GTE/Mathematics/MinimalCycleBasis.h

Feature request for code to approximate continuous parametric curves with circular arcs. This includes a sample application. A PDF describing the algorithm is at the Geometric Tools website (ApproximateCurve-ByArcs.pdf).

GTE/BuildAll*.sln
GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}
GTE/Mathematics/ApprCurveByArcs.h
GTE/Samples/Mathematics/ApproximateBezierCurveByArcs/*
GTE/Samples/Mathematics/CMakeLists.txt

Rewrote the biquadratic implementation. The old code had a problem in one computation where subtractive cancellation caused many significant bits to be eliminated. This was the problem in the github issue about intersection of two ellipses, one rotated by $\pi/4 + \varepsilon$ for a small epsilon. An algebraic identity was used to avoid the subtractive cancellation, and the ellipse-ellipse intersection problem was eliminated. (The problem with ellipse-ellipse intersection for rotation ε still exists when you are using floating-point arithmetic. It does not occur with rational arithmetic, but I am rewriting that code to fix the floating-point problems).

GTE/Mathematics/RootsPolynomial.h

Factored out the area computations from IntrEllipse2Ellipse2.h for separation of concerns.

GTE/Mathematics/IntrEllipse2Ellipse2.h
GTE/Mathematics/IntrAreaEllipse2Ellipse2.h

The code for approximating and ellipse and points used the old OBB tree code. Modified it to use the new OBB tree code.

GTE/Mathematics/ApprCone3EllipseAndPoints.h

Added a recomputation of mValue (in debug builds) when using the functions frexp and ldexp specialized to BSRational. Without this, the value was incorrect.

GTE/Mathematics/BSRational.h

April 11, 2023. Redesigned the OBB tree of points, factoring out the tree construction to OBBTree.h. I also fixed an incorrect sizing of the nodes of the tree. A feature request was made of an OBB tree of triangles, which was easy to implement by derivation from OBBTree. The OBBTreeOfTriangles class has additional support for computing the intersection of linear component (line, ray, segment) with a triangle mesh. I also added code for computing an OBB of segments.

GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}
GTE/Mathematics/OBBTree.h
GTE/Mathematics/OBBTreeOfPoints.h
GTE/Mathematics/OBBTreeOfSegments.h
GTE/Mathematics/OBBTreeOfTriangles.h

March 27, 2023. Modified EnableTextures to bind UAVs in pixel shaders in a single call to OMSetRenderTargetsAndUnorderedAccessViews. I included a lengthy comment in EnableTextures about why the modification is necessary.

GTE/Graphics/DX11/DX11Engine.cpp

March 20, 2023. Fixed a regression when switching from <set> to <unordered_set>; the hash function and equality operator incorrectly included in these functions the ProcessedVertex location member. Also, when the input dataset has duplicated vertices, the InsertEdge function needs to replace the vertex indices by the equivalence-class representatives (via mDuplicates) before processing the edge. The sample application has another small test dataset, this one with a duplicated vertex.

GTE/Mathematics/Delaunay2.h
GTE/Mathematics/ConstrainedDelaunay2.h
GTE/Samples/Geometrics/Delaunay2D/Delaunay2DWindow.cpp

February 25, 2023. New file for sorting points in a circular manner about a center point. The website has a PDF describing the algorithm (SortPointsOnCircle.pdf).

GTE/Mathematics/SortPointsOnCircle.h
GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}

I have a template instantiation tool that was running debug (-g) builds. I added the ability to run release (-O2) builds. The Fedora gcc compiler warned about potentially uninitialized variables in the FastMarch* files. It also warned about an out-of-range index in the C++ std::sort code in the DistLine3Circle3 file for the line std::sort(candidates.begin(), candidates.begin() + numRoots). The numRoots is known not to exceed 4, but the compiler cannot determine this. I added the line numRoots = std::min(numRoots, static_cast<size_t>(4)) before the call to std::sort to avoid the warning.

GTE/Mathematics/FastMarch{2,3}.h
GTE/Mathematics/DistLine3Circle3.h

February 20, 2023. Eliminated the Fedora gcc12.{1,2}.1 release-build warnings about uninitialized members in std_function.h, probably due to incorrect optimization by the compiler. The warnings occurred in release builds of PVWUpdater.cpp where the member default constructors were called in the class constructors followed by a Set call to assign the members. I removed the Set calls and initialize the members directly in the constructor initializer list.

GTE/Graphics/PVWUpdater.cpp

February 2, 2023. Added the parent window handle to Window::Parameters for GTE on Microsoft Windows. That handle is passed to the CreateWindow functions.

GTE/Applications/MSW/Window.{h,cpp}
GTE/Applications/MSW/WindowSystem.cpp

Added the OnCopyData callback for GTE on Microsoft Windows. This allows passing data from one application to another application.

GTE/Applications/WindowApplication.{h,cpp}

Fixed linter complaint suggesting that .empty() be used instead of .size() == 0.

GTE/Graphics/CLODMeshCreator.h
GTE/Applications/MSW/WindowSystem.cpp

Added a missing pragma.

GTE/Mathematics/ContTetrahedron3.h

January 24, 2023. Added a new file for fitting a rectangle to a convex quadrilateral that is nearly a rectangle. The algorithm uses a least-squares formulation.

GTE/Mathematics/ApprConvexQuadByRect.h
GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}

January 17, 2023. Modified ETManifoldMesh.h to conform to commenting standards and choosing using instead of typedef. Added explicit initialization in the constructors of nested classes. Added braces to default constructor calls. The most important change is that during unit testing of the port of ETManifoldMesh from GTE to GTL, I created a small mesh for which GetBoundaryPolygons triggered an assertion (unexpected for a correct implementation of the algorithm). The idea of processing triangle strips at boundary vertices that are shared by 4 or more edge is correct. The code works for the strict definition of manifold mesh, where bow-tie configurations are not allowed. However, GTE has relaxed the constraint and allows bow-ties. For such meshes, the implementation of GetBoundaryPolygons is flawed and can lead to failure. Moreover, the approach I used in the implementation is too complicated. I re-implemented the algorithm to be simpler.

GTE/Mathematics/ETManifoldMesh.h

Modified VETManifoldMesh.h to conform to commenting standards and explicitly initializing nested class members in their constructors.

GTE/Mathematics/VETManifoldMesh.h

Modified TSManifoldMesh.h to conform to commenting standards and choosing using instead of typedef. Added explicit initialization in the constructors of nested classes. I renamed the T member to S in the nested class Triangle. The other mesh classes use T to stand for *Triangle*. Using T for *Tetrahedron* makes the code difficult to read at times. Switching to S (for *Simplex*) helps with readability. Finally, I added the keyword virtual to Insert and Remove to support a new class VTSManifoldMesh that adds vertex manipulation to the base class TSManifoldMesh.

GTE/Mathematics/TSManifoldMesh.h
GTE/Mathematics/Delaunay3.h
GTE/GTMathematics.{v16,v17}.{vcxproj,vcxproj.filters}
GTE/Mathematics/VTSManifoldMesh.h

The static 2-manifold class has a member function GetBoundaryPolygons, just as the dynamic class ETManifoldMesh does. This function has the same flawed implementation mentioned previously. It is now re-implemented using the same approach as in ETManifoldMesh.

GTE/Mathematics/StaticVETManifoldMesh2.h

Added a static 3-manifold class for tetrahedra meshes. This is significantly faster than the dynamic classes TSManifoldMesh and VTSManifoldMesh.

GTE/Mathematics/StaticVTSManifoldMesh3.h

Added newer static manifold mesh classes, MeshStaticManifold2 and MeshStaticManifold3, that perform better than StaticVETManifoldMesh2 and StaticVTSManifoldMesh3. The newer classes are also multithreaded. The older static classes will be deprecated for GTL in favor of the newer static classes.

GTE/Mathematics/MeshStaticManifold2.h
GTE/Mathematics/MeshStaticManifold3.h

Important Note. ETManifoldMesh, VETManifoldMesh and StaticVETManifoldMesh2 store adjacency pointers and indices according to the following design: Let the triangle have ordered vertices (V[0],V[1],V[2]). The triangle adjacent to edge (V[0],V[1]) is A[0]. The triangle adjacent to edge (V[1],V[2]) is A[1] and the triangle adjacent to edge (V[2],V[0]) is A[2]. The new class MeshStaticManifold2 uses a different design: The triangle adjacent to edge (V[1],V[2]) is A[0], the triangle adjacent to edge (V[2],V[0]) is A[1] and the triangle adjacent to edge (V[0],V[1]) is A[2]. That is, A[i] is the triangle adjacent to the edge opposite vertex V[i].

The design for the new 2-manifold triangle meshes is consistent with the design of the 3-manifold tetrahedra meshes TSManifoldMesh and VTSManifoldMesh. Let the tetrahedron have ordered vertices (V[0],V[1],V[2],V[3]). The triangle faces are counterclockwise ordered when viewed from outside the tetrahedron, T[0] = (V[1],V[2],V[3]), T[1] = (V[0],V[3],V[2]), T[2] = (V[0],V[1],V[3]) and T[3]] = (V[0],V[2],V[1]). Each face is opposite the vertex that is not listed in the face. The tetrahedron adjacent to face T[i] is A[i]. Unfortunately, I had not noticed previously that the primitive class Tetrahedron3 assigns the faces in the opposite order: T[0] = (V[0],V[2],V[1]), T[1] = (V[0],V[1],V[3]), T[2] = (V[0],V[3],V[2]) and T[3] = (V[1],V[2],V[3]).

The edge and face design in GTL will be consistent and uses the designs of MeshStaticManifold2 and MeshStaticManifold3.

The profiling results are listed next for the 2-manifold classes, running on an Intel i9-10900 CPU. The test triangle mesh has 1048576 vertices and 2097117 triangles.

class	numThreads	milliseconds
VETManifoldMesh	1	8471
StaticVETManifoldMesh2	1	1129
	2	904
	4	817
	8	761
	16	759
MeshStaticManifold2	1	816
	2	629
	4	526
	8	542
	16	551

The profiling results are listed next for the 3-manifold classes, running on an Intel i9-10900 CPU. The test tetrahedron mesh has 262144 vertices and 1765604 tetrahedra.

class	numThreads	milliseconds
VTSManifoldMesh	1	10706
StaticVTSManifoldMesh3	1	2035
	2	1512
	4	1256
	8	1160
	16	1092
MeshStaticManifold3	1	2202
	2	691
	4	507
	8	391
	16	362

January 11, 2023. The Error function and the comment in the file about the error formula were incorrect. They appear to have been copied-and-pasted from the ApprOrthogonalLine3.h implementation. The error code was introduced in the port from WM5 to GTE1. Thanks to Scott Johnson for informing me about the mistakes.

GTE/Mathematics/ApprOrthogonalPlane3.h

January 4, 2023. The posted file StaticVETManifoldMesh2.h was out of sync with the development version and was missing the initialization of the members of mAdjacents to *invalid* (-1). I also modified some comments and added some const modifiers where necessary.

GTE/Mathematics/StaticVETManifoldMesh2.h

January 1, 2023. New file that provides an additional speed-up to VETManifoldMeshNR for constructing static vertex-edge-triangle manifold meshes from collections of triangles.

GTE/GTMathematics.{v16,v17}.vcxproj*
GTE/Mathematics/StaticVETManifoldMesh2.h

December 24, 2022. Feature request for polyline/polygon offset. The code and a sample application has been added.

GTE/GTMathematics.*.*
GTE/Mathematics/PolylineOffset.h
GTE/BuildAll*.sln
GTE/Samples/Geometrics/CMakeLists.txt
GTE/Samples/Geometrics/PolylineOffset/*

6 Updates to Version 6.4

December 14, 2022. The following are modifications based on complaints from ClangCL for v143 MSVS tools regarding unused variables. The unused variables were actually used in the code (for watching during debugging), but the final values were never consumed by the functions.

GTE/Graphics/Shader.cpp (handle)
GTE/Graphics/GL45/GLSLShader.cpp (blockIndex)
GTE/Samples/Intersection/IntersectConvexPolyhedra/ConvexPolyhedron.h (numZero)
GTE/Samples/Graphics/MultipleRenderTargets/MultipleRenderTargetsWindow3.cpp (hr)
GTE/Samples/Physics/RoughPlaneParticle1/RoughPlaneParticle1Window2.h (mSize)
GTE/Samples/Physics/WrigglingSnake/WrigglingSnakeWindow3.cpp (t)

Added override to DrawScreenOverlay to resolve ClangCL complaint.

GTE/Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow2.h

Added override to DrawScreenOverlay to resolve ClangCL complaint.

GTE/Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow2.h

December 12, 2022. Added new code and sample application for 2D find-intersection queries between a linear component (line, ray, segment) and a segment mesh.

GTE/GTMathematics.*
GTE/Mathematics/IntrLine2SegmentMesh2.h
GTE/Mathematics/IntrRay2SegmentMesh2.h
GTE/Mathematics/IntrSegment2SegmentMesh2.h
GTE/BuildAll*.sln
GTE/Samples/Intersection/CMakeLists.txt
GTE/Samples/Intersection/IntersectRayPolygon/*

December 1, 2022. Added new 2D distance queries for point-circle and point-arc. Also fixed some missing header file references in the GTMathematics.v16 project.

GTE/Mathematics/DistPoint2Arc2.h
GTE/Mathematics/DistPoint2Circle2.h
GTE/GTMathematics.{v16,v17}.vcxproj
GTE/GTMathematics.{v16,v17}.vcxproj.filters

November 27, 2022. The CopyBackBuffer call mapped mBackBufferStaging but failed to unmap it after the copy.

GTE/Graphics/DX11/DX11Engine.cpp

November 11, 2022. Rolling back the changes from the 4 November 2002 update. The code is faster but the rounding errors cause misclassification of the number of points of intersection. Welcome to the trade-off of floating-point versus rational arithmetic. At some point I need to implement rational arithmetic classes using the GNU Multiprecision Library to take advantage of the better performance for arbitrary precision multiplication. At a later date, I can implement a similar speed-up for BSNumber and BSRational.

GTE/Mathematics/IntrEllipse2Ellipse2.h

November 4, 2022. When using BSRational<UIntegerAP32> for template type T, the number of words used in the SolveQuartic root finder can be extremely large—on the order of 100000. This leads to poor performance. The root finder has numerical rounding errors whether T is a floating-point type or a rational type, so to avoid the poor performance, the rational-valued quartic coefficients are converted to double and then reconverted to T. This introduces a small amount of rounding error, but performance is significantly better.

GTE/Mathematics/IntrEllipse2Ellipse2.h

November 2, 2022. Added a fuzzy containment that tests whether a point is on a circular arc. The non-fuzzy containment requires the point to be on the circle of the arc (or nearly on the circle within floating-point rounding errors).

GTE/Mathematics/Arc2.h

October 17, 2022. Back-ported QuadricSurface from GTL to GTE, both versions unit tested.

GTE/Mathematics/QuadricSurface.h

September 23, 2022. Fixed a bug in ComputeSqrDistance on line 247 (incorrect sign on s). Thanks to Tom Foster for reporting the bug.

GTE/Mathematics/IntrCanonicalBox3Cylinder3.h

August 25, 2022. Reimplemented the TCB spline code based on the rewrite of the PDF document *Kochanek–Bartels Cubic Splines (TCB Splines)*. I removed the tangent adjustments suggested by Kochanek and Bartels (based on discrete variables), replacing them with adjustments based on continuous variables. I added support for specifying the boundary tangents explicitly rather than duplicating the endpoints of the keyframes and computing the boundary tangents internally. Finally, I added a parameter (λ) per key frame position to control the speed at those positions and ensure continuity of speed.

GTE/Mathematics/TCBSplineCurve.h

August 23, 2022. Feature request for code to compute the minimum-area aligned box that contains a circular arc.

GTE/Mathematics/ContAlignedBox2Arc2.h

August 16, 2022. Reset some std::vector arrays for re-use of the operator() function. Added mConvexHull3 member and accessor (from Leopard20 at github). Modified the file version.

GTE/Mathematics/MinimumVolumeBox3.h

July 13, 2022. The managed library project compiles with the directory structure of GTE, but if folders are moved elsewhere, the managed library needed $(GTE_PATH) for the header search path. GTE_PATH is an environment variable that stores the path to the top-level folder GeometricTools/GTE (see the installation and release notes).

GTE/Samples/CSharpCppManaged/ManagedLibrary/ManagedLibrary.*,vcxproj

July 12, 2022. Removed the file ManagedObject.h, consolidating the contents with the managed files MinimumVolumeBox.{h,cpp}. Updated the copyright date in the assembly file. Modified the comments in the program file.

GTE/Samples/CSharpCppManaged/CSharpCppManaged.*.sln
GTE/Samples/CSharpCppManaged/ManagedLibrary/ManagedLibrary.*
GTE/Samples/CSharpCppManaged/ManagedLibrary/MinimumVolumeBox.{h,cpp}
GTE/Samples/CSharpCppManaged/ManagedLibrary/ManagedObject.h
GTE/Samples/CSharpCppManaged/CSharpApplication/Properties/AssemblyInfo.cs
GTE/Samples/CSharpCppManaged/CSharpApplication/Program.cs

July 4, 2022. The Window2 class was implemented not to allow window resizing. I removed this constraint and added comments in the header file about the responsibilities of an application class derived from Window2. The 2D fluid simulation sample application was modified to illustrate the resizing.

GTE/Applications/Window2.{h,cpp}
GTE/Samples/Physics/Fluids2D/Fluids2DMain.cpp
GTE/Samples/Physics/Fluids2D/Fluids2DWindow2.{h,cpp}

Modified comments that referred to the old file names GteSOMEFILE.h, replacing them by SOMEFILE.h.

GTE/Applications/Window2.cpp
GTE/Graphics/DX11/DX11InputLayout.h
GTE/Mathematics/ApprCone3.h
GTE/Mathematics/ApprTorus3.h
GTE/Mathematics/BSNumber.h
GTE/Mathematics/BSplineReduction.h
GTE/Mathematics/Cone.h
GTE/Mathematics/Math.h
GTE/Mathematics/MinimumAreaCircle2.h
GTE/Mathematics/MinimumVolumeSphere3.h
GTE/Mathematics/Quaternion.h
GTE/Mathematics/SlerpEstimate.h

June 29, 2022. The IsOnSurface function is a copy-and-paste error from the implicit curve file. I modified the name to IsOnCurve. I also set the output curvature on early exit in order not to leave output uninitialized.

GTE/Mathematics/ImplicitCurve2.h

Greatly simplified the implementation based on a revision of the document PrincipleCurvature.pdf.

GTE/Mathematics/ImplicitSurface3.h

June 26, 2022. Added a new class VETManifoldMeshNR which represents a vertex-edge-triangle manifold mesh for which triangles are provided as a single batch and NO REMOVE operations are going to be performed on the mesh. VETManifoldMeshNR significantly outperforms VETManifoldMesh. VETManifoldMesh is general purpose, allowing triangle insertions and triangle removals. The underlying C++ container classes lead to significant memory allocation/deallocation costs and are also expensive for find operations. It turns out that the design of VETManifoldMeshNR automatically gives you vertex adjacency information, so there is no ETManifoldMeshNR implementation. It is a requirement that the input triangles form a manifold mesh with consistently ordered triangles. In most applications, this requirement is already satisfied.

GTE/Mathematics/VETManifoldMeshNR.h

The VETManifoldMeshNR class is designed to minimize the allocations. For example, using this class for a collection of 1131652 positions and 2242293 triangles, the CPU times on an Intel (R) Core (TM) i9-10900 are as follows.

	ETManifoldMesh	VETManifoldMeshNR
graph creation	2234 milliseconds	118 milliseconds
connected component labeling	2169 milliseconds	40 milliseconds
get boundary polygons	232 milliseconds	25 milliseconds

Added comments to indicate that VETManifoldMeshNR might be a better choice for your application if it meets the preconditions.

GTE/Mathematics/ETManifoldMesh.h
GTE/Mathematics/VETManifoldMesh.h

7 Updates to Version 6.3

June 8, 2022. Added new files that contain an efficient implementation of natural cubic splines and natural quintic splines. The PDF documentation has been revised accordingly, `https://www.geometrictools.com/Documentation/NaturalSplines.pdf`. The class NaturalSplineCurve will be deprecated in favor of the new classes.

GTE/Mathematics/NaturalCubicSpline.h
GTE/Mathematics/NaturalQuinticSpline.h

Fixed gcc compiler warnings (signed/unsigned integer comparisons).

GTE/Mathematics/NaturalSplineCurve.h

Fixed gcc errors about the keyword template required before the dependent typenames involving the Degree* function calls. The errors occurred with gcc 12.1.1 on Fedora 36 but not with gcc 9.3.0 on Ubuntu or with Microsoft compilers.

GTE/Mathematics/ASinEstimate.h
GTE/Mathematics/ExpEstimate.h
GTE/Mathematics/LogEstimate.h

Fixed an error in the deprecated file for UniqueVerticesTriangles; the <array> header is not explicitly included. This occurred with gcc 12.1.1 on Fedora 36 even though my instantiation tool did not include explicit instantiation for that class. The error did not occur with gcc 9.3.0 on Ubuntu 20.04.1 LTS or Microsoft Visual Studio when running the tool.

GTE/Mathematics/UniqueVerticesTriangles.h

Modified code to avoid a potentially uninitialized variable warning from gcc 12.1.1 on Fedora 36.

GTE/Samples/Geometrics/IncrementalDelaunay2/IncrementalDelaunay2Window2.cpp
GTE/Samples/Intersection/IntersectTriangles2D/IntersectTriangles2DWindow2.cpp
GTE/Samples/Mathematics/Interpolation2DWindow3.cpp

Fedora 36 with gcc 12.1.1 complains about potential index-out-of-range because Evaluate references all possible elements, but the compiler is unaware that the logic of Evaluate is designed to assign only those elements that are valid. For now, I am passing in an array of 3-tuples knowing that only one will be filled in. Added arrays of N-tuples to avoid the warnings.

GTE/Mathematics/ParametricCurve.h
GTE/Mathematics/ParametricSurface.h
GTE/Samples/Mathematics/BSplineCurveReduction/BSplineCurveReductionWindow3.cpp

June 4, 2022. The constructor for free or closed splines allocated more than enough elements for mCoefficients and the GetNumPoints() function returned the number of points based on the size of mCoefficients. The constructor for clamped splines allocated exactly the correct number of elements for mCoefficients, but then GetNumPoints() returned the incorrect number of points. This caused too few points to be processed in clamped splines. This occurred when the optimization was added to minimize memory allocations and deallocations for free splines. The optimization code has been reworked and the same memory optimization was added for closed and clamped splines.

GTE/Mathematics/NaturalSplineCurve.h

May 30, 2022. The IsPowerOfTwo(int32_t) function called itself (infinite recursion). Modified the static cast to use uint32_t as intended.

GTE/Mathematics/BitHacks.h

May 16, 2022. In the GTL development, fixed a bug in the Inverse function for Transform<T> objects. The unit tests were incomplete in that they tested the inverse directly using queries to the internal matrices of Transform<T> rather than verifying the correctness of the output of Inverse. The corrections were backported to GTE.

GTE/Mathematics/Transform.h

The RectangleManager code was copied, pasted and modified to produce the BoxManager code. The comments were part of this and the BoxManager code referred to rectangles rather than boxes. Also, non-const references in both classes were modified to const references.

GTE/Mathematics/BoxManager.h
GTE/Mathematics/RectangleManager.h

8 Updates to Version 6.2

April 3, 2022. By user request, on Microsoft Windows I added virtual OnWindowsMessage to WindowApplication and called by WindowSystem::WindowProcedure. This allows the application to use a third-party UI package, processing windows messages intended for that package. A return value of true tells the window procedure not to pass the message to the application window.

GTE/Applications/WindowApplication.{h,cpp}
GTE/Applications/MSW/WindowSystem.cpp

April 2, 2022. Ported the Wild Magic 5 graphics sample ShadowMaps to GTE6. This is the last of the Wild Magic code I had planned on porting.

GTE/BuildAll*.*
GTE/Samples/Graphics/CMakeLists.txt
GTE/Samples/Graphics/ShadowMaps/*
GTE/Samples/Data/Stone.png

March 29, 2022. Ported the Wild Magic 5 MeshSmoother class to GTE6.

GTE/Mathematics/MeshSmoother.h

March 28, 2022. I finally trapped the problem with Windows OpenGL (WGL) where the first-drawn frame is not correct. I added a swap-buffers call immediately after creating the WGLEngine object (in the application layer), which eliminated the problem.

GTE/Applications/MSW/WindowSystem.cpp

Ported the Wild Magic 5 skinning sample to GTE6.

GTE/BuildAll*.*
GTE/Samples/Graphics/CMakeLists.txt
GTE/Samples/Graphics/Skinning/*

Added new constructors to avoid the repeated pattern of default construction of MeshFactory followed by the SetVertexFormat call.

GTE/Graphics/MeshFactory.{h,cpp}

Ported the line-torus find-intersection query from Wild Magic5 to GTE6 (at user request). Added a sample application to show that the code is working correctly. Internal unit tests have also been added.

GTE/BuildAll*.*
GTE/GTMathematics*.*
GTE/Mathematics/IntrLine3Torus3.h
GTE/Samples/Intersection/CMakeLists.txt
GTE/Samples/Intersection/IntersectLine3Torus3/*

March 25, 2022. The current 2D segment-segment queries use the find-intersection query for two lines, but the segments are converted to center-direction-extent form. The two-point representation has endpoints p_0 and p_1 and the parameterization is $p_0 + t(p_1 - p_0)$ for $t \in [0, 1]$. The center-direction-extent form is $c + sd$ for $|s| \leq e$. The center is $c = (p_0 + p_1)/2$, the direction is $d = (p_1 - p_0)/|p_1 - p_0|$ and the extent is $e = |p_1 - p_0|/2$. The computation of d involves a normalization which generally cannot be computed exactly with rational arithmetic. The computation of e is also not exact. The segment parameters returned from the queries are relative to s, not to t. The old queries are operator() functions and still exist. To allow for exact rational computation, I added new queries named Exact which uses the two-point form for segments. [In GTL development, I have been eliminating the use of the segment center-direction-extent form.]

GTE/Mathematics/IntrSegment2Segment2.h

The code had a call Normalize(diff) that is not necessary because the sign tests dependent on diff are (theoretically) independent of the length of diff. Removing the normalize call now allows the code to produce theoretically correct results when type T is a rational type.

GTE/Mathematics/IntrLine2Line2.h

Replace parenthesized casts by static_casts.

GTE/Mathematics/IntrLine2Line2.h
GTE/Mathematics/IntrLine2Ray2.h
GTE/Mathematics/IntrLine2Segment2.h
GTE/Mathematics/IntrRay2Ray2.h
GTE/Mathematics/IntrRay2Segment2.h
GTE/Mathematics/IntrSegment2Segment2.h

March 23, 2022. The SetRow function was missing a return statement.

GTE/Mathematics/GMatrix.h

March 22, 2022. Added abstract base classes for implicit curves in 2D, class ImplicitCurve2, and implicit surface in 3D, class ImplicitSurface3. The latter is a port of ImplicitSurface from Wild Magic 5. These classes support computation of coordinate frames and curvature information. For parametric curves and surfaces, see FrenetFrame and DarbouxFrame.

GTE/GTMathematics.*
GTE/Mathematics/ImplicitCurve2.h
GTE/Mathematics/ImplicitSurface3.h

Ported the Wild Magic 5 graphics sample VolumeFog to GTE6.

GTE/BuildAll*
GTE/GTGraphics.*.*
GTE/Graphics/GTGraphics.h
GTE/Graphics/VolumeFogEffect.{h,cpp}
GTE/Samples/Graphics/CMakeLists.txt
GTE/Samples/Graphics/VolumeFog/*
GTE/Data/BlueSky.png
GTE/Data/Shaders/VolumeFogEffect.{vs,ps}.{hlsl,glsl}

Fixed the minor number in file versions. Looks like versioning is the bane of the month.

GTE/Graphics/BoundTree.h
GTE/Graphics/CollisionGroup.h
GTE/Graphics/CollisionMesh.{h,cpp}
GTE/Graphics/CollisionRecord.h
GTE/Graphics/PlanarShadowEffect.{h,cpp}
GTE/Mathematics/ApprCone3EllipseAndPoints.h
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/FitConeByEllipseAndPoints*.{h,cpp}
GTE/Samples/Physics/CollisionsBoundTree/CollisionsBoundTree*.{h,cpp}
GTE/Samples/Physics/CollisionsMovingSpheres/CollisionsMovingSpheres*.{h,cpp}
GTE/Samples/Physics/CollisionsMovingSphereTriangle/CollisionsMovingSphereTriangle*.{h,cpp}

March 21, 2022. Ported the Wild Magic 5 sample CollisionsMovingSphereTriangle to GTE6.

GTE/BuildAll*
GTE/Samples/Physics/CMakeLists.txt
GTE/Samples/Physics/CollisionsMovingSphereTriangle/*

March 20, 2022. Ported the Wild Magic 5 sample CollisionsMovingSpheres to GTE6. The CollisionsBoundTree sample should have been in the Physics folder.

GTE/BuildAll*
GTE/Samples/SceneGraphs/CMakeLists.txt
GTE/Samples/Physics/CMakeLists.txt
GTE/Samples/Physics/CollisionsMovingSpheres/*
GTE/Samples/SceneGraphs/CollisionsBoundTree/*
GTE/Samples/Physics/CollisionsBoundTree/*

Ported the Wild Magic 5 collision detection code to GTE6: BoundTree, CollisionGroup and CollisionRecord. Added a wrapper class, CollisionMesh, that provides the interface to a Visual-based triangle mesh to be used at a template parameter for the collision detection code. Added test-intersection and find-intersection queries for moving triangles. Ported the WM5 sample CollisionsBoundTree to illustrate use of the collision detection.

GTE/GTGraphics.{v16,v16}.vcxproj
GTE/GTGraphics.{v16,v16}.vcxproj.filters
GTE/Graphics/GTGraphics.h
GTE/Graphics/CMakeLists.txt
GTE/Mathematics/IntrTriangle3Triangle3.h
GTE/Samples/CMakeLists.txt
GTE/Graphics/BoundTree.h
GTE/Graphics/CollisionGroup.h
GTE/Graphics/CollisionRecord.h
GTE/Graphics/CollisionMesh.{h,cpp}
GTE/Samples/SceneGraphs/CollisionsBoundTree/*

I forgot to update the minor version from 1 to 2 when I posted GTE6.2. Time to automate this step.

GTE/CMakeLists.txt
GTE/Samples/CMakeLists.txt
GTE/Samples/Distance/CMakeLists.txt
GTE/Samples/Geometrics/CMakeLists.txt
GTE/Samples/Graphics/CMakeLists.txt
GTE/Samples/Imagics/CMakeLists.txt
GTE/Samples/Intersection/CMakeLists.txt
GTE/Samples/Mathematics/CMakeLists.txt
GTE/Samples/Physics/CMakeLists.txt
GTE/Samples/SceneGraphs/CMakeLists.txt

March 18, 2022. Fixed a bug in ContainsPoint. Modified the internal unit tests to include the test that exposes the bug.

GTE/Mathematics/IntrTriangle3Triangle3.h

Removed the MSVS 2015 and MSVS 2017 versions of the C#/managed/C++ projects because I no longer support these compilers. The MSVS 2019 projects still exist. I added projects for MSVS 2022. Fixed the path problem in the MSVS 2019 version (modified environment variable from GTE4_PATH to GTE_PATH).

GTE/Samples/CSharpCppManaged/CSharpCppManaged.{v14,v15}.sln
GTE/Samples/CSharpCppManaged/CSharpApplication/CSharpApplication.{v14,v15}.csproj

GTE/Samples/CSharpCppManaged/CppLibrary/CppLibrary.{v14,v15}.*
GTE/Samples/CSharpCppManaged/ManagedLibrary/ManagedLibrary.{v14,v15}.*
GTE/Samples/CSharpCppManaged/CppLibrary/CppLibrary.v16.vcxproj
GTE/Samples/CSharpCppManaged/CSharpCppManaged.v16.sln
GTE/Samples/CSharpCppManaged/CSharpApplication/CSharpApplication.v16.csproj
GTE/Samples/CSharpCppManaged/CppLibrary/CppLibrary.v16.*
GTE/Samples/CSharpCppManaged/ManagedLibrary/ManagedLibrary.v16.*

March 13, 2022. The Math.h file generated compiler errors when I used one of the inline non-templated functions (needed <cstdint>). The UniqueVerticesTriangles.h file generated compiler errors when I tried to use it (needed <set>). This file is deprecated, but until it is removed it needs to compile. I updated my internal instantiation tool to handle these files appropriately.

GTE/Mathematics/Math.h
GTE/UniqueVerticesTriangles.h

9 Updates to Version 6.1

March 7, 2022. The planar reflection constructor had out-of-order initialization of members, and gcc on Linux complained about this. The Microsoft compilers (2019/2022) did not complain. I also removed a block of code in the constructor (unexposed in a conditional compilation block) that was moved to a class member function.

GTE/Graphics/PlanarReflectionEffect.cpp

The change to the interface for PlanarReflectionEffect required changes to the bouncing ball sample application.

GTE/Samples/Physics/BouncingBall/BouncingBallWindow3.{h,cpp}

The MinimumWidthPoints2 class is templated with class T. It contained a line using RotatingCalipers = typename RotatingCalipers<T> which Microsoft compilers allow. The gcc compilers complained about the typename, so I removed it. I removed two unused variables. There was a sign mismatch in a loop counter.

GTE/Mathematics/MinimumWidthPoints2.h

Removed unused lines of code (old iterator code that was replaced by range-based iteration) and fixeds a sign mismatch in a loop. Removed the declaration for static constexpr maxFloat = std::numeric_limits<float>::max(); and now use the max() value itself in the code. I must be missing something about declaring class-member constexpr values; the loader for gcc complained that maxFloat is undefined. The Microsoft compilers (2019/2022) never complain about such a declaration.

GTE/Graphics/CLODMeshCreator.h

An unreferenced variable for the camera position needed to be accessed in two lines of code rather than re-lookup the variable.

GTE/Samples/Physics/DLODNodes/DLODNodesWindow3.cpp

The gcc compiler complained about the compound Boolean expression in the GTE_ASSERT statement in SwitchNode::SetActiveChild. It wanted parentheses around the and expression.

GTE/Graphics/SwitchNode.cpp

The CMake list of source files needed to be updated for PlanarShadowEffect.cpp, SwitchNode.cpp, CLODMesh.cpp and DLODNode.cpp.

GTE/Graphics/CMakeLists.txt

March 6, 2022. I wrote a GTE6 sample application for planar shadows that is similar to the one for Wild Magic 5. I revised the planar reflections code so that the two effects have essentially the same design.

GTE/BuildAll*.sln
GTE/Graphics/PlanarShadowEffect.{h,cpp}
GTE/Graphics.h
GTE/Samples/Graphics/PlanarShadows/PlanarShadows*.vcxproj
GTE/Samples/Graphics/PlanarShadows/PlanarShadows*.vcxproj.filter
GTE/Samples/Graphics/PlanarShadows/PlanarShadowsMain.cpp
GTE/Samples/Graphics/PlanarShadows/PlanarShadowsWindow3.{h,cpp}
GTE/Samples/Graphics/PlanarShadows/PlanarShadows.code-workspace
GTE/Samples/Graphics/PlanarShadows/CMakeLists.txt
GTE/Samples/Graphics/PlanarShadows/CMakeSample.sh
GTE/Samples/Graphics/PlanarShadows/cmake-variants.json
GTE/Samples/Graphics/PlanarShadows/.vscode/launch.json
GTE/Samples/Graphics/PlanarShadows/.vscode/settings.json
GTE/Samples/Graphics/CMakeLists.txt
GTE/Graphics/PlanarReflectionEffect.{h,cpp}
GTE/Samples/Graphics/PlanarReflections/PlanarReflectionsMain.cpp
GTE/Samples/Graphics/PlanarReflections/PlanarReflectionsWindow3.{h,cpp}

March 4, 2022. I wrote a GTE6 sample application for planar reflections that is similar to the one for Wild Magic 5. The PlanarReflectionEffect drawing function had a bug. It needed to set the back-face stencil parameters in addition to the front-face stencil parameters. Without the back-face parameters, the reflection caster was drawn on both sides of the reflection plane.

GTE/BuildAll*.sln
GTE/Samples/Graphics/PlanarReflections/PlanarReflections*.vcxproj
GTE/Samples/Graphics/PlanarReflections/PlanarReflections*.vcxproj.filter
GTE/Samples/Graphics/PlanarReflections/PlanarReflections.cpp
GTE/Samples/Graphics/PlanarReflections/PlanarReflectionsWindow3.{h,cpp}
GTE/Samples/Graphics/PlanarReflections/PlanarReflections.code-workspace
GTE/Samples/Graphics/PlanarReflections/CMakeLists.txt
GTE/Samples/Graphics/PlanarReflections/CMakeSample.sh

GTE/Samples/Graphics/PlanarReflections/cmake-variants.json
GTE/Samples/Graphics/PlanarReflections/.vscode/launch.json
GTE/Samples/Graphics/PlanarReflections/.vscode/settings.json
GTE/Samples/Graphics/CMakeLists.txt
GTE/PlanarReflectionEffect.{h,cpp}

After looking at the requirements for the feature request, I modified the return value of MinimumWidthPoints2::operator() to be OrientedBox2<T> to be consistent with the minimum-area box code. The sample application now draws the minimum-area box and the minimum-width box for comparison.

GTE/Mathematics/MinimumWidthPoints2.h
GTE/Samples/Geometrics/MinimumAreaBox2DWindow2.{h,cpp}

March 3, 2022. Added an implementation of the rotating calipers algorithm, class RotatingCalipers. The goal was to refactor MinimumAreaBox2 to pull out the rotating calipers. However, this does not work because the minimum-area algorithm uses a pair of rotating calipers that work together. Added a new class MinimumWidthPoints2 that uses the rotating calipers algorithm to compute the width of a 2D point set. For now I have modified the MinimumAreaBox2D sample application to show also the results from computing the width of a point set. I also cleaned up the comments to use current formatting rules. And I added functions to allow passing std::vector inputs rather than raw pointers.

GTE/Mathematics/RotatingCalipers.h
GTE/Mathematics/MinimumWidthPoints2.h
GTE/Mathematics/MinimumAreaBox2.h
GTE/Mathematics/GTMathematics.*.vcxproj
GTE/Mathematics/GTMathematics.*.vcxproj.filters
GTE/Samples/Geometrics/MinimumAreaBox2DWindow2.{h,cpp}

The ConvexHull2 class was supposed to have switched from FPInterval to SWInterval, but I apparently overlooked this. The FPInterval class uses floating-point hardware operations to support interval arithmetic; however, the GCC compiles ignores the changes to the floating-point environment (it does not support modifying the rounding mode of the FPU). The SWInterval class implements interval arithmetic and rounds using software operations.

GTE/Mathematics/ConvexHull2.h

Two class members were not initialized by the constructor.

GTE/Graphics/DLODNode.cpp

February 25, 2022. Feature request to fit a cone to known elliptical cross sections and some additional points (ApprCone3EllipseAndPoints). If the cross sections are provided as point samples approximately on the ellipses, the new code also includes a class to extract points for each ellipse because the points might be stored in a file in some unknown order (ApprCone3ExtractEllipses). 3D ellipses are then fit to each subset of points in order to obtain the ellipse input to ApprCone3EllipseAndPoints. A PDF has been added to the documentation to describe the fitting algorithm, FitConeToEllipseAndPoints.pdf.

GTE/BuildAll*.sln
GTE/Mathematics/GTMathematics.*.vcxproj
GTE/Mathematics/GTMathematics.*.vcxproj.filters
GTE/Mathematics/ApprCone3EllipseAndPoints.h
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/FitConeByEllipseAndPoints*.vcxproj
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/FitConeByEllipseAndPoints*.vcxproj.filter
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/FitConeByEllipseAndPointsMain.cpp
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/FitConeByEllipseAndPointsWindow3.{h,cpp}
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/Data/CircleAndVertex.txt
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/Data/OneCircleOneEllipse.txt
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/Data/TwoEllipses.txt
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/Data/TwoPartialEllipses.txt
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/FitConeByEllipseAndPoints.code-workspace
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/CMakeLists.txt
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/CMakeSample.sh
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/cmake-variants.json
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/.vscode/launch.json
GTE/Samples/Mathematics/FitConeByEllipseAndPoints/.vscode/settings.json
GTE/Samples/Mathematics/CMakeLists.txt

February 24, 2022. Ported the WM5 classes ClodMesh, CreateClodMesh and CollapseRecord to GTE6 classes CLODMesh, CLODMeshCreator and CLODCollapseRecord. Ported the WM5 sample ClodMeshes to the GTE6 sample CLODMeshes.

GTE/BuildAll*.sln
GTE/Graphics/GTGraphics.h
GTE/Graphics/GTGraphics.*.vcxproj
GTE/Graphics/GTGraphics.*.vcxproj.filters
GTE/Graphics/CLODMeshCreator.h
GTE/Graphics/CLODMesh.{h,cpp}
GTE/Graphics/CLODCollapseRecord.h
GTE/Samples/SceneGraphs/CLODMeshes/CLODMeshes*.vcxproj
GTE/Samples/SceneGraphs/CLODMeshes/CLODMeshes*.vcxproj.filter
GTE/Samples/SceneGraphs/CLODMeshes/CLODMeshesMain.cpp
GTE/Samples/SceneGraphs/CLODMeshes/CLODMeshesWindow3.{h,cpp}
GTE/Samples/SceneGraphs/CLODMeshes/Data/FunctionX64Y64R8.png
GTE/Samples/SceneGraphs/CLODMeshes/CLODMeshes.code-workspace
GTE/Samples/SceneGraphs/CLODMeshes/CMakeLists.txt
GTE/Samples/SceneGraphs/CLODMeshes/CMakeSample.sh
GTE/Samples/SceneGraphs/CLODMeshes/cmake-variants.json
GTE/Samples/SceneGraphs/CLODMeshes/.vscode/launch.json
GTE/Samples/SceneGraphs/CLODMeshes/.vscode/settings.json
GTE/Samples/SceneGraphs/CMakeLists.txt

February 11, 2022. Modified the GenerateProject tool to generate the Visual Studio Code workspace files for sample applications.

GTE/Tools/GenerateProject/GenerateProject.*

GTE/Tools/GenerateProject/ProjectTemplateVSCode.{h,cpp}

Added missing *.code-workspace files to the recently ported physics samples.

GTE/Samples/Physics/BouncingSpheres/BouncingSpheres.code-workspace
GTE/Samples/Physics/BouncingTetrahedra/BouncingTetrahedra.code-workspace
GTE/Samples/Physics/RoughPlaneFlatBoard/RoughPlaneFlatBoard.code-workspace
GTE/Samples/Physics/RoughPlaneParticle1/RoughPlaneParticle1.code-workspace
GTE/Samples/Physics/RoughPlaneParticle2/RoughPlaneParticle2.code-workspace
GTE/Samples/Physics/RoughPlaneSolidBox/RoughPlaneSolidBox.code-workspace
GTE/Samples/Physics/RoughPlaneThinRod1/RoughPlaneThinRod1.code-workspace
GTE/Samples/Physics/RoughPlaneThinRod2/RoughPlaneThinRod2.code-workspace
GTE/Samples/Physics/WaterDropFormation/WaterDropFormation.code-workspace
GTE/Samples/Physics/WrigglingSnake/WrigglingSnake.code-workspace

Ported the SwitchNode class and SwitchNodes sample application from Wild Magic 5 to Geometric Tools Engine 6.

GTE/BuildAll*.sln
GTE/Graphics/GTGraphics.h
GTE/Graphics/GTGraphics.*.vcxproj
GTE/Graphics/GTGraphics.*.vcxproj.filters
GTE/Graphics/SwitchNode.{h,cpp}
GTE/Samples/SceneGraphs/SwitchNodes/SwitchNodes*.vcxproj
GTE/Samples/SceneGraphs/SwitchNodes/SwitchNodes*.vcxproj.filter
GTE/Samples/SceneGraphs/SwitchNodes/SwitchNodesMain.cpp
GTE/Samples/SceneGraphs/SwitchNodes/SwitchNodesWindow3.{h,cpp}
GTE/Samples/SceneGraphs/SwitchNodes/SwitchNodes.code-workspace
GTE/Samples/SceneGraphs/SwitchNodes/CMakeLists.txt
GTE/Samples/SceneGraphs/SwitchNodes/CMakeSample.sh
GTE/Samples/SceneGraphs/SwitchNodes/cmake-variants.json
GTE/Samples/SceneGraphs/SwitchNodes/.vscode/launch.json
GTE/Samples/SceneGraphs/SwitchNodes/.vscode/settings.json
GTE/Samples/SceneGraphs/CMakeLists.txt

Ported the DLODNode (dynamic level of detail node) class and DLODNodes sample application from Wild Magic 5 to Geometric Tools Engine 6.

GTE/BuildAll*.sln
GTE/Graphics/GTGraphics.h
GTE/Graphics/GTGraphics.*.vcxproj
GTE/Graphics/GTGraphics.*.vcxproj.filters
GTE/Graphics/DLODNode.{h,cpp}
GTE/Samples/SceneGraphs/DLODNodes/DLODNodes*.vcxproj
GTE/Samples/SceneGraphs/DLODNodes/DLODNodes*.vcxproj.filter
GTE/Samples/SceneGraphs/DLODNodes/DLODNodesMain.cpp

GTE/Samples/SceneGraphs/DLODNodes/DLODNodesWindow3.{h,cpp}
GTE/Samples/SceneGraphs/DLODNodes/DLODNodes.code-workspace
GTE/Samples/SceneGraphs/DLODNodes/CMakeLists.txt
GTE/Samples/SceneGraphs/DLODNodes/CMakeSample.sh
GTE/Samples/SceneGraphs/DLODNodes/cmake-variants.json
GTE/Samples/SceneGraphs/DLODNodes/.vscode/launch.json
GTE/Samples/SceneGraphs/DLODNodes/.vscode/settings.json
GTE/Samples/SceneGraphs/CMakeLists.txt

10 Updates to Version 6.0

February 7, 2022. Fixed a compiler error on Linux for a mismatch in size_t and int32_t. Changed the type of mNumCtrlPoints and mDegree to size_t.

GTE/Samples/Physics/WrigglingSnake/WrigglingSnakeWindow3.h

February 6, 2022. Ported the Wild Magic 5 physics sample BouncingTetrahedra to GTE. The port of BouncingSpheres was modified so that the two samples have the same conceptual framework.

GTE/Samples/Physics/BouncingTetrahedra/BouncingTetrahedra*.vcxproj
GTE/Samples/Physics/BouncingTetrahedra/BouncingTetrahedra*.vcxproj.filter
GTE/Samples/Physics/BouncingTetrahedra/BouncingTetrahedraMain.cpp
GTE/Samples/Physics/BouncingTetrahedra/BouncingTetrahedraWindow3.{h,cpp}
GTE/Samples/Physics/BouncingTetrahedra/PhysicsModule.{h,cpp}
GTE/Samples/Physics/BouncingTetrahedra/RigidPlane.{h,cpp}
GTE/Samples/Physics/BouncingTetrahedra/RigidTetrahedron.{h,cpp}
GTE/Samples/Physics/BouncingTetrahedra/Initial.txt
GTE/Samples/Physics/BouncingSpheres/BouncingSpheres*.vcxproj
GTE/Samples/Physics/BouncingSpheres/BouncingSpheres*.vcxproj.filter
GTE/Samples/Physics/BouncingSpheres/BouncingSpheresMain.cpp
GTE/Samples/Physics/BouncingSpheres/BouncingSpheresWindow3.{h,cpp}
GTE/Samples/Physics/BouncingSpheres/PhysicsModule.{h,cpp}
GTE/Samples/Physics/BouncingSpheres/BouncingSpheresMain.cpp
GTE/Samples/Physics/BouncingSpheres/BouncingSpheresWindow3.{h,cpp}
GTE/Samples/Physics/BouncingSpheres/RigidPlane.{h,cpp}
GTE/Samples/Physics/BouncingSpheres/RigidSphere.{h,cpp}
GTE/Samples/Physics/BouncingSpheres/Initial.txt

I factored out the impulse computations from the physics samples and moved them to RigidBody. The code now can be shared by the applications.

GTE/Mathematics/RigidBody.h

The Tetrahedron3 class has additional framework to support computing normals at features (vertices, edges, faces). The DistPoint3Tetrahedron3 query was modified because of an interface change in Tetrahedron3.

GTE/Mathematics/Tetrahedron3.h
GTE/Mathematics/DistPoint3Tetrahedron3.h

The algorithm I implemented for distance between tetrahedra was not robust because of testing for containment of vertices of one tetrahedron in another. I modified it to test centroids instead.

GTE/Mathematics/DistTetrahedron3Tetrahedron3.h

Added a new file for the test-intersection query between tetrahedra using the method of separating axes. This allowed for better performance in BouncingTetrahedra compared to using distance between tetrahedra.

GTE/Mathematics/IntrTetrahedron3Tetrahedron3.h

Fixed comments in the file.

GTE/Mathematics/IntrOrientedBox3OrientedBox3.h

February 3, 2022. Added missing projects from the MSVS 2019 build-all solutions. Added new build-all solutions for MSVS 2022.

GTE/BuildAll.v16.sln
GTE/BuildAllDX11.v16.sln
GTE/BuildAllGL45.v16.sln
GTE/BuildAll.v17.sln
GTE/BuildAllDX11.v17.sln
GTE/BuildAllGL45.v17.sln

February 1, 2022. Added new file that implements tetrahedron-tetrahedron query. This is needed for the port of BouncingTetrahedra from Wild Magic 5 to Geometric Tools Engine.

GTE/Mathematics/DistTetrahedron3Tetrahedron3.h

Modified the interfaces for computing barycentric coordinates. The raw arrays bary[] were replaced by std::array objects.

GTE/Mathematics/Vector2.h
GTE/Mathematics/Vector3.h
GTE/Mathematics/Delaunay2Mesh.h
GTE/Mathematics/Delaunay3Mesh.h
GTE/Mathematics/DistPoint3Tetrahedron3.h
GTE/Mathematics/IntpQuadraticNonuniform2.h
GTE/Mathematics/PlanarMesh.h
GTE/Samples/Mathematics/Interpolation2D/Interpolation2DWindow3.cpp

Added a function for computing the centroid of a tetrahedron.

GTE/Mathematics/Tetrahedron3.h

Updated a comment to be consistent with other distance-query comments.

GTE/Mathematics/DistTriangle3Triangles.h

Updated project files that reference the newly added mathematics header files.

GTE/Mathematics/GTMathematics.*.{vcxproj,vcxproj.filters}

January 31, 2022. Added new file that implements point-in-tetrahedron query. This is needed for the port of BouncingTetrahedra from Wild Magic 5 to Geometric Tools Engine.

GTE/Mathematics/ContTetrahedron3.h

January 30, 2022. The point-tetrahedron distance query code failed to compile because of a mismatch in index type. (Reorganization of my internal tools caused the template instantiation tool not to launch when Tetrahedron3 was modified.)

GTE/Mathematics/DistPoint3Tetrahedron3.h

January 26, 2022. I modified the sample application to allow the spheres to spin based on angular momentum, taking into account transfer of momentum during colliding contact. This required a different algorithm for impulsive function construction than what is in "Game Physics, 2nd edition" (Section 6.2.2). The algorithm details are in a new document at my website, ComputingImpulsiveForces.pdf. I also added friction to stop eventually the spheres from sliding and spinning when they are on the floor. Eliminated the texture seam by hacking it to a toroidal texture. Modified the RigidBody interface for simpler presentation of the Runge-Kutta solver steps and to make it clear the dependency between physics parameters and quantities derived from them (to support synchronizing the state).

GTE/Mathematics/RigidBody.h
GTE/Samples/Physics/BouncingSpheres/BouncingSpheresMain.cpp
GTE/Samples/Physics/BouncingSpheres/BouncingSpheresWindow3.{h,cpp}
GTE/Samples/Physics/BouncingSpheres/PhysicsModule.{h,cpp}
GTE/Samples/Physics/BouncingSpheres/Initial.txt
GTE/Samples/Data/BallTextureWrap.png

January 17, 2022. The function FitIndexed has tests for the components of mean being finite floating-point numbers. The test for mean[2] was missing.

GTE/ApprOrthogonalPlane3.h

Converted the GetFaceIndices function to static because the indices for the triangular faces are the same no matter which tetrahedron. Added also the static function GetAllFaceIndices to access the entire array of indices as a single object; this is useful for index buffers associated with tetrahedra.

GTE/Tetrahedron3.h

January 16, 2022. I replaced the hard-coded physics time steps by a physics clock that requires the simulation to run at 60 frames per second. I also added comments that the physics implementation is based on Section 6.6 of my Game Physics book (2nd edition).

GTE/Samples/Physics/BouncingSpheres/BouncingSpheresWindow3.{h,cpp}

January 14, 2022. I ported the Wild Magic 5 physics sample BouncingSpheres to GTE 5. The sample illustrates impulsed-based physics for a collection of spheres. I refactored the code so that the physics portion is isolated to the PhysicsModule.* files. Update the CMake list of projects for physics.

GTE/Samples/Physics/BouncingSpheres/BouncingSpheres*.*
GTE/Samples/Physics/BouncingSpheres/PhysicsModule.{h,cpp}
GTE/Samples/Physics/CMakeLists.txt

The function SetMass had type float instead of the template parameter Real.

GTE/Mathematics/RigidBody.h

January 12, 2022. I ported the Wild Magic 5 physics sample WaterDropFormation to GTE 6. Update the CMake list of projects for physics.

GTE/Samples/Physics/WaterDropFormation/WaterDropFormation*.*
GTE/Samples/Physics/WaterDropFormation/RevolutionSurface.{h,cpp}
GTE/Samples/Physics/CMakeLists.txt

Fixed the file versions to show 6.1 instead of 6.0 to be consistent with my versioning rules.

GTE/Samples/Physics/RoughPlaneParticle1/RoughPlaneParticle1*.*
GTE/Samples/Physics/RoughPlaneParticle1/PhysicsModule.{h,cpp}
GTE/Samples/Physics/RoughPlaneParticle2/RoughPlaneParticle2*.*
GTE/Samples/Physics/RoughPlaneParticle2/PhysicsModule.{h,cpp}
GTE/Samples/Physics/RoughPlaneThinRod1/RoughPlaneThinRod1*.*
GTE/Samples/Physics/RoughPlaneThinRod1/PhysicsModule.{h,cpp}
GTE/Samples/Physics/RoughPlaneThinRod2/RoughPlaneThinRod2*.*
GTE/Samples/Physics/RoughPlaneThinRod2/PhysicsModule.{h,cpp}
GTE/Samples/Physics/RoughPlaneFlatBoard/RoughPlaneFlatBoard*.*
GTE/Samples/Physics/RoughPlaneFlatBoard/PhysicsModule.{h,cpp}
GTE/Samples/Physics/RoughPlaneSolidBox/RoughPlaneSolidBox*.*
GTE/Samples/Physics/RoughPlaneSolidBox/PhysicsModule.{h,cpp}
GTE/Samples/Physics/WrigglingSnake/WrigglingSnake*.*
GTE/Samples/Physics/WrigglingSnake/TubeSurface.{h,cpp}

Added hyperlinks to the website PDF that describes some simple friction-based algorithms.

GTE/Samples/Physics/RoughPlaneParticle1/RoughPlaneParticle1.h
GTE/Samples/Physics/RoughPlaneParticle2/RoughPlaneParticle2.h
GTE/Samples/Physics/RoughPlaneThinRod1/RoughPlaneThinRod1.h
GTE/Samples/Physics/RoughPlaneThinRod2/RoughPlaneThinRod2.h
GTE/Samples/Physics/RoughPlaneFlatBoard/RoughPlaneFlatBoard.h
GTE/Samples/Physics/RoughPlaneSolidBox/RoughPlaneSolidBox.h

January 11, 2022. I ported the Wild Magic 5 physics sample WrigglingSnake to GTE 6. Updated the CMake list of projects for physics.

GTE/Samples/Physics/WrigglingSnake/WrigglingSnake*.*
GTE/Samples/Physics/WrigglingSnake/TubeSurface.{h,cpp}
GTE/Data/Snake.png
GTE/Samples/Physics/CMakeLists.txt

January 10, 2022. I ported the Wild Magic 5 physics samples to GTE 6, the ones involving friction.

GTE/Samples/Physics/RoughPlaneParticle1/RoughPlaneParticle1*.*
GTE/Samples/Physics/RoughPlaneParticle1/PhysicsModule.{h,cpp}
GTE/Samples/Physics/RoughPlaneParticle2/RoughPlaneParticle2*.*
GTE/Samples/Physics/RoughPlaneParticle2/PhysicsModule.{h,cpp}
GTE/Samples/Physics/RoughPlaneThinRod1/RoughPlaneThinRod1*.*
GTE/Samples/Physics/RoughPlaneThinRod1/PhysicsModule.{h,cpp}
GTE/Samples/Physics/RoughPlaneThinRod2/RoughPlaneThinRod2*.*
GTE/Samples/Physics/RoughPlaneThinRod2/PhysicsModule.{h,cpp}
GTE/Samples/Physics/RoughPlaneFlatBoard/RoughPlaneFlatBoard*.*
GTE/Samples/Physics/RoughPlaneFlatBoard/PhysicsModule.{h,cpp}
GTE/Samples/Physics/RoughPlaneSolidBox/RoughPlaneSolidBox*.*
GTE/Samples/Physics/RoughPlaneSolidBox/PhysicsModule.{h,cpp}

I removed the post-build copies of the executables to the folder GeometricTools/GTE/Executable. The tools are used infrequently enough that I now run them from their _Output folders. Naturally, you can manually copy the executables to where you want them.

GTE/Tools/BitmapFontCreator/BitmapFontCreator.*.vcxproj
GTE/Tools/GenerateProject/GenerateProject.*.vcxproj

January 9, 2022. I added projects and solutions for Microsoft Visual Studio 2022. I removed the Microsoft Visual Studio 2015 projects and solutions because that version of the IDE reached mainstream end date on October 13, 2020. Microsoft Visual Studio 2017 reaches its mainstream end date on April 12, 2022. However, I have also removed the Microsoft Visual Studio 2017 projects and solutions because I do not have enough time to maintain so many versions of the compiler. From now on, I will support two versions of MSVS, the current one (whatever that is) and the previous one.

The Tools folder has various projects that were expanded to include projects and solutions for MSVS 2022. The GenerateProject tool has been modified to generate only MSVS 2019 and MSVS 2022 projects and solutions.

GTE/Tools/BitmapFontCreator/BitmapFontCreator.v17.*
GTE/Tools/BitmapFontCreator/BitmapFontCreator.v16.vcxproj
GTE/Tools/ChangePlatformToolset/ChangePlatformToolset.v17.*
GTE/Tools/ChangePlatformToolset/ChangePlatformToolset.v16.vcxproj
GTE/Tools/FiniteDifferences/FiniteDifferencesDX11.v17.*
GTE/Tools/FiniteDifferences/FiniteDifferencesGL45.v17.*
GTE/Tools/FiniteDifferences/FiniteDifferencesDX11.v16.vcxproj
GTE/Tools/FiniteDifferences/FiniteDifferencesGL45.v16.vcxproj
GTE/Tools/GenerateApproximations/GenerateApproximations.v17.*
GTE/Tools/GenerateApproximations/GenerateApproximations.v16.vcxproj
GTE/Tools/GenerateOpenGLWrapper/GenerateOpenGLWrapper.v17.*
GTE/Tools/GenerateOpenGLWrapper/GenerateOpenGLWrapper.v16.vcxproj
GTE/Tools/GenerateOpenGLWrapper/GenerateOpenGLWrapper.cpp
GTE/Tools/PrecisionCalculator/PrecisionCalculator.v17.*
GTE/Tools/PrecisionCalculator/PrecisionCalculator.v16.vcxproj
GTE/Tools/RotationApproximation/RotationApproximationDX11.v17.*
GTE/Tools/RotationApproximation/RotationApproximationGL45.v17.*
GTE/Tools/RotationApproximation/RotationApproximationDX11.v16.vcxproj
GTE/Tools/RotationApproximation/RotationApproximationGL45.v16.vcxproj

The GenerateProject tool now creates project, solutions, filter files and source files only for MSVS 2019 and MSVS 2022.

GTE/Tools/GenerateProject/GenerateProject.v17.*
GTE/Tools/GenerateProject/GenerateProject.v16.{vcxproj,vcxproj.filters}
GTE/Tools/GenerateProject/GenerateProject.cpp
GTE/Tools/GenerateProject/ProjectTemplate.{h,cpp}
GTE/Tools/GenerateProject/ProjectTemplate.v17.{h,cpp}
GTE/Tools/GenerateProject/ProjectTemplate.v16.{h,cpp}
GTE/Tools/GenerateProject/ProjectTemplate.v15.{h,cpp}
GTE/Tools/GenerateProject/ProjectTemplate.v14.{h,cpp}

January 3, 2022. Version 6.0 is posted. The major revision is based on running the code analysis tools for Microsoft Visual Studio 2019 16.11.8 and for ClangCL. The reported issues were addressed with the exception of some incorrect warnings from MSVS 2019. For now, the incorrect warnings are encapsulated by #pragma commands and will be removed in the GTL development track.

The MSVS code analysis tool is not robust. Sometimes warnings occur in both the Output and Error List windows. Sometimes they occur in only one or the other window (but not both). And sometimes they do not occur in either window, but the 3-dot markers show up in source files that are opened after which the warnings show up in the Error List window. As I discover source files where the warnings show up only when the source file is opened, I will fix the issues. I believe most of these will be warnings about potentially uninitialized variables, typically for class objects whose default constructors should be called but MSVS seems to believe the class members are not initialized. This is the case for the Vector classes, but in the code using these classes, the members are set soon after the declaration. I was hoping the tool would figure out that the member initialization was deferred. (If I were to allocate a std::vector of Vector with a very large number of elements, and the code fills them in by loading data from a file, it would be a shame to waste all that time having the constructor initialize the members only to fill them in again from the loaded data.)

Another tool issue occurred with SymmetricEigensolver.h, in the GetEigenvector function. I have comments in that file about why the tool report is incorrect. For a long time I have been able to ignore the warning because code using it compiles fine, even with *treat warnings as errors*. The warnings occurred as part of a regular source-code build. For the first time I ran the code analyzer explicitly from the MSVS IDE menu on a source file that contains only the include of the aforementioned header file. The report now includes more warnings that appear to be based on the same incorrect diagnosis. The original warning was about a potential out-of-range index, and the Error List window allows you to drop-down a list of steps and assumptions to support the diagnosis. The new warning is about the same issue but for some reason displays a list of line numbers that show up in the drop-down list. After these new warnings occurred, I can no longer successfully build code using the eigensolver. I had to add #pragma commands to prevent the warnings. After that, a couple of other files generated similar warnings that had to be disabled using #pragma commands.

I had also spent a lot of time eliminating the warning about preferring scoped enumerations over unscoped ones. The unscoped ones typically defined enumerants that were used as array indices (in the DX11 and GL45 engine code). This is not allowed with scoped enumerations. I replaced the unscoped enumerations with nested struct, each structure containing constant expressions of the form static uint32_t constexpr someName = someValue;. MSVS 2019 and ClangCL provided with MSVS 2019 allowed this modification, but unfortunately gcc on my Linux boxes did not. Searching stackoverflow, it appears that C++ considers such structs to be *incomplete*. I actually got linker errors about the various constants referenced by the sample applications but not found in the libraries linked to the applications. I restored the unscoped enumerations and disabled the code analysis warning number in the project settings.

A couple of the sample applications use BSRational<UIntegerFP<N>> where N is large. These lead to code analysis warnings about /analyze:stacksize numKBs indicating that numKBs is larger than the maximum stack size and the you should consider moving data from the stack to the heap. The samples have run correctly without stack overflow errors, so it is not clear to me what the problem is. Regardless, I modified the project settings and specified numKBs large enough to avoid the warnings. When I switched to using ClangCL, the compiler complains it cannot find files and lists the name /analyze:stacksize. It appears that ClangCL does not understand this analysis tool option. Unfortunately, with *treat warnings as errors*, those sample applications will not compile. So I removed the setting of numKBs and then disabled the code analysis warning number in the project settings.

11 Updates to Version 5.15

January 3, 2022. To remove Microsoft Visual Studio 2019 code analysis warnings, an extremely large number of files were modified. I am posting these as GTE 6.0. I suspect that even more changes are necessary because the code analysis tool is not robust. Sometimes warnings show up in the Output and Error List windows. Sometimes they show up only in the Output window or only in the Error List window. And sometimes they do not show up in either; I have to open a file with an issue (typically an "auto cannot deduce references" warning) and see the three-dot notification below the offending variable. I also had to suppress some warnings with pragmas because my analysis shows the code is correct, but I was unable to modify the code to eliminate the warnings.

The graphics engine is the main system that had many changes. The typical issues are listed next and are some what annoying to deal with.

1. Prefer scoped enumerations over unscoped ones. I used unscoped enumerations as indices into arrays for DX11 and OpenGL 4.5. If I modified these to scoped enumerations, you cannot use them for indexing. I choose to replace such enumerations by structs with constexpr values for the indices. The trade-off is that you loose the strict typing the enumerations give you when passing variables to functions.

2. The auto cannot deduce references shows up a lot. I had auto vbuffer = someObject→GetVertexBuffer(). The return value is a shared ptr, so I do not care that a copy is made (with an increment on the reference count), even though the code analysis warning says I should consider otherwise. I modified these anyway to auto const& vbuffer = someObject→GetVertexBuffer().

3. The final annoying issue is the use of int indices for STL container classes that use size_t indices. This is not a problem until you have index arithmetic enclosed with the []. The complaint is that the 4-byte arithmetic can lead to overflow before typecasting to an 8-byte value. This has plagued me for years, because I had chosen to use int as the natural index into arrays. I added a lot of ugly static casts. As I am porting the code to GTL, I am fixing this problem. In GTE you will have to live with the ugly casts for now.

4. I replaced the unsigned T types by those in cstdint.

Some code analysis issues I fixed were relevant in that there were potential bugs.

December 29, 2021. The following modifications are based on warnings from 'LLVM - clang-cl' provided with Microsoft Visual Studio 2019.

Removed the unused variable mTextColor.

GTE/Samples/Intersection/AllPairsTriangles/AllPairsTrianglesWindow3.h

The BouncingBall sample creates a PlanarReflectionEffect via a std::unique_ptr. To do so requires PlanarReflectionEffect to implement a virtual destructor because it has a virtual member function (Draw).

GTE/Graphics/PlanarReflectionEffect.h

The BSplineInterpolation sample creates a IntpBSplineUniform<double,Controls,1> object whose first parameter is the degree of the fit. That parameter had braces which are not needed.

GTE/Samples/Imagics/BSplineInterpolation/BSplineInterpolation.cpp

Removed the unused variable mCurrentIndex.

GTE/Samples/Geometrics/Delaunay2D/Delaunay2DWindow.{h,cpp}
GTE/Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunayWindow2D.{h,cpp}

Removed the unused variable mLightWorldDirection.

GTE/Samples/Physics/FoucaultPendulum/FoucaultPendulumWindow3.{h,cpp}

Removed unused variable mNumSlices.

GTE/Samples/Physics/MassSprings3D/GpuMassSpringVolume.{h,cpp}

The mTIQuery is used only when the preprocessor symbol USE_TIQUERY_OVERRIDE is defined. Moved the define to the header file and defined mTIQuery only when the preprocessor symbol is defined.

GTE/Samples/Intersection/IntersectTriangleBox/IntersectTriangleBoxWindow3.{h,cpp}

Added override to functions OnMouseClick, OnMouseMotion and OnCharPress.

GTE/Samples/Intersection/MovingCircleRectangle/MovingCircleRectangleWindows.h

Removed unused this as a capture parameter in DrawPath.

GTE/Samples/Mathematics/ShortestPath/ShortestPathWindow2.cpp

Removed the unused variable mLightWorldDirection.

GTE/Samples/Physics/SimplePendulumFriction/SimplePendulumFrictionWindow3.{h,cpp}

Removed the unused variable mTextColor.

GTE/Samples/Geometrics/VertexCollapseMesh/VertexCollapseMeshWindow3.{h,cpp}

December 28, 2021. Improved the performance of Delaunay3 by using the newly modified TSManifoldMesh and by including interval arithmetic with the rational arithmetic in ToPlane and ToCircumsphere to amortize the cost of the computations. In many cases, interval arithmetic can determine the signs of determinants using only floating-point arithmetic which avoids the more expensive rational-only arithmetic. For a user-supplied dataset of over 3000 points and executing on an Intel i9-10900 processor, the following measurements were made for float with compute type BSNumber< UIntegerFP32<44>> and for double with compute type BSNumber< UIntegerFP32<330>>. The old code Delaunay3<InputType,ComputeType> required 3611 milliseconds for float input and 3858 milliseconds for double. The new code Delaunay3<T> required 1274 milliseconds for float and 1280 milliseconds for double. The old code is deprecated, to be removed in GTL, but the current file still contains it in addition to the new code.

GTE/Mathematics/Delaunay3.h
GTE/Samples/Geometrics/Delaunay3DWindow3.{h,cpp}

Improved the performance of TSManifoldMesh, replacing std::map by std::unordered_map, replacing std::shared_ptr by std::unique_ptr and replacing std::weak_ptr by raw pointers.

GTE/Mathematics/TSManifoldMesh.h

Updated some comments. The maximum values of N listed for UIntegerFP32 with double input were incorrect (much too large).

GTE/Mathematics/Delaunay2.h

12 Updates to Version 5.14

December 21, 2021. Removed the LCP solver approach for the test-intersection query between a box and a finite cylinder. The approach turned out to work correctly when using exact arithmetic but in some configurations had significant rounding errors when using floating-point arithmetic. The code has been modified using the third algorithm described in Intersection of a Box and a Finite Cylinder.

GTE/Mathematics/IntrCanonicalBox3Cylinder3.h
GTE/Mathematics/IntrAlignedBox3Cylinder3.h
GTE/Mathematics/IntrOrientedBox3Cylinder3.h
GTE/Mathematics/IntrPlane3Cylinder3.h

December 20, 2021. Removed unused variables (warnings by clang). The OPENGL_VERSION_1_1 constant is required on Linux, but it is unused on Windows. I moved that constant to a conditional block for Linux.

GTE/Applications/GLX/WindowSystem.{h,cpp}
GTE/Graphics/MorphController.cpp
GTE/Graphics/GL45/GL45.cpp
GTE/Graphics/GL45/GL45InputLayout.{h,cpp}

Added required override modifier to virtual functions (warnings by clang).

GTE/Applications/GLX/Window.h
GTE/Graphics/GraphicsEngine.h
GTE/Graphics/DX11/DX11Engine.h
GTE/Graphics/GL45/GL45Engine.h

The CMake option commands had incorrect commas in 140 of the CMakeLists.txt files.

GTE/Applications/CMakeLists.txt
GTE/Graphics/CMakeLists.txt
GTE/MathematicsGPU/CMakeLists.txt
GTE/Samples/*/CMakeLists.txt

Clang complained about using (char*)0 + offset. I replaced this with reinterpret_cast¡char*¿(offset).

GTE/Graphics/GL45/GL45Engine.cpp

Clang complained that the programHandle to the GL45InputLayout constructor is unused. I deleted the name as per the standards. I will remove the parameter altogether once I rewrite the graphics code.

GTE/Graphics/GL45/GL45InputLayout.cpp

Clang suggested using braces to initialize subobjects in the ConfiguraionTable at the end of the file.

GTE/Mathematics/MarchingCubes.h

Clang complained that the YFunction lambda has a this parameter but does not use it. Removed that parameter.

GTE/Mathematics/RootsBisection2.h

Clang complained about various Microsoft header structures. Added zeroing of the PROPBAG2 option. Added parentheses to the declaration of PROPBAG2 option and VARIANT varValue. Clang complains that varValue does not have members vt, boolVal or bVal. Microsoft headers indicate they do exist (and the WICFileIO code has worked for years). The code probably needs to use the CCom* class to encapsulate VARIANT and use that. More importantly, the Windows SDK header files need a serious upgrade.

GTE/Applications/MSW/WICFileIONative.cpp

Removed include of Logger.h because the header file does not use it.

GTE/Mathematics/Cylinder3.h

Clarified a comment about vertex extraction.

GTE/Mathematics/AlignedBox.h

December 1, 2021. The source code failed when the plane normal and cylinder axis direction were parallel. The mathematical derivation in the document IntersectionPlaneCylinder.pdf was incorrect because it overlooked the singularity when computing the major axis of the ellipse of intersection. The PDF was rewritten and describes a different major axis construction. The new source code implements this. Moreover, it also now supports both finite and infinite cylinders (test-intersection and find-intersection queries).

GTE/Mathematics/IntrPlane3Cylinder3.h

Added a comment that you cannot specify plane.origin and plane.constant independently (with an example of why not).

GTE/Mathematics/Hyperplane.h

November 28, 2021. Added implementations of two algorithms in *Geometric Tools for Computer Graphics* (Sections 8.6 and 8.7) at user request. The implementation for the algorithm of Section 8.6 differs from that of the book–a simpler approach is used, described in the file comments. The book has pseudocode for the algorithm in Section 8.7, but there are numerous errors. I have added the corrections to the book correction page at the website. The approach I used is described in the file comments.

GTE/GTMathematics.*.{vcxproj,vcxproj.filters}
GTE/Mathematics/CircleThroughTwoPointsSpecifiedRadius.h (Section 8.6)
GTE/Mathematics/CircleThroughPointSpecifiedTangentAndRadius.h (Section 8.7)

MSVS 2015 complained that std::greater was undefined. I added the include of header <functional> to access the definition. MSVS 2015 also complained about unchecked iterators when using std::copy. I replaced the std::copy calls by C++ loops.

GTE/Mathematics/SingularValueDecomposition.h

I will be quite happy when MSVS 2015 reaches end-of-life.

13 Updates to Version 5.13

November 19, 2021. The bookkeeping for decomposing the bidiagonal matrix into a block matrix with bidiagonal blocks was not working correctly when a superdiagonal entry was zero. I decided to revisit every piece of the SVD code and updated the unit tests to ensure the code is working properly. The output of the SVD now produces only a singular-value matrix with diagonal elements occurring in descending order. The interfaces were modified slightly.

GTE/Mathematics/SingularValueDecomposition.h

The November 2, 2021 entry fixed a bug in the non-deprecated version of FillRegion. I also added that fix to the deprecated version in case folks are still using it.

GTE/Mathematics/TriangulateCDT.h

The revision of Logger.h removed the implicit include of <limits> that ETManifoldMesh.h requires.

GTE/Mathematics/ETManifoldMesh.h

Added definitions of zero and one but forgot to use them in the code.

GTE/Mathematics/IntrLine2Segment2.h

I removed static int32_t constexpr INTERIOR_VERTEX = -1 from the header file. On Fedora 34 with gcc 11.1.1, the linker complained that this symbol is undefined, which is incorrect given it is a constexpr. Without the change, gcc 9.3.0 on Ubuntu 20.04.1 LTS does not generate the linker error.

GTE/Mathematics/GenerateMeshUV.h

November 11, 2021. I removed the Logger class to avoid linker errors about duplicate symbols when the class is used in separate modules that are linked to an application. The issue was reported by someone using Unreal Engine 5 (Plugins/Experimental/GeometryProcessing in the GeometryAlgorithms module) and trying to link in the mathematics code from GTE. The problem is that the LogAssert and LogError macros contain Logger constructor calls, but the compiler does not always inline the calls. The new code in Logger.h contains only macros. The logging system no longer supports LogInformation or LogWarning messages. The system does throw C++ exceptions. If you disable C++ exceptions and undefine the macros, be aware that not throwing exceptions causes program flow to continue in an unknown—and most likely—unsafe state. I have retained the name of Logger.h even though no logging occurs; the file name in GTL will be Exceptions.h. I removed all the LogReporter code in the sample applications, replacing them by try-catch blocks so that an unhandled exception leads to its associated message displayed in the application command window. The modified sample application files (*Main.cpp) are numerous and not listed here.

GTE/Mathematics/Logger.h
GTE/Mathematics/LogToFile.h
GTE/Mathematics/LogToStdOut.h
GTE/Mathematics/LogToStringArray.h
GTE/Applications/GTApplications.h
GTE/Applications/Common/LogReporter.{h,cpp}
GTE/Applications/MSW/LogToMessage.{h,cpp}
GTE/Applications/MSW/LogToOutputWindow.{h,cpp}

Various library files are affected by the Logger redesign, mainly eliminating LogWarning and LogInformation. The RootBisection2 class needed a member mNoGuaranteeForRootBound and an accessor for the caller to determine information that used to be reported by the logging system.

GTE/Mathematics/BoundingSphere.h
GTE/Mathematics/ConformalMapGenus0.h
GTE/Mathematics/RootsBisection1.h
GTE/Mathematics/RootsBisection2.h
GTE/Mathematics/UIntegerALU32.h
GTE/Applications/Environment.{h,cpp}
GTE/Graphics/DX11/DX11Buffer.cpp
GTE/Graphics/DX11/DX11InputLayoutManager.cpp
GTE/Graphics/GL45/GL45Buffer.cpp
GTE/Graphics/GL45/GL45Engine.cpp
GTE/Graphics/GL45/GL45InputLayoutManager.cpp
GTE/Graphics/GL45/GL45SamplerState.cpp
GTE/Graphics/GL45/GL45TextureArray.cpp
GTE/Graphics/Resource.cpp

The Logger.h file had an include of <set> that was used indirectly by various files. Those files need the explicit include. The Delaunay2.h file had an include of FPInterval.h that is not needed.

GTE/Mathematics/AdaptiveSkeletonClimbing3.h
GTE/Mathematics/Delaunay2.h
GTE/Mathematics/ETNonmanifoldMesh.h
GTE/Mathematics/GenerateMeshUV.h
GTE/Mathematics/PlanarMesh.h
GTE/Mathematics/UniqueVerticesSimplices.h

Eliminated dependency on GTE. Modified the preprocessor defines of constants to UINT constexpr.

GTE/Graphics/DX11/DX11.{h,cpp}

Modified hard-coded exceptions to include file name, function name and line number. Modified the preprocessor defines of constants to `int constexpr`.

GTE/Graphics/GL45/GL45.{h,cpp}

The Microsoft Visual Studio 2019 code analysis tool generates a large number of warnings. Even though I have been eliminating them in the GTL development track, I want to eliminate them from the GTE track because the 2 tracks will exist simultaneously for a while.

I had designed Wild Magic and Geometric Tools Engine to use `int` indices into various classes, most notably `Vector` and `Matrix`. The use of `size_t` indices in the C++ Standard Library causes a lot of problems. Specifically for 64-bit builds where `size_t` is an 64-bit unsigned integer, any 32-bit integer arithmetic used to compute indices generates warnings. Replacing all these is a giant pain in the arse. I do not know the rationale for C++ standards to use `size_t`, but in my naive view I would have preferred a signed type. Loops that decrement the loop counter are problematic because a termination condition of $i \geq 0$ is not sufficient. It is what it is, so I am eliminating the warnings by static casts in GTE but using `size_t` in GTL.

GTE/Mathematics/GMatrix.h
GTE/Mathematics/Polynomial1.h
GTE/Mathematics/ApprEllipseByArcs.h
GTE/Mathematics/ApprPolynomial2.h
GTE/Mathematics/ApprPolynomial3.h
GTE/Mathematics/ApprPolynomial4.h
GTE/Mathematics/ApprPolynomialSpecial2.h (also fixed out-of-range loop counter)
GTE/Mathematics/ApprPolynomialSpecial3.h
GTE/Mathematics/ApprPolynomialSpecial4.h
GTE/Mathematics/UIntegerFP32.h
GTE/Mathematics/CLODPolyline.h
GTE/Mathematics/ConvexHull2.h
GTE/Mathematics/Delaunay2.h
GTE/Mathematics/Delaunay3.h
GTE/Mathematics/ExtremalQuery3.h
GTE/Mathematics/ExtremalQuery3BSP.h
GTE/Mathematics/GenerateMeshUV.h
GTE/Mathematics/MinimalCycleBasis.h
GTE/Mathematics/NearestNeighborQuery.h
GTE/Mathematics/SplitMeshByPlane.h
GTE/Mathematics/TriangulateEC.h
GTE/Mathematics/BasisFunction.h
GTE/Mathematics/BezierCurve.h
GTE/Mathematics/BSplineCurveFit.h
GTE/Mathematics/BSplineReduction.h
GTE/Mathematics/BSplineSurface.h
GTE/Mathematics/BSplineSurfaceFit.h
GTE/Mathematics/NaturalSplineCurve.h
GTE/Mathematics/NURBSSurface.h
GTE/Mathematics/ParametricCurve.h
GTE/Mathematics/RiemannianGeodesic.h

GTE/Mathematics/TCBSplineCurve.h
GTE/Mathematics/DistOrientedBox3Cone3.h
GTE/Mathematics/DistPoint3ConvexPolyhedron3.h
GTE/Mathematics/AdaptiveSkeletonClimbing2.h
GTE/Mathematics/AdaptiveSkeletonClimbing3.h
GTE/Mathematics/CurveExtractor.h
GTE/Mathematics/Image2.h
GTE/Mathematics/Image3.h
GTE/Mathematics/PdeFilter1.h
GTE/Mathematics/PdeFilter2.h
GTE/Mathematics/PdeFilter3.h
GTE/Mathematics/SurfaceExtractor.h
GTE/Mathematics/SurfaceExtractorCubes.h
GTE/Mathematics/IntpAkima1.h
GTE/Mathematics/IntpAkimaNonuniform1.h
GTE/Mathematics/IntpAkimaUniform1.h
GTE/Mathematics/IntpAkimaUniform2.h
GTE/Mathematics/IntpAkimaUniform3.h
GTE/Mathematics/IntpBSplineUniform.h
GTE/Mathematics/IntpLinearNonuniform2.h
GTE/Mathematics/IntpLinearNonuniform3.h
GTE/Mathematics/IntrArc2Arc2.h
GTE/Mathematics/Integration.h
GTE/Mathematics/LCPSolver.h
GTE/Mathematics/LDLTDecomposition.h
GTE/Mathematics/LinearSystem.h
GTE/Mathematics/MinimizeN.h
GTE/Mathematics/QuarticRootsQR.h
GTE/Mathematics/RootsPolynomial.h
GTE/Mathematics/SingularValueDecomposition.h
GTE/Mathematics/SymmetricEigensolver.h
GTE/Mathematics/UnsymmetricEigenvalues.h
GTE/Mathematics/Polygon2.h
GTE/Mathematics/IndexAttribute.h
GTE/Mathematics/PlanarMesh.h
GTE/Mathematics/RevolutionMesh.h
GTE/Mathematics/BoxManager.h
GTE/Mathematics/MassSpringCurve.h
GTE/Mathematics/MassSpringSurface.h
GTE/Mathematics/MassSpringVolume.h
GTE/Mathematics/RectangleManager.h

The following list are files that contain constructors that do not initialize class members. Sometimes the MSVS code analysis tool will complain about constructor calls such as `MyClass object;`, saying that `object` might not be initialized. In some cases `object` is initialized but the tool seems not to dig deeply enough to figure this out.

GTE/Applications/Environment.h

GTE/Mathematics/BandedMatrix.h
GTE/Mathematics/ConvertCoordinates.h
GTE/Mathematics/GVector.h
GTE/Mathematics/GMatrix.h
GTE/Mathematics/Matrix.h (for nested class Table, initialized mStorage to zero)
GTE/Mathematics/Quaternion.h
GTE/Mathematics/Rotation.h
GTE/Mathematics/Transform.h
GTE/Mathematics/Vector2.h (initialized max[2] in IntrinsicVector2)
GTE/Mathematics/Vector3.h (initialized max[3] in IntrinsicVector3)
GTE/Mathematics/UIntegerFP32.h
GTE/Mathematics/ConformalMapGenus0.h
GTE/Mathematics/Delaunay3.h (in SearchInfo)
GTE/Mathematics/ExtremalQuery3.h
GTE/Mathematics/ExtremalQuery3BSP.h
GTE/Mathematics/MinimumAreaBox2.h
GTE/Mathematics/MinimumVolumeBox3.h
GTE/Mathematics/BasisFunction.h (in BasisFunctionInput)
GTE/Mathematics/BSplineGeodesic.h
GTE/Mathematics/EllipsoidGeodesic.h
GTE/Mathematics/ParametricSurface.h
GTE/Mathematics/AdaptiveSkeletonClimbing3.h (in TVertex)
GTE/Mathematics/CurveExtractor.h
GTE/Mathematics/PdeFilter.h
GTE/Mathematics/SurfaceExtractor.h
GTE/Mathematics/SurfaceExtractorCubes.h (in TVertex and VETable)
GTE/Mathematics/IntpThinPlateSpline2.h
GTE/Mathematics/IntpThinPlateSpline3.h
GTE/Mathematics/IntrAlignedBox2AlignedBox2.h (in TIQuery::Result, FIQuery::Result)
GTE/Mathematics/IntrAlignedBox2Circle2.h (in TIQuery::Result, FIQuery::Result)
GTE/Mathematics/IntrAlignedBox2OrientedBox2.h (in TIQuery::Result)
GTE/Mathematics/IntrArc2Arc2.h (in FIQuery::Result)
GTE/Mathematics/IntrCircle2Arc2.h (in FIQuery::Result)
GTE/Mathematics/IntrCircle2Circle2.h (in TIQuery::Result, FIQuery::Result)
GTE/Mathematics/IntrDisk2Sector2.h (in TIQuery::Result)
GTE/Mathematics/IntrEllipse2Ellipse2.h (in FIQuery::Result and EllipseInfo)
GTE/Mathematics/IntrHalfspace2Polygon2.h (in FIQuery::Result)
GTE/Mathematics/IntrLine2AlignedBox2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrLine2Arc2.h (in TIQuery::Result)
GTE/Mathematics/IntrLine2Circle2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrLine2OrientedBox.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrOrientedBox2Circle2.h (in TIQuery::Result)
GTE/Mathematics/IntrOrientedBox2Cone2.h (in TIQuery::Result)
GTE/Mathematics/IntrOrientedBox2OrientedBox.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrOrientedBox2Sector2.h (in TIQuery::Result)
GTE/Mathematics/IntrRay2AlignedBox2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrRay2Arc2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrRay2Circle2.h (in TIQuery::Result and FIQuery::Result)

GTE/Mathematics/IntrRay2OrientedBox2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrRay2Ray2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrRay2Segment2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrRay2Triangle2.h (in FIQuery::Result)
GTE/Mathematics/IntrSegment2AlignedBox2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSegment2Arc2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSegment2Circle2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSegment2OrientedBox2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSegment2Segment2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSegment2Triangle2.h (in FIQuery::Result)
GTE/Mathematics/IntrTriangle2Triangle2.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrOrientedBox2Cone2.h (in TIQuery::Result)
GTE/Mathematics/IntrAlignedBox3AlignedBox3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrAlignedBox3Cone3.h (in TIQuery::Result)
GTE/Mathematics/IntrAlignedBox3Sphere3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrCapsule3Capsule3.h (in TIQuery::Result)
GTE/Mathematics/IntrConvexMesh3Plane3.h (in FIQuery::Result)
GTE/Mathematics/IntrEllipsoid3Ellipsoid3.h (in TIQuery::Result)
GTE/Mathematics/IntrHalfspace3Capsule3.h (in TIQuery::Result)
GTE/Mathematics/IntrHalfspace3Cylinder3.h (in TIQuery::Result)
GTE/Mathematics/IntrHalfspace3Ellipsoid3.h (in TIQuery::Result)
GTE/Mathematics/IntrHalfspace3OrientedBox3.h (in TIQuery::Result)
GTE/Mathematics/IntrHalfspace3Segment3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrHalfspace3Sphere3.h (in TIQuery::Result)
GTE/Mathematics/IntrHalfspace3Triangle3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrLine3Cone3.h (in FIQuery::Result)
GTE/Mathematics/IntrLine3Plane3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrLine3Triangle3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrOrientedBox3Cone3.h (in TIQuery::Result)
GTE/Mathematics/IntrOrientedBox3Cylinder3.h (in TIQuery::Result)
GTE/Mathematics/IntrOrientedBox3Frustum3.h (in TIQuery::Result)
GTE/Mathematics/IntrOrientedBox3Sphere3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrPlane3Capsule3.h (in TIQuery::Result)
GTE/Mathematics/IntrPlane3Cylinder3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrPlane3Ellipsoid3.h (in TIQuery::Result)
GTE/Mathematics/IntrPlane3OrientedBox3.h (in TIQuery::Result)
GTE/Mathematics/IntrPlane3Plane3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrPlane3Sphere3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrPlane3Triangle3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrRay3AlignedBox3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrRay3Capsule3.h (in FIQuery::Result)
GTE/Mathematics/IntrRay3Cone3.h (in FIQuery::Result)
GTE/Mathematics/IntrRay3Cylinder3.h (in FIQuery::Result)
GTE/Mathematics/IntrRay3Ellipsoid3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrRay3OrientedBox3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrRay3Plane3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrRay3Sphere3.h (in FIQuery::Result)
GTE/Mathematics/IntrRay3Triangle3.h (in TIQuery::Result and FIQuery::Result)

GTE/Mathematics/IntrSegment3AlignedBox3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSegment3Capsule3.h (in FIQuery::Result)
GTE/Mathematics/IntrSegment3Cone3.h (in FIQuery::Result)
GTE/Mathematics/IntrSegment3Cylinder3.h (in FIQuery::Result)
GTE/Mathematics/IntrSegment3Ellipsoid3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSegment3OrientedBox3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSegment3Plane3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSegment3Sphere3.h (in FIQuery::Result)
GTE/Mathematics/IntrSegment3Triangle3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSphere3Cone3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSphere3Frustum3.h (in TIQuery::Result)
GTE/Mathematics/IntrSphere3Sphere3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/IntrSphere3Triangle3.h (in FIQuery::Result and FIQuery::ExactResult)
GTE/Mathematics/IntrTriangle3OrientedBox3.h (in TIQuery::Result and FIQuery::Result)
GTE/Mathematics/Mesh.h
GTE/Mathematics/SymmetricEigensolver.h (in SortItem)
GTE/Mathematics/UnsymmetricEigenvalues.h

The following list are files that contain unscoped enumerations enum Name. I modified these to be scoped enum class Name.

GTE/Mathematics/EulerAngles.h (EulerResult)
GTE/Mathematics/Rotation.h (RepresentationType, new name is Type)
GTE/Mathematics/GenerateMeshUV.h (INTERIOR_VERTEX modified to a constexpr)
GTE/Mathematics/PrimalQuery2.h (OrderType)
GTE/Mathematics/QuadricSurface.h (Classification)
GTE/Mathematics/PdeFilter.h (ScaleType)
GTE/Mathematics/IntrArc2Arc2.h (enum of FIQuery modified to enum class Configuration)
GTE/Mathematics/IntrEllipse2Ellipse2.h (enum of TIQuery modified to enum class Classification)
GTE/Mathematics/IntrEllipse2Ellipse2.h (enum of AreaResult modified to enum class Configuration)
GTE/Mathematics/IntrAlignedBox3Cone3.h (enum to static size_t constexpr)
GTE/Mathematics/IntrEllipsoid3Ellipsoid3.h (enum to enum class Classification)

Replaced auto desc by auto const& desc. I intended a copy but a const reference is sufficient.

GTE/Graphics/DX11/DX11Engine.cpp

Added a cast to GLintptr in the buffer file. Added a cast of size_t to the engine file.

GTE/Graphics/GL45/GL45Buffer.cpp
GTE/Graphics/GL45/GL45Engine.cpp

The code analysis tool generates a warning for expressions such as

```
int i0 = 1, i1 = 2;
double d = static_cast<double>(i1 - i0);
```

stating that i1 - i0 is the difference of 4-byte quantities and is being cast to an 8-byte quantity. To avoid the warning,

```
int i0 = 1, i1 = 2;
double d = static_cast<double>(i1) - static_cast<double>(i0);
```

I do not believe such a warning should be generated.

GTE/Mathematics/BSNumber.h
GTE/Mathematics/OBBTreeForPoints.h
GTE/Mathematics/Histogram.h
GTE/Mathematics/IntpBSplineUniform.h
GTE/Mathematics/IntpBSplineUniform.h
GTE/Mathematics/IntpBicubic.h
GTE/Mathematics/IntpBilinear.h
GTE/Mathematics/IntpTricubic.h
GTE/Mathematics/IntpTrilinear.h
GTE/Mathematics/RevolutionMesh.h

Other casts appear to be required. The UIntegerALU32 needs a cast of 1 to static_cast<size_t>(1) in a shift operation. The UIntegerAP32 needs a cast in the constructor with uint64_t argument and in SetNumBits.

GTE/Mathematics/UIntegerALU32.h
GTE/Mathematics/UIntegerAP32.h

Added noexcept to the move constructor and move assignment operator.

GTE/Mathematics/BSPPolygon2.h
GTE/Mathematics/DisjointIntervals.h
GTE/Mathematics/DisjointRectangles.h
GTE/Mathematics/Image.h
GTE/Mathematics/Image2.h
GTE/Mathematics/Image3.h

Modified macro arguments involving level to use std::to_string(level). In GL45SamplerState, removed the argument samplerState→filter from the LogError message. The filter will eventually become a scoped enumeration.

GTE/Graphics/GL45/GL45TextureArray.cpp
GTE/Graphics/GL45/GL45TextureSingle.cpp
GTE/Graphics/GL45/GL45SamplerState.cpp

Added comments about MSVS 2019 code analysis warnings that appear to be incorrect.

GTE/Mathematics/SymmetricEigensolver.h
GTE/Mathematics/Hyperellipsoid.h

The files had hard-coded throw calls. These were replaced with LogAssert or LogError macros.

GTE/Mathematics/PlanarMesh.h
GTE/Mathematics/TetrahedraRasterizer.h

Added the ability to specify an infinite cylinder using interfaces rather than setting the height to a maximum value.

GTE/Mathematics/Cylinder.h

November 2, 2021. In FillRegion, accessing tkey from the front of regionQueue using a const reference is incorrect, because the next line of code is a pop of the queue. A copy must be made of the front of the queue. Thanks to Aaron Barony for tracking down the problem.

GTE/Mathematics/TriangulateCDT.h

October 20, 2021. The project settings had incorrect paths to the dependent libraries and to the GTE header files.

GTE/Tools/FiniteDifferences/FiniteDifferences*.{vcxproj,sln}

October 19, 2021. Added a new file to build libraries and samples together on Linux.

GTE/CMakeLibrariesAndSamples.sh

October 17, 2021. Modified enum Type to enum class Type.

GTE/Mathematics/BSPrecision.h
GTE/Tools/PrecisionCalculator/PrecisionCalculator.cpp

Added initialization of class members to the constructor. Switched to using int32_t instead of int.

GTE/Mathematics/ConvexMesh3.h

Replaced SomeClass<3, T> by SomeClass3<T> for consistency when implementing 3-dimensional algorithms.

GTE/Mathematics/DistOrientedBox3Cone3.h

GTE Vector constructors do not initialize their data (GTL does). Added an initialization of closest.

GTE/Mathematics/DistPoint3Circle3.h
GTE/Mathematics/DistPointTriangle.h

Added initialization of class members to the Result constructor.

GTE/Mathematics/DistPoint3ConvexPolyhedron3.h
GTE/Mathematics/DistLine3Circle3.h
GTE/Mathematics/DistCircle3Circle3.h
GTE/Samples/Distance/DistancePointConvexPolyhedron/DistancePointConvexPolyhedronWindow3.cpp

Modified the Result to have a squared-distance member and an array of two closest points, the input point and the closest cylinder point. Added initialization of class members to the Result constructor.

GTE/Mathematics/DistPoint3Cylinder3.h

Modified the Result to have an array of two closest points, the input point and the closest frustum point. Added initialization of class members to the Result constructor.

GTE/Mathematics/DistPoint3Frustum3.h

October 15, 2021. Ported the GTE code to GTL with modifications in both tracks, more comments and unit tests.

GTE/Mathematics/IntrLine2Line2.h
GTE/Mathematics/IntrLine2Ray2.h
GTE/Mathematics/IntrLine2Segment2.h

The application created an index buffer for each tetrahedron, but only one is needed and can be shared by all tetrahedra. The vertex buffer for each tetrahedron created enough space for all the points in the dataset, but only 4 vertices are needed per buffer.

GTE/Samples/Geometrics/Delaunay3D/Delaunay3DWindow3.{h,cpp}

Modified the axis member to be of type std::array. Moved the initialization of axis to the initializer portion of the constructor.

GTE/Mathematics/Ellipse3.h

Modified enum Result to enum class Result. Added initialization to the Variable constructor.

GTE/Mathematics/LCPSolver.h

October 12, 2021. The numTriangles1 variable was initialized incorrectly.

GTE/Mathematics/SeparatePoints3.h

September 30, 2021. Fixed a bug where the return value was not of type Result. The Result constructor had an input inEpsilon, but the caller of operator()(...) has no chance to specify this. The operator function now has a third parameter epsilon, defaulted to 0 to avoid breaking application code. The addition of explicit to the Result constructor in IntrTriangle3Cylinder3 appears to be unnecessary because all branches the code can take have return of a Result object. It is not possible to reach the end of the if-else statements without exiting, so there is no need for a return statement as the last line of the function. However, I modified the code to set result.intersect in each of the blocks and added a single return statement as the last line.

GTE/Mathematics/IntrOrientedBox3OrientedBox3.h
GTE/Mathematics/IntrAlignedBox3OrientedBox3.h
GTE/Mathematics/IntrTriangle3Cylinder3.h

September 29, 2021. Several of the result members in the FIQuery were uninitialized. Added initialization in the default constructor for Result. Removed the isPoints member (not essential to the query), setting numIntersections to the maximum size_t to indicate when the circle is contained fully by the plane.

GTE/Mathematics/IntrPlane3Circle3.h

14 Updates to Version 5.12

September 25, 2021. The August 24, 2021 modification replaced the GTE Math.h by the C++ header <cmath>. The eigensolver code also needs <limits> which Math.h provided. The code compiled with Microsoft Visual Studio 2019 (latest) and Ubuntu 20.04.1 LTS gcc 9.3.0 but not Fedora 34 gcc 11.1.1.

GTE/Mathematics/SymmetricEigensolver3x3.h

The function GetUyy had F[zp1][yp1][zp1], but it should be F[zp1][yp1][xp1].

GTE/Mathematics/PdeFilter3.h

September 23, 2021. An occurrence of float should have been Real in the second Extract function.

GTE/Mathematics/SurfaceExtractorMC.h

The expressions in GetGradient using the float number 1.0f needed to have (Real)1.

GTE/Mathematics/SurfaceExtractorCubes.h

Fixed a comment about the constraints on the number of control points.

GTE/Mathematics/BSplineCurveFit.h

The mask type in the constructors was bool const*, which is from old code that was ported. The type should be instead int const*.

GTE/Mathematics/PdeFilter1.h
GTE/Mathematics/PdeFilter2.h
GTE/Mathematics/PdeFilter3.h
GTE/Mathematics/CurvatureFlow2.h
GTE/Mathematics/CurvatureFlow3.h
GTE/Mathematics/GaussianBlur2.h
GTE/Mathematics/GaussianBlur3.h
GTE/Mathematics/GradientAnisotropic2.h
GTE/Mathematics/GradientAnisotropic3.h

September 20, 2021. In the tests for modifying DistPoint3Circle3.h, the Logger.h header was implicitly included in the environment before the query header file. The inclusion must be explicit.

GTE/Mathematics/DistPoint3Circle3.h

September 19, 2021. The source code did not match the description for computing the distance robustly in the PDF entitled *Distance to Circles in 3D* PDF (Section 2). It computed a closest point first, which was not robust, and then computed the distance using the input point and that closest point. The robustness problem occurs when the input point is nearly on the normal line C+tN, where C is the circle center and N is a unit-length normal to the plane of the circle. The distance computation now matches the PDF and is robust. The closest-point code was revised to be robust, and the PDF was expanded to describe the new code.

GTE/Mathematics/DistPoint3Circle3.h

September 7, 2021. Two loops in DoLeastSquares had incorrect upper bounds.

GTE/Mathematics/ApprPolynomialSpecial3.h

The last blocks of code in function ApprCone3::ComputeInitialCone. The roots were supposed to be searched for the minimum, but the code for updating minItem did not update the minError. The quartic roots are generally stored in increasing order except when a repeated root 0 is found (for the depressed quartic). The cone fitting should not lead to a root of 0, so the previous code worked. The least-squares fitting PDF was updated accordingly.

GTE/Mathematics/ApprCone3.h

September 6, 2021. Implemented the test-intersection query for two finite cylinders that is described in IntersectionOfCylinders.pdf. The original version of the PDF showed that generally computing a separating axis requires minimizing a convex function. No details were provided about implementing the minimizer. In an attempt to implement the query, it was clear that a deeper analysis was required. The latest version of the PDF has those details. Added a sample application to illustrate the ideas.

GTE/BuildAll*.sln
GTE/GTMathematics{v14,v15,v16}.vcxproj
GTE/GTMathematics{v14,v15,v16}.vcxproj.filters
GTE/Mathematics/IntrCylinder3Cylinder3.h
GTE/Samples/Intersection/IntersectCylinders/IntersectCylinders*.*
GTE/Samples/Intersection/IntersectCylinders/.vscode/launch.json
GTE/Samples/Intersection/IntersectCylinders/.vscode/settings.json
GTE/Samples/Intersection/IntersectCylinders/CMakeLists.txt
GTE/Samples/Intersection/IntersectCylinders/CMakeSample.sh
GTE/Samples/Intersection/IntersectCylinders/cmake-variants.json
GTE/Samples/Intersection/CMakeLists.txt

August 24, 2021. The GetCosSin function was failing when the floating-point model was set to /fp:fast instead of the default model /fp:precise used by GTE. The function was simplified to avoid the problem. The

PDF documentation (*A Robust Eigensolver for* 3×3 *Symmetric Matrices*) and the source code were revised to match each other.

GTE/Mathematics/SymmetricEigensolver3x3.h

15 Updates to Version 5.11

August 1, 2021. Most of the distance subsystem of GTE was revised or rewritten, mainly to avoid the LCP solver which was a performance problem when object features were (nearly) parallel. The subsystem consists of linear algebraic objects: point, line, ray, segment, triangle, rectangle, plane, box. The Result objects now have constructors that initialize the members explicitly, and the member names are consistent among files. Derivation of distance queries involving aligned boxes and oriented boxes was removed by factoring out *canonical boxes*, which are aligned boxes with centers at the origin. An aligned box can be reduced to a canonical box by a translation. An oriented box can be reduced to a canonical box by a rotation and a translation. Unit tests were updated, and some classes needed unit testing.

Several bugs were discovered and fixed.

- DistPointOrientedBox.h: Template alias for DCPPointOrientedBox had AlignedBox for the last parameter.
- DistSegment3OrientedBox3.h: Had segExtent instead of -segExtent in the last 'else' block.
- Bug fix DistSegment3Rectangle3.h: Had segExtent instead of -segExtent in the last 'else' block.

GTE/Mathematics/DistLine2AlignedBox2.h
GTE/Mathematics/DistRay2AlignedBox2.h
GTE/Mathematics/DistSegment2AlignedBox2.h
GTE/Mathematics/DistLine2OrientedBox2.h
GTE/Mathematics/DistRay2OrientedBox2.h
GTE/Mathematics/DistSegment2OrientedBox2.h
GTE/Mathematics/DistLine3AlignedBox3.h
GTE/Mathematics/DistLine3CanonicalBox3.h
GTE/Mathematics/DistLine3OrientedBox3.h
GTE/Mathematics/DistLine3Rectangle3.h
GTE/Mathematics/DistLine3Triangle3.h
GTE/Mathematics/DistOrientedBox3OrientedBox3.h
GTE/Mathematics/DistPlane3AlignedBox3.h
GTE/Mathematics/DistPlane3CanonicalBox3.h
GTE/Mathematics/DistPlane3OrientedBox3.h
GTE/Mathematics/DistPoint3Tetrahedron3.h
GTE/Mathematics/DistRay3AlignedBox3.h
GTE/Mathematics/DistRay3CanonicalBox3.h
GTE/Mathematics/DistRay3OrientedBox3.h
GTE/Mathematics/DistRay3Rectangle3.h
GTE/Mathematics/DistRay3Triangle3.h
GTE/Mathematics/DistRectangle3AlignedBox3.h

GTE/Mathematics/DistRectangle3CanonicalBox3.h
GTE/Mathematics/DistRectangle3OrientedBox3.h
GTE/Mathematics/DistSegment3AlignedBox3.h
GTE/Mathematics/DistSegment3CanonicalBox3.h
GTE/Mathematics/DistSegment3OrientedBox3.h
GTE/Mathematics/DistSegment3Rectangle3.h
GTE/Mathematics/DistSegment3Triangle3.h
GTE/Mathematics/DistTriangle3AlignedBox3.h
GTE/Mathematics/DistTriangle3CanonicalBox3.h
GTE/Mathematics/DistTriangle3OrientedBox3.h
GTE/Mathematics/DistAlignedBoxAlignedBox.h
GTE/Mathematics/DistLineLine.h
GTE/Mathematics/DistLineRay.h
GTE/Mathematics/DistLineSegment.h
GTE/Mathematics/DistPointAlignedBox.h
GTE/Mathematics/DistPointCanonicalBox.h
GTE/Mathematics/DistPointHyperellipsoid.h
GTE/Mathematics/DistPointHyperplane.h GTE/Mathematics/DistPointLine.h
GTE/Mathematics/DistPointOrientedBox.h
GTE/Mathematics/DistPointRay.h
GTE/Mathematics/DistPointRectangle.h
GTE/Mathematics/DistPointSegment.h
GTE/Mathematics/DistPointTriangle.h
GTE/Mathematics/DistRayRay.h
GTE/Mathematics/DistRaySegment.h
GTE/Mathematics/DistSegmentSegment.h
GTE/Mathematics/CanonicalBox.h
GTE/Graphics/Picker.cpp
GTE/Mathematics/ContLozenge3.h
GTE/Mathematics/IntrLine3Plane3.h
GTE/Mathematics/IntrPlane3Capsule3.h
GTE/Mathematics/IntrPlane3Cylinder3.h
GTE/Mathematics/IntrPlane3Ellipsoid3.h
GTE/Mathematics/IntrPlane3OrientedBox3.h
GTE/Mathematics/IntrPlane3Sphere3.h
GTE/Mathematics/IntrSphere3Triangle3.h
GTE/Mathematics/Rectangle.h
GTE/Mathematics/SplitMeshByPlane.h
GTE/Samples/Distance/DistanceAlignedBoxes/DistanceAlignedBoxesWindow3.cpp
GTE/Samples/Distance/DistanceAligedBoxOrientedBox/DistanceAlignedBoxOrientedBoxWindow3.cpp
GTE/Samples/Distance/DistanceOrientedBoxes/DistanceOrientedBoxesWindow3.cpp
GTE/Samples/Distance/DistancePointHyperellipsoid/DistancePointHyperellipsoidConsole.cpp
GTE/Samples/Distance/DistanceRectangleBox/DistanceRectangleBoxWindow3.cpp
GTE/Samples/Distance/DistanceTriangleBox/DistanceTriangleBoxWindow3.cpp
GTE/Samples/Distance/DistanceSegments3/DistanceSegments3Console.h
GTE/Samples/Distance/DistanceSegments3/DistanceSegments3Console.cpp
GTE/Samples/Intersection/IntersectConvexPolyhedra/ConvexPolyhedron.h

Added GetVertices member function.

GTE/Mathematics/AlignedBox.h

Small modification to avoid typecasting from integer to floating point.

GTE/Mathematics/OrientedBox.h

Added origin member, a point on the hyperplane.

GTE/Mathematics/Hyperplane.h

The 3D point-to-plane and point-to-rectangle queries works in general dimensions. The 3D files were renamed to DistPointHyperplane.h and DistPointRectangle.h.

GTE/Mathematics/DistPoint3Plane3.h
GTE/Mathematics/DistPoint3Rectangle3.h

The exact rational arithmetic code for point-triangle distance was moved previously into DistPointTriangle.h.

GTE/Mathematics/DistPointTriangleExact.h

The exact rational arithmetic code for segment-segment distance was moved into DistSegmentSegment.h.

GTE/Mathematics/DistSegmentSegmentExact.h

July 12, 2021. Reordered the coefficients of the fitted quadratic functions. Modified the coefficients argument to use std::array. This allows the array of output coefficients of ApprQuadratic3::operator() to be passed directly as an input to the QuadricSurface constructor. **NOTE: The reordering will affect any of your applications that use the quadratic approximations.**

GTE/Mathematics/ApprQuadratic2.h
GTE/Mathematics/ApprQuadratic3.h

July 9, 2021. Added new code and sample for computing the distance in 2D between a box (aligned or oriented) and a linear component (line, ray or segment).

GTE/BuildAll*.sln
GTE/GTMathematics{v14,v15,v16}.vcxproj
GTE/GTMathematics{v14,v15,v16}.vcxproj.filters
GTE/Mathematics/DistLine2AlignedBox2.h
GTE/Mathematics/DistLine2OrientedBox2.h
GTE/Mathematics/DistRay2AlignedBox2.h
GTE/Mathematics/DistRay2OrientedBox2.h
GTE/Mathematics/DistSegment2AlignedBox2.h

GTE/Mathematics/DistSegment2OrientedBox2.h
GTE/Samples/Distance/DistanceLine2Box2/DistanceLine2Box2*.*
GTE/Samples/Distance/DistanceLine2Box2/.vscode/launch.json
GTE/Samples/Distance/DistanceLine2Box2/.vscode/settings.json
GTE/Samples/Distance/DistanceLine2Box2/CMakeLists.txt
GTE/Samples/Distance/DistanceLine2Box2/CMakeSample.sh
GTE/Samples/Distance/DistanceLine2Box2/cmake-variants.json
GTE/Samples/Distance/CMakeLists.txt

June 21, 2021. When the line parameter for the point-ray distance test is negative, the Result.parameter member was not clamped to 0.

GTE/Mathematics/DistPointRay.h

Fixed a copy-and-paste error that should have been trapped by my instantiation tool; replaced T by Real.

GTE/Mathematics/IntrRay2Triangle2.h

June 20, 2021. Ported from Wild Magic to GTE the test-intersection and find-intersection queries for two stationary triangles in 3D. The template type can be floating-point, but rounding errors can cause misclassifications when two triangles are nearly touching. The template type can also be an arbitrary precision type, such as BSRational<UIntegerAP32>, and the intersection set (if any) is theoretically correct.

GTE/GTMathematics{v14,v15,v16}.vcxproj
GTE/GTMathematics{v14,v15,v16}.vcxproj.filters
GTE/Mathematics/IntrTriangle3Triangle3.h

June 17, 2021. Modified the 2D line-triangle find-intersection query to allow for Line2 objects with direction vectors that are not unit length. This is necessary to support exact arithmetic in the forthcoming port of IntrTriangle3Triangle3 from Wild Magic to GTE. This query projects points from 3D to 2D and calls the IntrSegment2Triangle2 query.

The modified IntrSegment2Triangle2 is a breaking change. In the previous code, the Result.parameter[] values were relative to the center-direction-extent form of the segment, where $t \in [-e, e]$ for extent e. The modified code does not convert the segment to center-direction-extent form in order to avoid floating-point rounding errors when computing the unit-length direction vector. Consequently, the modified code now has Result.parameter[] values relative to $[0, 1]$ where the segment is parameterized by $p[0] + t(p[1] - p[0])$ for $t \in [0, 1]$.

GTE/Mathematics/IntrLine2Triangle2.h
GTE/Mathematics/IntrRay2Triangle2.h
GTE/Mathematics/IntrSegment2Triangle2.h

Cosmetic changes to make the post-line-query code the same for all find-intersection queries of the code modified in the June 14, 2021 entry.

GTE/Mathematics/IntrSegment3AlignedBox3.h
GTE/Mathematics/IntrSegment3Capsule3.h
GTE/Mathematics/IntrSegment3Cylinder3.h
GTE/Mathematics/IntrSegment3Ellipsoid3.h
GTE/Mathematics/IntrSegment3Sphere3.h

16 Updates to Version 5.10

June 14, 2021. Modified the Result structures to use consistent naming for the intersection queries between convex volumetric objects and linear components (lines, rays, segments). The code for clamping line-object results to ray-object and segment-object has also been made consistent. The std::numeric_limits<T>::max() in the ray-object blocks was removed as a step forward to supporting exact arithmetic using QFNumber in GTL.

GTE/Mathematics/IntrLine3AlignedBox3.h
GTE/Mathematics/IntrLine3Capsule3.h
GTE/Mathematics/IntrLine3Cylinder3.h
GTE/Mathematics/IntrLine3Ellipsoid3.h
GTE/Mathematics/IntrLine3OrientedBox3.h
GTE/Mathematics/IntrLine3Sphere3.h
GTE/Mathematics/IntrRay3AlignedBox3.h
GTE/Mathematics/IntrRay3Capsule3.h
GTE/Mathematics/IntrRay3Cylinder3.h
GTE/Mathematics/IntrRay3Ellipsoid3.h
GTE/Mathematics/IntrRay3OrientedBox3.h
GTE/Mathematics/IntrRay3Sphere3.h
GTE/Mathematics/IntrSegment3AlignedBox3.h
GTE/Mathematics/IntrSegment3Capsule3.h
GTE/Mathematics/IntrSegment3Cylinder3.h
GTE/Mathematics/IntrSegment3Ellipsoid3.h
GTE/Mathematics/IntrSegment3OrientedBox3.h
GTE/Mathematics/IntrSegment3Sphere3.h

June 9, 2021. The modification of 21 May 2021 fixed a logic error for subdivision, but it also exposed an edge case that is overly expensive when the minimum occurs at a parabola vertex. I modified the logic in the bracketed minimum search not to subdivide. Instead, only bisection is applied.

GTE/Mathematics/Minimize1.h

Ported the code for perspective projection of an ellipsoid onto a plane from Wild Magic to GTE.

GTE/Mathematics/Projection.h

May 27, 2021. Added support for fitting a cylinder to a triangle mesh. I also made some cosmetic code changes for the style associated with the port to GTL.

GTE/Mathematics/ApprCylinder3.h

The Intel C++ Compiler 19.2 warned about a signed-unsigned mismatch in two loops.

GTE/Applications/MSW/WICFileIONative.cpp

May 24, 2021. After installing Fedora 34 with gcc 11.1.1, several files led to compiler failures. For example, DisjointIntervals.h has a single header include of <vector>. A source cpp file that includes DisjointIntervals.h and then references size_t will fail to compile, the compiler suggesting you might have meant std::size_t. This implies that the <vector> implementation uses only std::size_t and does not expose access to a namespace-less size_t. The file HashCombine.h has a single header include of <functional> but has the same problem with size_t.

GTE/Mathematics/DisjointIntervals.h
GTE/Mathematics/HashCombine.h
GTE/Mathematics/Image.h
GTE/Mathematics/MinHeap.h

17 Updates to Version 5.9

May 23, 2021. The IncrementalDelaunay2 triangulation (with insertions and/or removals) uses a supertriangle approach. The construction of the insertion polygon can still include triangles sharing a supervertex, and in some cases the polygon is incorrect. Delaunay2 avoids the pitfalls of the supertriangle approach by not having one. To avoid a complete redesign, the IncrementalDelaunay2 code was modified to create the supertriangle with the input bounding rectangle included in the initial triangulation. The insertion polygon is now guaranteed to include only Delaunay triangles but not triangles having at least one supervertex. After all insertions and removals, if you do not want the input bounding rectangle as part of the triangulation, a new member function FinalizeTriangulation can be called to remove all triangles sharing an input rectangle vertex. After this call, you can no longer insert or remove points; that is, the triangulation really is final. I also fixed the version number for the header file and for the 3 files in the sample application. They all had 4.8.* when they should have had 5.8.*.

GTE/Mathematics/IncrementalDelaunay2.h
GTE/Samples/Geometrics/IncrementalDelaunay2/IncrementalDelaunay2Main.cpp
GTE/Samples/Geometrics/IncrementalDelaunay2/IncrementalDelaunay2Window2.{h,cpp}

Microsoft Visual Studio 2019 16.9.6 allowed a comparison of a size_t to an int32_t without complaint from the code analysis tool. GCC on Linux did complain.

GTE/Mathematics/LDLTDecomposition.h

I forgot to modify the target names in two sample applications.

GTE/Samples/Distance/DistanceOrientedBoxConeFrustum/CMakeLists.txt
GTE/Samples/Intersection/IntersectTriangleCylinder/CMakeLists.txt

May 22, 2021. Added new code to compute the distance and pair of closest points between a box and a cone frustum.

GTE/BuildAll*.sln
GTE/GTMathematics{v14,v15,v16}.vcxproj
GTE/GTMathematics{v14,v15,v16}.vcxproj.filters
GTE/Mathematics/DistOrientedBox3Cone3.h
GTE/Mathematics/Cone.h
GTE/Samples/Distance/DistanceOrientedBoxConeFrustum/DistanceOrientedBoxConeFrustum*.*
GTE/Samples/Distance/DistanceOrientedBoxConeFrustum/.vscode/launch.json
GTE/Samples/Distance/DistanceOrientedBoxConeFrustum/.vscode/settings.json
GTE/Samples/Distance/DistanceOrientedBoxConeFrustum/CMakeLists.txt
GTE/Samples/Distance/DistanceOrientedBoxConeFrustum/CMakeSample.sh
GTE/Samples/Distance/DistanceOrientedBoxConeFrustum/cmake-variants.json
GTE/Samples/Distance/CMakeLists.txt

I forgot to add the CMake files for the sample application that illustrates the test-intersection query for a triangle and a cylinder.

GTE/Samples/Intersection/IntersectTriangleCylinder/.vscode/launch.json
GTE/Samples/Intersection/IntersectTriangleCylinder/.vscode/settings.json
GTE/Samples/Intersection/IntersectTriangleCylinder/CMakeLists.txt
GTE/Samples/Intersection/IntersectTriangleCylinder/CMakeSample.sh
GTE/Samples/Intersection/IntersectTriangleCylinder/cmake-variants.json
GTE/Samples/Intersection/CMakeLists.txt

May 21, 2021. Fixed a bug in the subdivision logic. The logic itself was simplified to be compact and more readable.

GTE/Mathematics/Minimize1.h

May 13, 2021. Added an implementation of the LDL^{T} decomposition for positive definite matrices.

GTE/GTMathematics{v14,v15,v16}.vcxproj
GTE/GTMathematics{v14,v15,v16}.vcxproj.filters
GTE/Mathematics/LDLTDecomposition.h

May 6, 2021. The line-triangle find-intersection query needed to store the t-parameters as a pair of ordered numbers. This ensures derived classes can correctly compute interval-interval queries. The line-triangle test-intersection query determines on which side of the line the triangle vertices lie. The Boolean test for the related signed quantities was incorrectly implemented. The ray-triangle and segment-triangle queries call the line-triangle query. If that query reports an intersection, interval-interval queries are applied. It is possible that the interval queries report no-intersection, but the information was not propagated to the caller. While I was in there, I modified constructors to initialize Result members (to avoid the code analysis complaints by Microsoft Visual Studio 2019).

GTE/Mathematics/IntrLine2Triangle2.h
GTE/Mathematics/IntrRay2Triangle2.h
GTE/Mathematics/IntrSegment2Triangle2.h
GTE/Mathematics/IntrIntervals.h

April 30, 2021. The Runge-Kutta 4th-order algorithm for solving the quaternion-valued differential equation $dq/dt = \omega q/2$ uses finite-differences to estimate $q(t + \Delta t)$ (variable name newQuatOrient). These must be normalized before converting them to rotation matrices and before passing them to the force and torque functions. At the end of the update call, mQuatOrient must also be normalized.

GTE/Mathematics/RigidBody.h

Ported the test-intersection query for a triangle and a finite cylinder from Wild Magic 5 to GTE based on user request. The WM5 code had several bugs in it which were fixed during unit testing of the GTE code. A sample application is provided.

GTE/BuildAll*.sln
GTE/GTMathematics{v14,v15,v16}.vcxproj
GTE/GTMathematics{v14,v15,v16}.vcxproj.filters
GTE/Mathematics/IntrTriangle3Cylinder3.h
GTE/Samples/Intersection/IntersectTriangleCylinder/IntersectTriangleCylinder*.*

18 Updates to Version 5.8

April 26, 2021. The modifications of 24 April 2021 also required modifying the extremal query file. I had tested the sample application associated with this file, and it appeared to work. Another tool showed that the file does not compile. It is not clear why this occurred in the test environment.

GTE/Mathematics/ExtremalQuery3BSP.h

Fixed a spelling error in the top-level comment of the file.

GTE/Graphics/Node.h

I investigated the Hyperplane modification of 23 April 2021, where I thought the problem might be with the QEMU/KVM virtual machine. That is not correct. The problem turned out to be that the preprocessor settings for the project needed to include GTE_USE_ROW_MAJOR and GTE_USE_MAT_VEC. Once these are added, the plane construction for dimension 3 using singular value decomposition is accurate. I have kept the modification in place, because the algorithm for plane construction is trivial for dimension 3.

April 25, 2021. The known-problems document mentioned that GenerateMeshUVs used to work on the GPU with HLSL but then stopped working, the returned texture coordinates all NaNs. It turns out that I had introduced the bug some time ago when factoring out the GPU code to GPUGenerateMeshUVs. The GenerateMeshUVs::Vertex data structure had 3 int members for use by the HLSL shader, where that shader declared a structured buffer array with int3 members. The data structure had 4 int members, the last one padding, for use by the GLSL. That shader declared a structured buffer array with ivec3 members, but GLSL pads each element to 16 bytes as if you had declared the members ivec4. The CPU padding was conditionally compiled. The factoring removed the conditional compilation, so now the GenerateMeshUVs::Vertex is 16 bytes regardless of using HLSL or GLSL. Unfortunately, I did not modify the HLSL code to use int4 instead of int3. This is now fixed. The document also mentions that when running the GLSL version, the PlanarMesh call to GetContainingTriangle was failing some of the time, unable to find a containing triangle. This occurs for GenerateMeshUVs<double> but not for GenerateMeshUVs<float>. The failure indicates fold-overs in the planar mesh. The double-precision arithmetic in the GLSL code appears not to be of the same precision as in that of the HLSL code.

GTE/Mathematics/GenerateMeshUV.h
GTE/MathematicsGPU/GPUGenerateMeshUV.h
GTE/Samples/Geometrics/GenerateMeshUVs/GenerateMeshUVsWindow3.{h,cpp}

April 24, 2021. Replaced std::shared_ptr by std::unique_ptr in the vertex, edge and triangle maps. Replaced the use of std::weak_ptr<T> by T*. This improves the performance of the manifold meshes because of reduced memory management. (Thanks to Aaron Barany for these modifications.)

GTE/Mathematics/ConformalMapGenus0.h
GTE/Mathematics/ConstrainedDelaunay2.h
GTE/Mathematics/ConvexHull3.h
GTE/Mathematics/Delaunay2.h
GTE/Mathematics/ETManifoldMesh.h
GTE/Mathematics/GenerateMeshUV.h
GTE/Mathematics/IncrementalDelaunay2.h
GTE/Mathematics/MinimumVolumeBox3.h
GTE/Mathematics/PlanarMesh.h
GTE/Mathematics/TriangulateCDT.h
GTE/Mathematics/VertexCollapseMesh.h
GTE/Mathematics/VETManifoldMesh.h

April 23, 2021. The Hyperplane constructor whose input is a std::array of points uses singular value decomposition to build the plane through the points. The body has been replaced by a call to a function that is determined at compile time for dimension 3 and for other dimensions. This is a hack based on a bug report for the code executed on a QEMU/KVM virtual machine, which indicated the singular value decomposition was producing inaccurate results. I am unable to reproduce the problem on a non-virtual machine; the SVD works correctly for the dataset included in the bug report. I need to determine what the virtual machine is doing that causes such inaccurate results when using floating-point arithmetic.

GTE/Mathematics/Hyperplane.h

April 16, 2021. The Wild Magic 5 version of IncrementalDelaunay2 had support for computing the containing triangle for a query point. It also contained support for accessing the Delaunay triangles (not including triangles with at least one supervertex), for accessing the indices of triangles adjacent to a specified triangle, and for accesing the convex hull polygon of the Delaunay triangles. These have now been ported to the GTE version of IncrementalDelaunay2.

GTE/Mathematics/IncrementalDelaunay2.h
GTE/Samples/Geometrics/IncrementalDelaunay2/IncrementalDelaunay2Window2.{h,cpp}

April 13, 2021. Fixed a bug in the computation of E1.sqrExtent in the AreaOfIntersection function with Ellipse2 arguments. The unit tests included one for the other AreaOfIntersection but did not include one for this function.

GTE/Mathematics/IntrEllipse2Ellipse2.h

April 9, 2021. When the file was copied to another project that did not have precompiled headers, it failed to compile because of std::back_inserter. I added an include of <iterator> to fix this.

GTE/Applications/MSW/WICFileIONative.cpp

19 Updates to Version 5.7

March 26, 2021. Ported the incremental insertion and removal of points for a Delaunay 2D triangulation from Wild Magic to GTE. The Wild Magic code did not correctly handle removing points from the convex hull boundary of the triangulation. I fixed this by adding a new subsystem to correctly retriangulate the removal polygon.

GTE/BuildAll*.sln
GTE/GTMathematics{v14,v15,v16}.vcxproj
GTE/GTMathematics{v14,v15,v16}.vcxproj.filters
GTE/Mathematics/IncrementalDelaunay2.h
GTE/Samples/Geometrics/IncrementalDelaunay2/IncrementalDelaunay2*.*

Added constructor to MinHeap::Result with initialized members to avoid MSVS code analysis warnings.

GTE/Mathematics/MinHeap.h

March 15, 2021. The Delaunay2 code was failing on a platform using a virtual machine and Microsoft Visual Studio 2015. The problem is that the compiler is optimizing away some floating-point expressions that should not be due to rounding mode. I mentioned this problem in Section 3.4 of RAEFC Book Corrections. I removed the final uses of std::fesetround and std::fegetround, replacing them with SWInterval interval arithmetic support that works with the default FPU rounding mode (round to nearest ties to even).

GTE/Mathematics/APConversion.h
GTE/Mathematics/ConstrainedDelaunay2.h
GTE/Mathematics/Delaunay2.h

These modifications affect only the new ConstrainedDelaunay2 and Delaunay2 implementations. The older implementations do not use cfenv for floating-point environment changes, and they are currently tagged as *deprecated* but will exist until the full GTL distribution is available.

March 14, 2021. The deprecated code in ConstrainedDelaunay2 (the first implemention in the header file) had a bug in Retriangulate that was introduced based on my blindly following a suggestion by the Microsoft Visual Studio code analysis tool.

GTE/Mathematics/ConstrainedDelaunay2.h

The description here is long but intended to let others know to be careful about the code analysis suggestions.

The Retriangulate function originally had a line of code to access the top of the stack,

```
auto i = stack[top--];
```

The Microsoft Visual Studio code analysis tool complained 'auto' doesn't deduce references. A possibly unintended copy is being made and suggested modifying to a reference auto& (or auto const&). Without thinking about the message, I changed the code to

```
auto const& i = stack[top--];
```

which is an error, because now I have a reference to the stack[top+1] element. Later in the function,

```
stack[++top] = { i[0], isplit };
stack[++top] = { isplit, i[1] };
```

In the first line of this code, the value top is incremented first and then the pair is copied to the stack. The problem is that i is the pair to which the left-hand side of the assignment refers. The write of i[0] to this location overwrites itself (no problem), but the write of isplit to this location overwrites i[1] (the error). The second line of code reads i[1], which was inadventently modified to isplit. I have been burned several times by the faulty suggestions of the code analyzer and learned now to think about them, replacing the auto keyword by the actual type or ignoring them. The same problem had occurred in the new CDT code (the second implementation in the header file), but during unit testing I found and fixed the problem. I had not thought to go back to the old code to see whether it had the same problem. My solution was to replace

```
auto const& i = stack[top--];
```

by

```
std::array<size_t, 2> i = stack[top--];
```

20 Updates to Version 5.6

March 8, 2021. All but the first mAccumulatedLength elements are uninitialized, which causes GetTotalLength() to fail. The Release-build unit tests did not fail (via NaturalSplineCurve) using Microsoft Visual Studio because the compiler initializes the elements to zero. The compiler then optimizes away the first line of code in GetTotalLength() because it believes mAccumulatedLength.back() will return zero the first time GetTotalLength() is called. This hides the bug when using MSVS. Other compilers will most likely leave mAccumulatedLength uninitialized, in which case the bug shows up.

GTE/Mathematics/ParametricCurve.h

The BSNumber constructors that take integer inputs need to compute the debug-supporting mValue first. The previous code modifies the inputs before computing mValue, which causes the sign to be positive when the inputs are negative. The problem showed up only when GTE_BINARY_SCIENTIFIC_SHOW_DOUBLE was enabled for debugging and testing and does not affect the correctness of the actual BSNumber.

GTE/Mathematics/BSNumber.h

February 11, 2021. Added counting of maximum number of words for testing for sum of two squares and for (not yet published) Delaunay2 that computes the triangulation using a 3D convex hull for points $(x, y, x^2 + y^2)$.

GTE/Tools/PrecisionCalculator/PrecisionCalculator.cpp

February 10, 2021. The ray-sphere and segment-sphere queries were generating incorrect results when the linear component is tangent to the sphere. This occurred because result.parameter[1] is uninitialized in the line-sphere query for the tangent case, and the ray-sphere and segment-sphere code uses both result.parameter[] values for interval-interval intersection. The Result nested-class constructors now have all members initialized, and result.parameter[1] is set to result.parameter[0] in the tangent case. The same issues arise with ray-ellipsoid and segment-ellipsoid queries; they were fixed in the same manner.

GTE/Mathematics/IntrLine3Sphere3.h
GTE/Mathematics/IntrRay3Sphere3.h
GTE/Mathematics/IntrSegment3Sphere3.h
GTE/Mathematics/IntrLine3Ellipsoid3.h
GTE/Mathematics/IntrRay3Ellipsoid3.h
GTE/Mathematics/IntrSegment3Ellipsoid3.h

Added counting of maximum number of words for testing for coplanar points. Added a comment with a formula for determining the number of bytes used by BSNumber<UIntegerFP32<N>>. The value is $4(N + 4)$.

GTE/Tools/PrecisionCalculator/PrecisionCalculator.cpp

February 2, 2021. In Hull3, the incoming hull has coplanar points. When inserting the next point, the hull becomes volumetric. The incoming hull triangles are inserted into the VET mesh, but the winding orders are reversed. The triangles generated after that have the correct winding order. Fixed this.

GTE/Mathematics/ConvexHull3.h

January 29, 2021. Added debug support for counting the maximum number of blocks of bits used in UIntegerFP32<N> during application execution. The class UIntegerAP32 already has this support.

GTE/Mathematics/UIntegerFP32.h

Added counting of maximum number of words for testing for colinear points.

GTE/Tools/PrecisionCalculator/PrecisionCalculator.cpp

January 17, 2021. I figured out why MinimumVolumeBox3D was broken. It turns out that the member function CreateCompactMesh has an implicit assumption that the vertices in the VETManifoldMesh::Vertex are ordered by the integer key. This assumption was satisifed when the vertex map was a std::map, but not when the vertex map was changed to a std::unordered_map. I restored the vertex map to be an unordered map and added code to the minimum-volume box to sort the vertices before creating the compact array of vertex adjacencies.

GTE/Mathematics/MinimumVolumeBox3.h
GTE/Mathematics/VETManifoldMesh.h

January 15, 2021. The sample MinimumVolumeBox3D was broken in that the displayed box no longer contained the convex polyhedron. I traced this back to GTE 5.4 where I replaced the std::set and std::map objects by std::unordered_set and std::unordered_map objects. For a reason that I am unable to understand, the sample runs correctly when using std::map for VETManifoldMesh::VMap, but it does not run correctly when using std::unordered_map for VETManifoldMesh::VMap. The key type in either case is the native type int. I restored VMap to use std::map.

GTE/Mathematics/VETManifoldMesh.h

21 Updates to Version 5.5

January 14, 2021. The ConvexHull3 performance was absolutely awful, requiring over 1 hour to compute the hull of a 120K-vertex dataset. I reimplemented this to use incremental insertion (single-threaded) and a divide-and-conquer algorithm (multithread). On an Intel i9-10900 CPU, the incremental insertion for the 120K-vertex dataset executes in approximately 4 seconds. The divide-and-conquer algorithm using 8 threads executes in approximately 1.7 seconds. Files that depend on ConvexHull3 were modified accordingly.

GTE/Mathematics/ConvexHull3.h
GTE/Mathematics/MinimumVolumeBox3.h
GTE/Mathematics/SeparatePoints3.h
GTE/Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow3.cpp
GTE/Samples/Geometrics/ExtremalQuery/ExtremalQueryWindow3.cpp
GTE/Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow3.cpp

Added a new file for interval arithmetic. The implementation is software based and uses std::nextafter for determining the intervals. This avoids the floating-point hardware state changes when setting the rounding modes, and it avoids the problem with GCC that appears not to expose the ability to modify the floating-point environment.

GTE/GTMathematics{v14,v15,v16}.vcxproj
GTE/GTMathematics{v14,v15,v16}.vcxproj.filters
GTE/Mathematics/SWInterval.h

Replaced int V[3] by std::array<int, 3> V in the ETManifoldMesh::Triangle class. A similar change had been made already to ETManifoldMesh::Edge.

GTE/Mathematics/ETManifoldMesh.h

December 25, 2020. The DX11Engine constructors that have a window handle input are designed for rendering to the window. These constructors create default render states that GTE wants, and these states are not all the default DX11 render states. The DX11Engine constructors that do not have a window handle input are designed for windowless applications using compute shaders. These constructors do not create default render states for GTE, so the default states are those of DX11. To support a windowless application that renders to an offscreen render target, and to allow that code to work whether windowed or windowless, I added creation of the default GTE render states to the compute-intended DX11Engine constructors.

GTE/Graphics/DX11/DX11Engine.cpp

December 23, 2020. Eliminated the annoying MSVS 2019 warning about DFType being an unscoped enum, suggesting that enum class is better. I replaced the enumeration by a list of constexpr values. The type DFType is defined to be uint32_t.

GTE/Graphics/DataFormat.h

22 Updates to Version 5.4

December 18, 2020. Added code to rasterize a collection of tetrahedra into a 3D grid.

GTE/GTMathematics{v14,v15,v16}.vcxproj
GTE/GTMathematics{v14,v15,v16}.vcxproj.filters
GTE/Mathematics/TetrahedraRasterizer.h

December 1, 2020. Replaced the one remaining std::set object by a std::unordered_set object.

GTE/Mathematics/ConstrainedDelaunay2.h

While porting Delaunay2.h from GTE to GTL, the more robust GTL unit tests for the ToCircumcircle function failed for nearly cocircular points. Fixed the problem in the GTE code.

GTE/Mathematics/Delaunay2.h

November 30, 2020. Added interfaces to allow passing a raw pointer to vertices and the number of vertices.

GTE/Mathematics/Delaunay2.h
GTE/Mathematics/ConstrainedDelaunay2.h
GTE/Mathematics/TriangulateCDT.h

23 Updates to Version 5.3

November 16, 2020. Added a new file to create hash functions in order to use std::unordered_set and std::unordered_map in GTL code, replacing std::set and std::map when a performance increase is desired.

GTE/Mathematics.v*.*
GTE/Mathematics/HashCombine.h
GTE/Mathematics/FeatureKey.h
GTE/Mathematics/ETManifoldMesh.h
GTE/Mathematics/VETManifoldMesh.h

Added new classes that do not require the user to specify the computing type. The old classes are now tagged as *deprecated* and will be removed in a later version of GTE.

GTE/Mathematics/Delaunay2.h
GTE/Mathematics/ConstrainedDelaunay2.h
GTE/Mathematics/TriangulateCDT.h

Classes affected by the manifold mesh and Delaunay modifications.

GTE/Mathematics/Delaunay2Mesh.h
GTE/Mathematics/ExtremalQuery3BSP.h
GTE/Mathematics/IntpQuadraticNonuniform.h
GTE/Mathematics/IntpSphere2.h
GTE/Samples/Mathematics/Interpolation2D/Interpolation2DWindow3.h

Factored IntervalProductUp and IntervalProductDown from ConvexHull3.h to ProductLowerBound and ProductUpperBound in FPInterval.h so that the code can be shared by other computational geometry queries involving interval arithmetic.

GTE/Mathematics/ConvexHull3.h
GTE/Mathematics/FPInterval.h

Added a new member function CopyFrom that does a direct copy from UIntegerFP<N> to UIntegerFP<M> as long as N is no larger than M. This allows converting floating-point inputs to UIntegerFP<N> for N-value 2 or 4. These can then be copied on-demand into a pool of UIntegerFP<M> for much larger M that is used to represent an expression tree that must be computed with exact arithmetic. The goal is to reduce heap memory usage when computing with arbitrary-precision arithmetic. The pool can be allocated on the heap to avoid having to greatly increase the program stack size when M is large.

GTE/Mathematics/UIntegerFP32.h

Fixed a bug that was introduced when performance optimizations were added to the file. The original code had allocated vectors for mB, mC and mD. The optimized code packed these into a single chunk of memory, but the aforementioned variables which became pointers were not set to point to the proper locations in that chunk. This was only for the clamped-spline case.

GTE/Mathematics/NaturalSplineCurve.h

November 13, 2020. A conditionally compiled block that was disabled did not compile when enabled because of changes to the Vector2 constructors.

GTE/Samples/Geometrics/Delaunay2D/Delaunay2DWindow2.cpp

November 10, 2020. Added/updated comments about how large N must be to use ComputeType of BSNumber<UIntegerFP32<N>>. The performance is much better when you use UIntegerFP32 rather than UIntegerAP32 because the latter leads to significant heap management costs.

GTE/Mathematics/Delaunay2.h
GTE/Mathematics/ConstrainedDelaunay2.h
GTE/Mathematics/TriangulateCDT.h

November 8, 2020. When using ComputeType that is a floating-point type, rounding errors can cause sign misclassifications in the loop over linkEdges and prevent ProcessTriangleStrip and ProcessCoincidentEdge from being called. The consequence is that partition is an empty std::vector and the call to partition.back() causes a crash. I added a LogAssert call to throw an exception in this case. It is strongly recommended to use an arbitrary-precision type for ComputeType. Alternatively, use a try-catch block to triangulate using a floating-point type and, if an exception is thrown, repeat the call using an arbitrary-precision type. I also replaced the nondescriptive "Unexpected condition" in the LogAssert statements with function calls that return strings with more description.

GTE/Mathematics/ConstrainedDelaunay2.h

November 6, 2020. Added more members and queries to PolygonTreeEx. Previously, the insideTriangles vector stored all triangle in the polygon tree. For TriangulateCDT to be consistent with TriangulateEC, it is convenient to partition these into interiorTriangles, which are those triangles bounded by an outer polygon and its child inner polygons, and exteriorTriangles, which are those triangles bounded by an inner polygon and its child outer polygons. The point-containment query of TriangulateEC searches only the interior triangles. In fact, that class does not construct exterior triangles. I added point-containment queries for each type of vector member of PolygonTreeEx including comments about typical usage. You can create your own queries, especially if you want to use multithreading. Also, it is sometimes convenient to know which node of the polygon tree stores the containing triangle. The node index information has also been added to PolygonTreeEx for the various triangle vector members, and a couple of the point-containment queries use these.

GTE/Mathematics/PolygonTree.h
GTE/Mathematics/TriangulateCDT.h

24 Updates to Version 5.2

November 5, 2020. The ConstrainedDelaunay2 class had two bugs that needed to be fixed. The first bug was incorrect handling of adjacent triangles during retriangulation of the triangle strips containing the constrained edges. This problem shows up when the triangle strip snakes around and has a shared edge with another triangle that is not its immediate predecessor in the strip. The second bug was an incorrect computation of the pseudosquared distance function. This problem shows up when locating the closest vertex of a boundary polygon to the base polygon edge. In particular, the bug was exercised when the projections of the candidate vertices to the base edge of the polygon contained at least one projection on the segment and one projection outside the segment.

This code is approximately 13 years old and was complicated to read and debug because. In particular, it discarded the mGraph member from the Delaunay triangulation and instead used the compact arrays mIndices and mAdjacencies. I rewrote the ConstrainedDelaunay2 to continue using the mGraph member, in the process simplifying the implementation to be readable. Many comments were added, but I plan on adding a PDF to the website describing the pitfalls of the retriangulation and how to avoid them. The original code was based on an online research paper that discussed the simplest retriangulation configuration, but had no discussion about the pitfall cases. The various file changes are described next.

Reimplemented ConstrainedDelaunay2 according to the aforementioned comments.

GTE/Mathematics/ConstrainedDelaunay2.h

Factored out some code into a new function UpdateIndicesAdjacencies. This code occurred after the Delaunay triangulation to synchronize the information in mGraph with the compact arrays mIndices and mAdjacencies. To support ConstrainedDelaunay2, the factored function can be called after a user-determined set of edges have been inserted one at a time.

GTE/Mathematics/Delaunay2.h

Added two helper functions to ETManifoldMesh::Triangle, namely, GetAdjacentOfEdge and GetOppositeVertexOfEdge. I will continue to encapsulate commmon operations and add them to ETManifoldMesh. These operations tend to occur as in-application code, but that code should be shared.

GTE/Mathematics/ETManifoldMesh.h

The modifications to ConstrainedDelaunay2 required a small modification to the polygon triangulation that uses constrained Delaunay triangulation. The code copies the Delaunay mGraph object to the local graph object.

GTE/Mathematics/TriangulateCDT.h

The insertions are not batched; rather, they are executed one at a time based on key strokes. The application now needs to call UpdateIndicesAdjacencies after each edge insertion.

GTE/Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2DWindow2.cpp

October 26, 2020. I redesigned and rewrote TriangulateCDT to support shared vertices, coincident vertex-edge configurations and coincident edge-edge configurations. These configurations are *not* supported by TriangulateEC. To avoid repackaging polygon trees in an application using both classes, I factored out the nested Tree classes into a new class PolygonTree. Objects of this class are inputs to the triangulators, and they are not modified internally. I also created a new class PolygonTreeEx that currently represents the output of TriangulateCDT. Eventually, this class will become an input-output parameter for both triangulators. NOTE that TriangulateCDT does not support polygon trees where two edges intersect at an edge-interior point of both edges and that intersection point is not in the input points to the triangulator. The problem is that the constrained Delaunay triangulation works correctly when the constrained edges do not intersect other than at shared-vertex endpoints. If one were to insert two edges into the Delaunay triangulation that have an edge-interior intersection, each edge interferes with the local re-triangulation of the other edge. I plan on adding support for such edges, but the task is of significant size and will take some time.

GTE/Mathematics{v14,v15,v16}.vcxproj
GTE/Mathematics{v14,v15,v16}.vcxproj.filters
GTE/Mathematics/PolygonTree.h
GTE/Mathematics/TriangulateCDT.h
GTE/Mathematics/TriangulateEC.h
GTE/Samples/Geometrics/TriangulationEC/TriangulationECWindow2.{h,cpp}
GTE/Samples/Geometrics/TriangulationCDT/TriangulationCDTWindow2.{h,cpp}

October 26, 2020. Added to the nested class Triangle a function that tests whether a directed edge in mesh is in the same direction as an undirected triangle edge or in the opposite direction (or not a triangle edge at all). This is a common query that is now encapsulated.

GTE/Mathematics/ETManifoldMesh.h

October 26, 2020. Swap chain creation in DX11 now has "flip" flags that are recommended. The DX11Engine still uses the old flags. Apparently, the internal changes to DX11 for the flip flags sometimes causes creation of a second ID3D11Context during a swap-chain Present call in order to create an ID3D11RenderTargetView and related objects. The DXGI debug layer indicates that the back buffer is unbound from the pipeline, which is not what I thought was happening with the old swap-chain creation flags. The second context appears to be create to patch the problem. However, the time of its destruction is nondeterministic. In some cases, the destruction occurs before the GTE application mDevice is destroyed, and the FinalRelease(mDevice) is successful. In other cases, the destruction occurs after the final release of mDevice, but an exception is thrown (by the GTE code) indicating the reference count for mDevice is positive. Finally, in many cases, the second context is never created. Until I understand why this behavior occurs, I have replaced the DX11::FinalRelease calls in DX11Engine::DestroyDevice with DX11:SafeRelease calls, the latter not tracking the reference counts and not throwing exceptions.

GTE/Graphics/DX11/DX11Engine.cpp

October 11, 2020. The translation is now 3D, allowing for placement of the text within a 3D scene.

GTE/Graphics/TextEffect.{h,cpp}

25 Updates to Version 5.1

October 11, 2020. The CMakeLists.txt files were using add_compile_definitions, which requires CMake 3.12 or later. Replaced these by add_definitions. The modifications affected 135 files which will not be listed here.

October 10, 2020. Added comments that Sylvester's criterion is used to test for positive definiteness. The loop on halving the root was limited to 8 iterations, but the number has been increased to allow rational arithmetic for which an unlimited loop could be infinite. The new limit is 2048, which will not be reached for floating-point types float and double. The ApprEllipse2 code had a redundant test for the determinant of the entire matrix, but it has now been removed.

GTE/Mathematics/ApprEllipse2.h
GTE/Mathematics/ApprEllipsoid3.h

October 9, 2020. The old code ApprEllipse2 and ApprEllipsoid3 for fitting points with ellipses and ellipsoids were based on a nonlinear least-squares error function that was a sum of distances from points to the objects. The comments in those files described eigendecompositions of the M matrices but were in error, and the code did not correctly implement the mathematics. Moreover, ApprEllipsoid3 failed to initialize angle[2] before running the minimizer. The rewritten code uses a minimizer involving a 2-step gradient descent algorithm. Unit tests and end-to-end tests were added to the internal test suite to verify the algorithm and results.

GTE/Mathematics/ApprEllipse2.h
GTE/Mathematics/ApprEllipsoid3.h

26 Updates to Version 5.0

September 29, 2020. Added support to the Linux distribution for CMake executed from a command line and for Visual Studio Code. Please read the installation and release notes on how to build the libraries and samples on Linux.

The support for building the GTE libraries are in the files listed next.

GTE/.vscode/settings.json
GTE/GTE.code-workspace
GTE/CMakeLibraries.sh
GTE/CMakeLists.txt
GTE/cmake-variants.json
GTE/Applications/CMakeLists.txt
GTE/Applications/cmake-variants.json
GTE/Graphics/CMakeLists.txt
GTE/Graphics/cmake-variants.json
GTE/Mathematics/CMakeLists.txt
GTE/Mathematics/cmake-variants.json
GTE/MathematicsGPU/CMakeLists.txt
GTE/MathematicsGPU/cmake-variants.json

The CMake and Visual Studio Code support are listed next. The libraries must be built first before building any samples.

GTE/Samples/CMakeLists.txt
GTE/Samples/CMakeSamples.sh
GTE/Samples/<subfolder>/CMakeLists.txt
GTE/Samples/<subfolder>/CMakeSamples.sh
GTE/Samples/<subfolder>/<appfolder>/.vscode/launch.json
GTE/Samples/<subfolder>/<appfolder>/.vscode/settings.json
GTE/Samples/<subfolder>/<appfolder>/<appfolder>.code-workspace
GTE/Samples/<subfolder>/<appfolder>/cmake-variants.json
GTE/Samples/<subfolder>/<appfolder>/CMakeLists.txt
GTE/Samples/<subfolder>/<appfolder>/CMakeSample.sh

The sample subfolders are Distance, Geometrics, Graphics, Imagics, Intersection, Mathematics, Physics and SceneGraphs. The number of application folders is quite large and not listed here.

The shared-library executables in the Linux distribution were crashing on exit from the main program. Using gdb and the core dump, the problem appeared to be in the destruction of global objects in the gtapplications shared library. The singleton TheWindowSystem was actually created 3 times because of the way shared libraries are linked and loaded. I had added -rdynamic to the makesample.gte link line (suggested by a poster at stackoverflow). This eliminated the multiple creations of TheWindowSystem. Unfortunately, on-exit crashes still occurred, this time in the destruction of global objects in the gtgraphics shared library. The only global objects are several class-static members. To diagnose with a visual debugger, I decided to add Visual Studio Code support. With this environment, the on-exit crashes no longer occur. The CMake output files are cryptic, and I am not skilled enough to figure out what differences occur between CMake and my makefiles regarding linking and loading of shared libraries. Visual Studio Code and CMake work fine, so I am abandoning my simple Unix makefile approach.

GTE/makeapplications.gte
GTE/makegraphics.gte
GTE/makemathematicsgpu.gte
GTE/makelibraries.sh
GTE/Samples/makeallsamples.gte
GTE/Samples/makesamples.gte
GTE/Samples/makesample.gte

Moved the GTE/Mathematics/GPU folder to GTE/MathematicsGPU to support the new CMake and Visual Studio Code project organization on Linux. All the associated project source files had to be modified so the header includes access the new location. Some application files had to be modified.

GTE/GTMathematicsGPU.{v14,v15,v16}.{vcxproj,vcxproj.filters}
GTE/Samples/Geometrics/GenerateMeshUVs/GenerateMeshUVsWindows3.cpp
GTE/Samples/Physics/BlownGlass/BlownGlassWindows3.h
GTE/Samples/Physics/Fluids2D/Fluids2DWindows2.h
GTE/Samples/Physics/Fluids3D/Fluids2DWindows3.h

Visual Studio Code with g++ on Ubuntu 18.04.5 LTS complained about potentially uninitialized variables. They are initialized, but the compiler cannot determine this from the code. I set default values to avoid the warning.

GTE/Samples/Geometrics/GenerateMeshUVs/GenerateMeshUVsWindow3.cpp

Visual Studio Code with g++ on Ubuntu 18.04.5 LTS complained about potentially uninitialized variables. The vector components referred to are not initialized in some circumstances, but the program logic avoids using them in those circumstances. I initialized the vectors to zero to avoid the warning.

GTE/Mathematics/DistPointHyperellipsoid.h

September 25, 2020. The find-intersection query for moving spheres and oriented boxes calls the base-class protected query for moving spheres and aligned boxes. That call requires the center point to be in the first octant and makes adjustments accordingly. The find-intersection for the oriented boxes did not make this adjustment. I moved the transforming of the center point into the base-class query to fix the problem. The visual test harness for this code is the sample application MovingSpheresBoxes, where I apparently never moved the sphere into a position that exercised the bug.

GTE/Mathematics/IntrAlignedBox3Sphere3.h

September 18, 2020. Deprecated the class UniqueVerticesTriangles. Added a replacement, UniqueVerticesSimplices that works in any dimension 2 or larger. The class has been used for removing duplicated and unused vertices from triangle meshes and for generating meshes from a vertex-only based representation to one using an indexed representation. The generalized code applies to dimension 2 and will be used for computational geometry algorithms involving polygons (Boolean operations on polygons, for example). The deprecation is in the C++ 14 sense: the class is tagged with [[deprecated]]. It will be removed sometime in the near future.

GTE/Mathematics/UniqueVerticesSimplices.h
GTE/Mathematics/UniqueVerticesTriangles.h
GTE/Mathematics/IntrConvexMesh3Plane3.h
GTE/Mathematics/MinimumVolumeBox3.h
GTE/Mathematics/SurfaceExtractorMC.h

September 17, 2020. The BSPPolygon2::Finalize function passed *this as a non-const parameter and this→mEArray as a const parameter to the BSPTree2 constructor, which should have been a clue that the BSP implementation is badly done. A test data set was included in a bug report, and it led to an infinite loop in the BSPTree2 code because mEArray was modified via the *this object. A copy of mEArray is now passed as the second parameter of the BSPTree2 constructor.

The constructor also has a switch statement that classifies the intersection type of two edges. In the cases TRANSVERSE_POSITIVE and TRANSVERSE_NEGATIVE, it is possible that floating-point rounding error can cause the intersection point to be one of the segment endpoints. This must be trapped and handled separately. Before the change, the test data set triggered an assertion in SplitEdge for the user reporting the problem. I was unable to reproduces this, but compiled expressions using floating-point arithmetic can lead to different low-level code depending on the compiler, and the rounding behavior can vary. This is not just a matter of the low-level expression tree of operations, it can also be a matter of what precision registers are used and whether downcasts occur from a high-precision register (say, 80-bit) to a low-precision register (say, 64-bit).

GTE/Mathematics/BSPPolygon2.h

September 17, 2020. Removed the compiler define for GTE_THROW_ON_UINTEGERFP32_OUT_OF_RANGE and exposed the one line of code controlled by it in SetNumBits. An exception needs to be thrown if the bit request exceeds the number of bits provided by the class so that a caller can trap this with a try-catch block.

27 Updates to Version 4.9

September 15, 2020. Reverted the path variable GTE4_PATH to GTE_PATH.

GTE/Applications/Application.h
GTE/Applications/Environment.h
GTE/Applications/Environment.cpp

September 14, 2020. Initialized some std::array objects to avoid warnings by gcc 10.2.1 on Fedora 32 for potentially uninitialized variables. The variables are initialized, but the compiler cannot determine this solely from the code.

GTE/Mathematics/IntpQuadraticNonuniform2.h

Bypassed a couple of TriangleKey<true> constructor calls to avoid a warning by gcc 7.5.0 on Ubuntu 18.04.5 LTS regarding assumptions on strict overflow.

GTE/Samples/Imagics/AdaptiveSkeletonClimbing3.h

September 11, 2020. Updated the C#/C++ managed code projects to call the new interfaces for MinimumVolumeBox3.

GTE/Samples/CSharpCppManaged/CppLibrary/MinimumVolumeBox.{h,cpp}
GTE/Samples/CSharpCppManaged/ManagedLibrary/MinimumVolumeBox.{h,cpp}
GTE/Samples/CSharpCppManaged/CSharpApplication/Program.cs

September 11, 2020. The TIQuery code had logic:

```
if (fmid <= zero) { block0 } else if (fmid == zero) { block1 }
```

so block1 is dead code. The logic was modified to

```
if (fmid < zero) { block0 } else if (fmid == zero) { block1 }
```

September 8, 2020. Added operator+=, operator-=, operator*= and operator/= functions to support Vector<N,T> where T is FPInterval<T> or APInterval<T>.

GTE/Mathematics/APInterval.h
GTE/Mathematics/FPInterval.h

September 7, 2020. Removed a couple of commented lines that were part of the debugging and testing of the code. The virtual minimizer functions needed to be in protected scope.

GTE/Mathematics/MinimumVolumeBox3.h

28 Updates to Version 4.8

September 3, 2020. The clang compiler complained about missing istream and ostream symbols. Added header includes for these.

GTE/Mathematics/BSNumber.h

The edge type was EdgeKey<false> (unordered edge) but needed to be EdgeKey<true> (ordered edge) because the polygon needs to have ordered vertices.

GTE/Mathematics/BSPPolygon2.h

In revising the code from GTE3 to GTE4, the LogAssert(binfo...) calls in ProcessInterior should have been removed. Moreover, the −2 flag was inadvertently changed to 2, which is a valid index for the geometry. These calls have been removed.

GTE/Mathematics/ConstrainedDelaunay2.h

The function GetBoundaryPolygons had a call edgePairs.erase(vStart) that removed all edge pairs involving vStart, which is incorrect behavior. A block of code was added to remove only the edge from vStart to vNext.

GTE/Mathematics/ETManifoldMesh.h

September 1, 2020. The convex hull classes ConvexHull2 and ConvexHull3 performed all geometric queries using rational arithmetic when the ComputeType is an arbitrary precision class. I replaced the queries with ones that use floating-point interval arithmetic to determine signs of expressions, falling back to rational arithmetic only when the interval arithmetic cannot determine the exact sign. The ComputeType need no longer be specified. The class determines the number of bits of precision required to compute the hull. I will be making similar changes in other computational geometry code, replacing PrimalQuery2 and PrimalQuery3 by classes that use interval arithmetic with fallback to rational arithmetic.

GTE/Mathematics/ConvexHull2.h
GTE/Mathematics/ConvexHull3.h
GTE/Mathematics/SeparatePoints2.h
GTE/Mathematics/SeparatePoints3.h
GTE/Mathematics/MinimumAreaBox2.h
GTE/Mathematics/MinimumVolumeBox3.h
GTE/Samples/Geometrics/ConvexHull2D/ConvexHull2DWindow2.{h,cpp}
GTE/Samples/Geometrics/ConvexHull3D/ConvexHull2DWindow3.cpp
GTE/Samples/Geometrics/ExtremalQuery/ExtremalQueryWindow3.cpp

August 22, 2020. Fixed a bug in the operator* member function of BSPrecision. The word counting for BSRational just copied the results from that of BSNumber. Instead, the computations for BSRational use the same algorithm as that of BSNumber, but copying does not do that.

GTE/Mathematics/BSPrecision.h

August 14, 2020. Microsoft Visual Studio 2019 now incorporates a new C++ Standard Library, *Apache-2.0 WITH LLVM-exception*; previous versions used the library provided by Dinkumware. The comparison operators of FeatureKey used the std::array comparisons. Profiling code that includes ETManifoldMesh, which uses EdgeKey and TriangleKey in std::map objects showed that these operators are much slower in Debug builds than the Dinkumware version. I replaced the comparisons with simple ones to remove the slowdown.

GTE/Mathematics/FeatureKey.h

Sadly, running applications that use my rational arithmetic library with classes BSNumber and BSRational also have significant slowdowns in Debug builds. I am considering rolling reduced versions of std::array and std::vector that are included only in Debug builds.

August 14, 2020. Restored the Timer class, which is a simple and convenient high-resolution timer for applications.

GTE/Mathematics/Timer.h

29 Updates to Version 4.7

August 11, 2020. Removed the tools projects from the build-all solutions. These can be built separately when needed.

GTE/BuildAll.*.sln
GTE/BuildAllDX11.*.sln
GTE/BuildAllGL45.*.sln

Added an implementation of the Remez algorithm for minimax polynomial approximations to functions. Added robust and fast approximations to rotation matrices; see RotationEstimate.h. A tool was added that generates the minimax polynomial coefficients for the rotation matrix approximations.

GTE/Mathematics.{v14,v15,v16}.{vcxproj,vcxproj.filters}
GTE/Mathematics/RemezAlgorithm.h
GTE/Mathematics/RotationEstiate.h
GTE/Tools/RotationApproximation/*

July 21, 2020. The SetSign function needed to update the mValue (debugging support).

GTE/Mathematics/BSRational.h

July 20, 2020. The classes Matrix2x2 and Matrix3x3 are not used by the header files, so the corresponding include statements were replaced to get access to Vector2 and Vector3.

GTE/Mathematics/ApprHeightLine2.h
GTE/Mathematics/ApprHeightPlane3.h

30 Updates to Version 4.6

June 6, 2020. Refactored WICFileIO, adding a new class WICFileIONative that allows load/save operations without having to use any GTE class. Class WICFileIO is now an example of how to load/save Texture2 objects using only the services from WICFileIONative. I did not provide a Linux version of WICFileIONative but will do so at a later date.

GTE/GTApplications*.{v14,v16,v16}.vcproj*
GTE/Applications/WICFileIO.h
GTE/Applications/MSW/WICFileIO.{h,cpp}
GTE/Applications/MSW/WICFileIONative.{h,cpp}

June 2, 2020. Added a templated Set function, similar to the already present Get functions, for convenience of the user not to have to constantly add reinterpret_cast of pointers passed to SetData.

GTE/Graphics/Resource.h

May 30, 2020. The code did not compile when using Real set to BSRational<UIntegerAP32> because of the presence of some integer quantities in the arithmetic equations for the coefficients of the h-polynomial. I added typecasts of those to Real.

GTE/Mathematics/DistLine3Circle3.h

May 28, 2020. Added a test in CopyGpuToGpu to determine whether the entire resource is to be copied or only a subresource. The old code always called CopySubresource. The new code will call CopyResource when appropriate.

GTE/Graphics/DX11/DX11Buffer.cpp

May 3, 2020. The code blocks with the point-triangle distance queries were not copying the barycentric coordinates from the point-triangle query result to the segment-triangle query result. The last block where the segment endpoint for the negative extent is closest to the triangle had set the segment-triangle query result member result.segmentParameter to the positive extent, but it should have been the negative extent.

GTE/Mathematics/DistSegment3Triangle3.h

April 23, 2020. Added another Rotate function to transform 3-tuple vectors by a quaternion. The old and the new functions now also have conditional compilation based on whether or not GTE_USE_MAT_VEC is defined. The default for all projects is that this preprocessor symbol is defined.

GTE/Mathematics/Quaternion.h

April 16, 2020. Fixed a bug in the Convert function that converts one BSNumber to another one with a specified precision. The bug arose when the output number did not satisfy the invariant that the nonzero UInteger part is a positive odd number. A shift-right operation needed to be applied. The unit tests were updated with a case related to this. A new function IsValid has been added to BSNumber that, when exposed via a conditional define, verifies any constructured BSNumber satisfies the invariant.

GTE/Mathematics/BSNumber.h

31 Updates to Version 4.5

April 13, 2020. Rewrote the class UniqueVerticesTriangles. The class allows you to convert triangle soup to indexed triangles, removing duplicate vertices from the soup. It was later extended to remove duplicate vertices from indexed triangles. The revision adds the ability to remove unused vertices as well as duplicate vertices, which is desirable in applications wanting a compact representation of meshes.

GTE/Mathematics/UniqueVerticesTriangles.h
GTE/Mathematics/IntrConvexMesh3Plane3.h
GTE/Mathematics/SurfaceExtractorMC.h

Factored out the enumeration of IDXGIOutput objects for an adapter to a public static function DXGIOutput::Enumerate. This avoids repeated enumeration each time a DXGIAdapter constructor is called. The GTEngine code never uses these objects, but if your application needs them, you can access the adapter by DX11Engine::GetAdapter and then call the static function DXGIOutput::Enumerate.

GTE/Graphics/DX11/DXGIAdapter.cpp
GTE/Graphics/DX11/DXGIOutput.{h,cpp}

Added a new template function to DX11.h named SafeAddRef. Replaced the COM AddRef and Release calls by those to DX11::SafeAddRef and DX11::SafeRelease, even if it is known the pointers are not null. This supports debugging reference counting by allowing additional code to be added in one place (the DX11 class) to log the counts.

GTE/Graphics/DX11/DX11.h
GTE/Graphics/DX11/DXGIAdapter.cpp
GTE/Graphics/DX11/DXGIOutput.cpp
GTE/Graphics/DX11/DX11Engine.cpp
GTE/Graphics/DX11/DX11GeometryShader.cpp
GTE/Graphics/DX11/DX11GraphicsObject.cpp
GTE/Graphics/DX11/DX11Texture1.cpp
GTE/Graphics/DX11/DX11Texture1Array.cpp
GTE/Graphics/DX11/DX11Texture2.cpp
GTE/Graphics/DX11/DX11Texture2Array.cpp
GTE/Graphics/DX11/DX11Texture3.cpp
GTE/Graphics/DX11/DX11TextureCube.cpp
GTE/Graphics/DX11/DX11TextureCubeArray.cpp
GTE/Graphics/DX11/DX11TextureRT.cpp
GTE/Graphics/DX11/DX11VertexShader.cpp
GTE/Graphics/DX11/HLSLShaderFactory.cpp

April 12, 2020. Fixed a COM reference counting problem with the DXGI adapters and outputs.

GTE/Graphics/DX11/DXGIAdapter.cpp
GTE/Graphics/DX11/DXGIOutput.cpp
GTE/Graphics/DX11/DX11Engine.{h,cpp}

Added a template GetNumReferences for reading the reference count from a COM object.

GTE/Graphics/DX11/DX11.h

Revised the logic for creation of the DX11Engine object. If the application sets parameters.deviceCreationFlags to D3D11_CREATE_DEVICE_DEBUG, the DX11 debug layer will crash when the adapter is specified as a non-null pointer (obtained via DXGIAdapter::Enumerate). The debug layer will work only when a null adapter pointer is passed to D3D11CreateDevice in addition to the flags parameter set to D3D11_CREATE_DEVICE_DEBUG. If you want the debug layer, you cannot simultaneously select the GPU adapter.

GTE/Applications/MSW/WindowSystem.cpp

April 8, 2020. Added new files to support exact-arithmetic find-intersection queries for planes and convex polyhedra.

GTE/GTMathematics.{v14,v15,v15}.{sln,vcxproj,vcxproj.filters}
GTE/Mathematics/ConvexMesh3.h
GTE/Mathematics/IntrConvexMesh3Plane3.h
GTE/Samples/Intersection/IntersectPlaneConvexPolyhedron/IntersectPlaneConvexPolyhedron*.*

Added constructors to handle triangles stored as std::vector of std::array<int, 3>. The original constructors take std::vector of int, where the precondition is that the number of indices is a multiple of 3.

GTE/Mathematics/UniqueVerticesTriangles.h

April 4, 2020. Added noexcept to the move constructors and the move operators.

GTE/Mathematics/UIntegerAP32.h
GTE/Mathematics/UIntegerFP32.h
GTE/Mathematics/BSNumber.h
GTE/Mathematics/BSRational.h

Removed unnecessary gte namespace scoper.

GTE/Mathematics/APInterval.h

March 25, 2020. Modified the Font constructor to be protected because this class is supposed to be an abstract base class.

GTE/Graphics/Font.h

March 24, 2020. Added a function GetMostPowerful that enumerates the available GPU adapters and selects a discrete GPU if one exists or Intel Integrated Graphics if it exists. With current hardware, in the unlikely event that Intel Integrated Graphics does not exist, Microsoft WARP is selected.

GTE/Graphics/DX11/DXGIAdapter.{h,cpp}
GTE/Applications/MSW/WindowSystem.cpp

Added a function to copy the back buffer to a texture. For now this is available only for the DirectX 11 engine. I will add support later for the OpenGL 4.5 engine.

GTE/Graphics/DX11/DX11Engine.{h,cpp}

March 15, 2020. Modified the creation of the DX11Engine object for the application library. The old code passed a null adapter to allow the DXGI system to select the GPU. If an enumeration of adapters were to be performed first, DXGI selects the first adapter in the enumeration. On a Microsoft Surface Book with Intel Integrated Graphics and a discrete NVIDIA graphics chip, the Intel graphics shows up first. The new code will select a discrete adapter if available. If such an adapter is not on the system, it will then select Intel graphics. If this does not exist (and the computer is a dinosaur), Microsoft WARP is chosen.

GTE/Applications/MSW/WindowSystem.cpp

March 13, 2020. Added new function GetBoundaryPolygons that computes the connected components of boundary edges for the mesh. The comments for that function indicate how branch points of the vertex adjacency graph of the boundary edges are handled.

GTE/Mathematics/ETManifoldMesh.h

March 9, 2020. Added new functions CreateCompactGraph, GetComponentsConsistentChirality and MakeConsistentChirality that support updating the mesh so that each connected component has triangles of the same winding order.

GTE/Mathematics/ETManifoldMesh.h

March 8, 2020. Added FeatureKey, EdgeKey, TriangleKey and TetrahedronKey custom visualizers.

GTE/gtengine.vis

February 25, 2020. Apparently, the creation of a rectangle mesh using the Mesh and RectangleMesh classes was a work in progress that was accidentally checked in. The construction of the mesh was incorrect. Restored the old code.

GTE/Graphics/MeshFactory.cpp

February 21, 2020. Added a LogWarning call in the last block of function Sub. This case cannot occur by design, but it in fact did occur during experiments for rounding algorithms of arbitrary-precision numbers (while writing the RAEFGC book). I had failed to write down the example of failure and did not analyze the call stack to see how this could have happened. If this block is reached, please report it; the BSNumber inputs to operator+ or operator- need to be reported as part of the bug report. In the event this case does occur, the code returns the correct answer of zero.

GTE/Mathematics/UIntegerALU32.h

February 17, 2020. Fixed the last comment in operator< to "One or both numbers are zero."

GTE/Mathematics/UIntegerALU32.h

February 12, 2020. Unit tests of IEEEBinary16 in GTL exposed a couple of problems. The conversion of a 32-bit normal floating-point number to a 16-bit subnormal floating-point number was incorrect. The std::asinh(IEEEBinary16) wrapper function called asin instead of asinh.

GTE/Mathematics/IEEEBinary16.h

Modified code to match that of GTL. The Classification enumeration is now enum class Classification to conform to C++ core suggestions by Visual Studio. Added some clarifying comments. Implemented the TODO blocks regarding NaNs.

GTE/Mathematics/IEEEBinary.h

February 5, 2020. Added bisection algorithms that use floating-point or arbitrary-precision arithmetic, the latter with a user-specified precision.

GTE/GTMathematics.{v14,v15,v16}.{vcxproj,vcxproj.filters}
GTE/Mathematics/RootsBisection1.h
GTE/Mathematics/RootsBisection2.h

February 1, 2020. The wrong constructor was called when USE_COVARIANCE_W_DIRECTION was enabled. Modified the code to use the correct constructor.

GTE/Samples/Mathematics/FitCylinder/FitCylinderWindow3.cpp

Added debug blocks of code to places where the BSNumber is modified and the new debug member mValue needs to be updated.

GTE/Mathematics/BSNumber.h

The conversion from BSRational to BSRational of a specified precision had a bug that the unit tests did not catch. The unit tests all involved rational numbers that had repeating binary patterns. For a number that has a finite bit pattern, the code needs to detect this and properly format the bit pattern for output.

GTE/Mathematics/BSRational.h

Janaury 13, 2020. Fixed comments in the file (removed the Gte prefix from the file names).

GTE/Mathematics/GaussNewtonMinimizer.h
GTE/Mathematics/LevenbergMarquardtMinimizer.h

32 Updates to Version 4.4

January 12, 2020. The main reason for the update is to support Intel C++ compilers with version 17, 18 and 19.

January 11, 2020. Added the ported tools GenerateApproximations, GenerateOpenGLWrapper and PrecisionCalculator to the build-all solutions.

GTE/BuildAll.{v14,v15,v16}.sln
GTE/BuildAllDX11.{v14,v15,v16}.sln
GTE/BuildAllGL45.{v14,v15,v16}.sln

Ported the GenerateOpenGLWrapper tool from GTEngine 3 to GTEngine 4. The GTEngine 3 version has some manually modified lines of code in GL45.cpp. The tool was updated to generate those lines automatically.

GTE/Tools/GenerateOpenGLWrapper/GenerateOpenGLWrapper.{v14,v15,v16}.{vcxproj,vcxproj.filters}
GTE/Tools/GenerateOpenGLWrapper/GenerateOpenGLWrapper.cpp
GTE/Tools/GenerateOpenGLWrapper/GL45.h
GTE/Tools/GenerateOpenGLWrapper/Initialize.txt
GTE/Tools/GenerateOpenGLWrapper/Version.txt
GTE/Graphics/GL45/GL45.{h,cpp}

Ported the GenerateApproximations tool from GTEngine 3 to GTEngine 4.

GTE/Tools/GenerateApproximations/GenerateApproximations.{v14,v15,v16}.{vcxproj,sln}
GTE/Tools/GenerateApproximations/FitASin.h
GTE/Tools/GenerateApproximations/FitATan.h
GTE/Tools/GenerateApproximations/FitCos.h
GTE/Tools/GenerateApproximations/FitExp2.h
GTE/Tools/GenerateApproximations/FitInvSqrt.h
GTE/Tools/GenerateApproximations/FitLog2.h
GTE/Tools/GenerateApproximations/FitReciprocal.h
GTE/Tools/GenerateApproximations/FitSin.h
GTE/Tools/GenerateApproximations/FitSqrt.h
GTE/Tools/GenerateApproximations/FitTan.h
GTE/Tools/GenerateApproximations/GenerateApproximations.cpp

Ported the PrecisionCalculator tool from GTEngine 3 to GTEngine 4.

GTE/Tools/PrecisionCalculator/PrecisionCalculator.{v14,v15,v16}.{vcxproj,vcxproj.filters,sln}
GTE/Tools/GenerateOpenGLWrapper/PrecisionCalculator.cpp

At some point in time the inputs to some member functions were int but were later changed to uint32_t. This makes inequality comparisons to zero irrelevant.

GTE/Graphics/IndexBuffer.cpp
GTE/Graphics/Particles.cpp

The Intel Compiler 17 complained that Codeint wmain(int, wchar_t*[]) is an incorrect signature, which it is not. The tool does not use command-line parameters, so the signature was modified to int wmain().

GTE/Tools/BitmapFontCreator/BitmapFontCreator.cpp

The Intel Compiler 17 generated errors for CopyGpuToGpu(parameters), claiming these hide the virtual function CopyGpuToGpu(). The Intel Compiler 19 did not have this problem (nor does Microsoft Visual Studio or g++). These functions are not implemented yet, so I renamed them to CopyLevelGpuToGpu(parameters) for now.

GTE/Graphics/GL45/GL45Texture.h
GTE/Graphics/GL45/GL45TextureArray.h

The Intel Compiler 17 generated errors for OnUpdate(parameters), claiming these hide the virtual function OnUpdate(). The Intel Compiler 19 did not have this problem (nor does Microsoft Visual Studio or g++). The update function with parameters is protected, so I renamed them to OnUpdateSingle(parameters).

GTE/Mathematics/PdeFilter2.h
GTE/Mathematics/CurvatureFlow2.h
GTE/Mathematics/GaussianBlur2.h
GTE/Mathematics/GradientAnisotropic2.h
GTE/Mathematics/PdeFilter3.h
GTE/Mathematics/CurvatureFlow3.h
GTE/Mathematics/GaussianBlur3.h
GTE/Mathematics/GradientAnisotropic3.h

The Plane3 constructor calls involved a triple of points, which means the constructor with the std::array input is used. The code was written as Plane3<Real>{p0,p1,p1}, but the Intel Compiler 17 complained. The code was modified to Plane3<Real>({p0,p1,p2}). It is not clear why all the other compilers allowed the original code, which looks like an initializer-list constructor call.

GTE/Mathematics/SeparatingPoints3.h

The Intel Compiler 17 warned about hiding the base-class members for the Parameters constructor, where the inputs used the same names as the base class. The input names were modified.

GTE/Samples/Imagics/GpuGaussianBlur2Window2.h
GTE/Samples/Imagics/GpuGaussianBlur3Window2.h

January 8, 2020. The Intel compilers complained about the signature to main in all the samples. I had int main(int, char const*[]) but the standard format is int main(int, char*[]) or int main(int, char**). I modified all such main calls to have no parameters, which is a long list of files not mentioned here. A couple of the samples use command-line parameters, so the main functions for those were modified to use char*[].

The initializer-list constructors for classes were not handled properly by MSVS 2013. For example, to create a Vector2<Real> object using an initializer list, the syntax is Vector2<Real>{x, y}. However, MSVS 2013 did not allow this, requiring instead the syntax Vector2<Real>({x, y}). The Intel compiler had problems with some occurrences of the latter. They also has problems with some assignment statements that led to construction via initializer lists. Now that MSVS 2013 is no longer supported, syntax without parentheses is now used. While modifying these, some calls to BoundingSphere::SetCenter were found that passed an initializer list of 4 elements, because the input of that function used to be Vector4. Now the input is Vector3, which has an initializer-list constructor that allows more than 3 elements (the ones after the first three are ignored). The SetCenter calls were modified to have initializer lists of 3 elements.

GTE/Mathematics/AdaptiveSkeletonClimbing3.h
GTE/Graphics/CullingPlane.h
GTE/Mathematics/Ellipse3.h
GTE/Mathematics/QFNumber.h
GTE/Mathematics/SeparatePoints3.h
GTE/Mathematics/Transform.h
GTE/Mathematics/TriangulateCDT.h
GTE/Mathematics/TriangulateEC.h
GTE/Mathematics/Vector2.h

GTE/Mathematics/Vector3.h
GTE/Graphics/SamplerState.cpp
GTE/Graphics/BlendState.cpp
GTE/Graphics/MeshFactory.cpp
GTE/Graphics/BspNode.cpp
GTE/Graphics/PickRecord.cpp
GTE/Graphics/Node.cpp
GTE/Graphics/Material.cpp
GTE/Graphics/Lighting.cpp
GTE/Graphics/LightCameraGeometry.cpp
GTE/Samples/Distance/DistanceAlignedBoxes/DistanceAlignedBoxesWindow3.cpp
GTE/Samples/Distance/DistanceAlignedBoxOrientedBox/DistanceAlignedBoxOrientedBoxWindow3.cpp
GTE/Samples/Distance/DistanceOrientedBoxes/DistanceOrientedBoxesWindow3.cpp
GTE/Samples/Distance/DistancePointConvexPolyhedron/DistancePointConvexPolyhedronWindow3.cpp
GTE/Samples/Distance/DistanceRectangleBox/DistanceRectangleBoxWindow3.cpp
GTE/Samples/Distance/DistanceTriangleBox/DistanceTriangleBoxWindow3.cpp
GTE/Samples/Geometrics/Delaunay3D/Delaunay3DWindow3.cpp
GTE/Samples/Intersection/IntersectBoxCone/IntersectBoxConeWindow3.cpp
GTE/Samples/Intersection/IntersectBoxCylinder/IntersectBoxCylinderWindow3.cpp
GTE/Samples/Intersection/IntersectBoxSphere/IntersectBoxSphereWindow3.cpp
GTE/Samples/Intersection/IntersectSphereCone/IntersectSphereConeWindow3.cpp
GTE/Samples/Physics/HelixTubeSurface/HelixTubeSurfaceWindow3.cpp
GTE/Samples/SceneGraphs/BlendedAnimations/BlendedAnimationsWindow3.cpp
GTE/Samples/SceneGraphs/Picking/PickingWindow3.cpp

The inputs to some constructor calls used names number and encoding, which are the union-member names of the base class. The Intel compilers warned about this, so the constructor input names were modified.

GTE/Mathematics/IEEEBinary16.h

The SetMember and GetMember functions are supposed to return Boolean values, but the last blocks of the implementations were missing the return statements.

GTE/Mathematics/TextureBuffer.h

Intel Compiler 17 does not support the template alias helper remove_cv_t, so I replaced it with something simpler in my arbitrary-precision code.

GTE/Mathematics/Math.h

33 Updates to Version 4.3

January 4, 2020. Ported the managed code projects for GTEngine 4.x.

GTE/Samples/CSharpManaged_MSVS2019/CppLibrary/CppLibrary.v16.vcxproj
GTE/Samples/CSharpManaged_MSVS2019/CppLibrary/CppLibrary.v16.vcxproj.filters
GTE/Samples/CSharpManaged_MSVS2019/CppLibrary/CppLibrary.h
GTE/Samples/CSharpManaged_MSVS2019/CppLibrary/MinimumVolumeBox.h
GTE/Samples/CSharpManaged_MSVS2019/CppLibrary/MinimumVolumeBox.cpp
GTE/Samples/CSharpManaged_MSVS2019/CSharpApplication/CSharpApplication.v16.sln
GTE/Samples/CSharpManaged_MSVS2019/CSharpApplication/CSharpApplication.v16.csproj
GTE/Samples/CSharpManaged_MSVS2019/CSharpApplication/Program.cs
GTE/Samples/CSharpManaged_MSVS2019/CSharpApplication/App.config
GTE/Samples/CSharpManaged_MSVS2019/CSharpApplication/Properties/AssemblyInfo.cs
GTE/Samples/CSharpManaged_MSVS2019/ManagedLibrary/ManagedLibrary.v16.sln
GTE/Samples/CSharpManaged_MSVS2019/ManagedLibrary/ManagedLibrary.v16.vcxproj
GTE/Samples/CSharpManaged_MSVS2019/ManagedLibrary/ManagedLibrary.v16.vcxproj.filters

GTE/Samples/CSharpManaged_MSVS2019/ManagedLibrary/ManagedObject.h
GTE/Samples/CSharpManaged_MSVS2019/ManagedLibrary/MinimumVolumeBox.h
GTE/Samples/CSharpManaged_MSVS2019/ManagedLibrary/MinimumVolumeBox.cpp
GTE/Samples/CSharpManaged_MSVS2019/ManagedLibrary/AssemblyInfo.cpp

GTE/Samples/CSharpManaged_MSVS2017/CppLibrary/CppLibrary.v15.vcxproj
GTE/Samples/CSharpManaged_MSVS2017/CppLibrary/CppLibrary.v15.vcxproj.filters
GTE/Samples/CSharpManaged_MSVS2017/CppLibrary/CppLibrary.h
GTE/Samples/CSharpManaged_MSVS2017/CppLibrary/MinimumVolumeBox.h
GTE/Samples/CSharpManaged_MSVS2017/CppLibrary/MinimumVolumeBox.cpp
GTE/Samples/CSharpManaged_MSVS2017/CSharpApplication/CSharpApplication.v15.sln
GTE/Samples/CSharpManaged_MSVS2017/CSharpApplication/CSharpApplication.v15.csproj
GTE/Samples/CSharpManaged_MSVS2017/CSharpApplication/Program.cs
GTE/Samples/CSharpManaged_MSVS2017/CSharpApplication/App.config
GTE/Samples/CSharpManaged_MSVS2017/CSharpApplication/Properties/AssemblyInfo.cs
GTE/Samples/CSharpManaged_MSVS2017/ManagedLibrary/ManagedLibrary.v15.sln
GTE/Samples/CSharpManaged_MSVS2017/ManagedLibrary/ManagedLibrary.v15.vcxproj
GTE/Samples/CSharpManaged_MSVS2017/ManagedLibrary/ManagedLibrary.v15.vcxproj.filters
GTE/Samples/CSharpManaged_MSVS2017/ManagedLibrary/ManagedObject.h
GTE/Samples/CSharpManaged_MSVS2017/ManagedLibrary/MinimumVolumeBox.h
GTE/Samples/CSharpManaged_MSVS2017/ManagedLibrary/MinimumVolumeBox.cpp
GTE/Samples/CSharpManaged_MSVS2017/ManagedLibrary/AssemblyInfo.cpp

GTE/Samples/CSharpManaged_MSVS2015/CppLibrary/CppLibrary.v14.vcxproj
GTE/Samples/CSharpManaged_MSVS2015/CppLibrary/CppLibrary.v14.vcxproj.filters
GTE/Samples/CSharpManaged_MSVS2015/CppLibrary/CppLibrary.h
GTE/Samples/CSharpManaged_MSVS2015/CppLibrary/MinimumVolumeBox.h
GTE/Samples/CSharpManaged_MSVS2015/CppLibrary/MinimumVolumeBox.cpp
GTE/Samples/CSharpManaged_MSVS2015/CSharpApplication/CSharpApplication.v14.sln
GTE/Samples/CSharpManaged_MSVS2015/CSharpApplication/CSharpApplication.v14.csproj
GTE/Samples/CSharpManaged_MSVS2015/CSharpApplication/Program.cs
GTE/Samples/CSharpManaged_MSVS2015/CSharpApplication/App.config
GTE/Samples/CSharpManaged_MSVS2015/CSharpApplication/Properties/AssemblyInfo.cs
GTE/Samples/CSharpManaged_MSVS2015/ManagedLibrary/ManagedLibrary.v14.sln
GTE/Samples/CSharpManaged_MSVS2015/ManagedLibrary/ManagedLibrary.v14.vcxproj
GTE/Samples/CSharpManaged_MSVS2015/ManagedLibrary/ManagedLibrary.v14.vcxproj.filters
GTE/Samples/CSharpManaged_MSVS2015/ManagedLibrary/ManagedObject.h
GTE/Samples/CSharpManaged_MSVS2015/ManagedLibrary/MinimumVolumeBox.h
GTE/Samples/CSharpManaged_MSVS2015/ManagedLibrary/MinimumVolumeBox.cpp
GTE/Samples/CSharpManaged_MSVS2015/ManagedLibrary/AssemblyInfo.cpp

34 Updates to Version 4.2

December 28, 2019. Replaced the std::memset calls by std::fill calls.

GTE/Mathematics/NURBSSurface.h
GTE/Mathematics/NURBSVolume.h

December 27, 2019. Added a least-squares algorithm for fitting two parallel lines to points in the plane, where the points presumably lie on the lines. A document *Least Squares Fitting of Parallel Lines to Points in 2D* has been uploaded to the website to describe the algorithm.

GTE/Mathematics/ApprParallelLines2.h

The multiplication and division by the natural logarithm of 2 were swapped in the exponent and logarithm estimation files.

GTE/Mathematics/ExpEstimate.h
GTE/Mathematics/LogEstimate.h

35 Updates to Version 4.1

December 5, 2019. The minor version was modified to 2 (for GTE 4.2). Fixed the comment in the post-build soft-link setting.

GTE/makeapplications.gte
GTE/makegraphics.gte
GTE/makemathematicsgpu.gte

Removed a call rnd(mte) that was not used.

GTE/Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow2.cpp

Added an include of <algorithm> because the class accesses std::max (compiler error occurs now with MSVS 2019 16.4.0).

GTE/Mathematics/RootsPolynomial.h

Fixed a bug in the Evaluate function.

GTE/Mathematics/ApprPolynomial4.h

The GTGraphicsGL45 library had a dependency on the GTApplicationsGL45 library via the class GLXEngine when used for compute shaders; the constructor has to create an X-window that is not exposed to the caller. I removed that dependency.

GTE/Graphics/GL45/GLX/GLXEngine.cpp

Removed an include of Applications/Environment.h that caused the Graphics library to have a dependency on the GTApplications* libraries. The AreaLightEffect does not use an Environment object.

GTE/Graphics/AreaLightEffect.h

December 4, 2019. Fixed out-of-range indices in the GLSL reflection code. Modified the reflection functions to use enumerations for indexing rather than raw pointer incrementing.

GTE/Graphics/GL45/GLSLReflection.cpp

November 23, 2019. Replaced the std::fabs call with a sign test to allow using Gaussian elimination with QFNumber without yet having to implement std::fabs for QFNumber

GTE/Mathematics/GaussianElimination.h

November 22, 2019. Added functions GetTokens, GetTextTokens and GetAdvancedTextTokens as convenience functions for extracting tokens from a string.

GTE/Mathematics/StringUtility.h

November 4, 2019. The design of QFNumber requires explicitly passing the d-variable to constructors. This is problematic in implementations using generic template code for a Real type, because d is not available internally. The class LCPSolver uses generic 0 and 1 in copies and assignments. Constructors were added to allow passing the values of 0 and 1 for the specific d-value of the quadratic field numbers involved in the computations.

GTE/Mathematics/LCPSolver.h

November 3, 2019. Reformatted some comments and added more specific ideas about representations for infinities and NaNs of BSNumber.

GTE/Mathematics/BSNumber.h GTE/Mathematics/BSRational.h

October 23, 2019. Modified the Result nested class to provide more information about the returned result. This includes an LCPSolverShared<Real>::Result member rather than the previous bool queryIsSuccessful member.

GTE/Mathematics/IntrAlignedBox3Cylinder3.h

October 17, 2019. Fixed some comments. Removed the Gte file prefix in the ToTetrahedron calls. Fixed the counts for N for BSRational; they are much smaller than previously reported (incorrect counting in BSPrecision).

GTE/Mathematics/PrimalQuery3.h

Added std::enable_if to use division operations only if the arithmetic system allows it.

GTE/Mathematics/APInterval.h

October 9, 2019. Renamed class Interval and file Interval.h to FPInterval and FPInterval.h to emphasize these apply to native floating-point types float and double. Added class APInterval and file APInterval.h for interval arithmetic using arbitrary-precision types. These types do not need the rounding modes set as the floating-point types do.

GTE/Mathematics/Interval.h
GTE/Mathematics/FPInterval.h
GTE/Mathematics/APInterval.h
GTE/GTMathematics.*.vcxproj
GTE/GTMathematics.*.vcxproj.filters

Added support for conversion from text-string numbers to BSNumber and BSRational objects. This supports inputs of precision larger than native integer and floating-point types.

GTE/Mathematics/BSNumber.h
GTE/Mathematics/BSRational.h

September 27, 2019. Changed the member names from GetMin and GetMax to the more descriptive GetMinOfSqrt and GetMaxOfSqrt. Removed the minimum-closest parameter of the estimators. Added member functions to return a single estimate rather than a bounding interval.

GTE/Mathematics/APConversion.h

September 26, 2019. Fixed a bug in UIntegerALU32::Sub. The last statement that sets the number of bits after locating the leading 1-bit to allow reducing the size of the bits array fails when self has no 1-bits (the block value is -1, which causes an out-of-range lookup). Added a test for this case and set the number of bits explicitly to zero. Modified the SetNumBits functions in UIntegerAP32 and UIntegerFP32 to test whether the input is nonnegative and adjust accordingly.

GTE/Mathematics/UIntegerALU32.h
GTE/Mathematics/UIntegerAP32.h
GTE/Mathematics/UIntegerFP32.h

Added helper conversion functions that are used in the APPrecision class to avoid exposing BSNumber objects. One conversion computes a BSRational to a specified precision from another BSRational. The other conversion computes a float or double from a BSRational; the precision is that of the output floating-point type.

GTE/Mathematics/BSRational.h

Modified EstimateSqrt and EstimateApB to generate iterates at twice the precision specified by the user. This avoids the quadratic growth in number of bits that causes significant performance loss during Newton's method. Modified EstimateAmB to use a better algorithm for estimating the linear combination of two square roots by a BSRational.

GTE/Mathematics/APConversion.h

September 25, 2019. Updated the code to handle both BSNumber and BSRational simultaneously. The formulas were incorrect for bit counting, but fortunately they overcounted. The implementation was fixed and the construction of the formulas is described in the Geometric Computing book.

GTE/Mathematics/BSPrecision.h

Changed typedef statements to using statements.

GTE/Mathematics/IEEEBinary.h

Cleaned up white space and modified some comments. Removed the bodies of four of the comparisons, replacing them by the logical equivalences to equality and less-than comparison.

GTE/Mathematics/QFNumber.h

September 11, 2019. Removed the friend declarations for UnitTestBSNumber because now the unit tests have access to the private members via public member accessors.

GTE/Mathematics/UIntegerAP32.h
GTE/Mathematics/UIntegerFP32.h

September 10, 2019. Added an implementation of interval arithmetic.

GTE/GTMathematics.*.vcxproj
GTE/GTMathematics.*.vcxproj.filters
GTE/Mathematics/Interval.h

September 5, 2019. Renamed the template parameter UIntegerType to UInteger for consistency of naming. The geometry book uses the latter name for the type.

GTE/Mathematics/BSNumber.h
GTE/Mathematics/BSRational.h
GTE/Mathematics/UIntegerAP32.h

September 4, 2019. Replaced the Convert function with a new version. The old version was specifically for float and double. The new version allows you to specify the precision of a conversion to BSNumber, which can then be used for conversions to float and double.

GTE/Mathematics/BSRational.h

Added a Convert function that allows you to specify the precision of a conversion from a BSNumber to another BSNumber. Added a non-const GetUInteger function.

GTE/Mathematics/BSNumber.h

September 3, 2019. Added RoundUp function to support conversion from BSRational to a BSNumber of a specified precision.

GTE/Mathematics/UIntegerALU32.h

The GetMaxSize functions are now static because they do not depend on a particular object.

GTE/Mathematics/UIntegerAP32.h
GTE/Mathematics/UIntegerFP32.h

Added two more interface functions in the comments about UIntegerType.

GTE/Mathematics/BSNumber.h

Fixed comment where the multiplication operator occurred instead of the power operator.

GTE/Mathematics/BSRational.h

The constructors with the int32_t parameter and the function SetNumBits are intended to be used only internally (in the class UIntegerALU32). However, the constructors are public and there is a chance of a mismatch if a signed integer is passed to a constructor rather than an unsigned integer. To avoid this, the constructor with the int32_t parameter was removed. Also, the SetNumBits function has an input whose type needed to be int32_t. Finally, added a new function GetMaxSize that returns the maximum 32-bit integer for UIntegerAP32 and returns N for UIntegerFP32.

GTE/Mathematics/UIntegerALU32.h
GTE/Mathematics/UIntegerAP32.h
GTE/Mathematics/UIntegerFP32.h

Added functions to set all bits (up through maximum size) to zero.

GTE/Mathematics/UIntegerAP32.h
GTE/Mathematics/UIntegerFP32.h

Added an include of <cfenv> to allow access to the floating-point environment.

GTE/Mathematics/Math.h

36 Updates to Version 4.0

August 29, 2019. I introduced a bug when HeightInRange was added to the Cone class. The comparison needed to be for h^2, not h.

GTE/Mathematics/ContCone.h

August 26, 2019. The classes were missing some public-internal member functions and std::frexp that had been added to the GTEngine 3.x classes. Moved the public-internal member functions to public section with the other accessors.

GTE/Mathematics/BSNumber.h
GTE/Mathematics/BSRational.h

August 16, 2019. Added the ability to specify sorting of eigenvalues and eigenvectors in the noniterative solver NISymmetricEigensolver3x3. The sorting code was factored out of the iterative solver SymmetricEigensolver3x3 so it can be shared by both classes.

GTE/Mathematics/SymmetricEigensolver3x3.h

August 15, 2019. The relative paths to the libraries were hard-coded in the strings for the solution files. They needed to be the pattern _GT4_RELATIVE_PATH_. Added post-build copies of the executables to the appropriate subfolders of GTE/Executables.

GTE/Tools/GenerateProject/ProjectTemplate.{v14,v15,v16}.cpp
GTE/Tools/GenerateProject/GenerateProject.{v14,v15,v16}.vcxproj

37 Updates to Version 3.30

January 9, 2020. Added the changes to the Result structure that was already made for the GTEngine 4 track.

IntrAlignedBox3Cylinder3.h

38 Updates to Version 3.29

December 28, 2019. Replaced the std::memset calls by std::fill calls.

NURBSSurface.h
NURBSVolume.h

December 27, 2019. Added a least-squares algorithm for fitting two parallel lines to points in the plane, where the points presumably lie on the lines. A document *Least Squares Fitting of Parallel Lines to Points in 2D* has been uploaded to the website to describe the algorithm.

ApprParallelLines2.h

The multiplication and division by the natural logarithm of 2 were swapped in the exponent and logarithm estimation files.

ExpEstimate.h
LogEstimate.h

39 Updates to Version 3.28

December 5, 2019. The minor version was modified to 29 (for GTE 3.29). Fixed the comment in the post-build soft-link setting.

makeengine.gte
makeengine_werror.gte

Modified the GLXEngine constructor for compute shaders to access X11 directly rather than through TheWindowSystem. Although not relevant for GTEngine 3, it is relevant for GTE 4 where the graphics and application libraries have been separated.

GLXEngine.cpp

Removed a call rnd(mte) that was not used.

Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.cpp

Added an include of <algorithm> because the class accesses std::max (compiler error occurs now with MSVS 2019 16.4.0).

RootsPolynomial.h

Fixed a bug in the Evaluate function.

ApprPolynomial4.h

December 4, 2019. Fixed out-of-range indices in the GLSL reflection code. Modified the reflection functions to use enumerations for indexing rather than raw pointer incrementing.

GLSLReflection.cpp

Fixed a bug in UIntegerALU32::Sub. The last statement that sets the number of bits after locating the leading 1-bit to allow reducing the size of the bits array fails when self has no 1-bits (the block value is -1, which causes an out-of-range lookup). Added a test for this case and set the number of bits explicitly to zero. Modified the SetNumBits functions in UIntegerAP32 and UIntegerFP32 to test whether the input is nonnegative and adjust accordingly.

UIntegerALU32.h
UIntegerAP32.{h,cpp}
UIntegerFP32.h

The constructors with the int32_t parameter and the function SetNumBits are intended to be used only internally (in the class UIntegerALU32). However, the constructors are public and there is a chance of a mismatch if a signed integer is passed to a constructor rather than an unsigned integer. To avoid this, the constructor with the int32_t parameter was removed. Also, the SetNumBits function has an input whose type needed to be int32_t. Finally, added a new function GetMaxSize that returns the maximum 32-bit integer for UIntegerAP32 and returns N for UIntegerFP32.

UIntegerALU32.h
UIntegerAP32.{h,cpp}
UIntegerFP32.h

Added functions GetMaxSize. Added functions SetAllBitsToZero to set all bits (up through maximum size) to zero.

UIntegerAP32.h
UIntegerFP32.h

40 Updates to Version 3.27

August 29, 2019. I introduced a bug when HeightInRange was added to the Cone class. The comparison needed to be for h^2, not h.

ContCone.h

August 16, 2019. Added the ability to specify sorting of eigenvalues and eigenvectors in the noniterative solver NISymmetricEigensolver3x3. The sorting code was factored out of the iterative solver SymmetricEigensolver3x3 so it can be shared by both classes.

SymmetricEigensolver3x3.h

41 Updates to Version 3.26

August 8, 2019. A bug was introduced in the virtual Fit function that is used only in the RANSAC code. Also, the assignment operator used to copy the candidate model to the best-fit model did not work and required another virtual function CopyParameters, which had to be implemented in all the derived classes. The fixes were made and unit tests were added to verify RANSAC is working as designed.

ApprQuery.h
ApprGaussian2.h
ApprGaussian3.h
ApprHeightLine2.h
ApprHeightPlane3.h
ApprOrthogonalLine2.h
ApprOrthogonalLine3.h
ApprOrthogonalPlane3.h
ApprPolynomial2.h
ApprPolynomial3.h
ApprPolynomial4.h
ApprPolynomialSpecial2.h
ApprPolynomialSpecial3.h
ApprPolynomialSpecial4.h

42 Updates to Version 3.25

August 1, 2019. Added an include of the cstring header to access the definition for std::memcpy.

UnsymmetricEigenvalues.h

July 31, 2019. Replaced the C-style arrays by std::array-based members.

EulerAngles.h

Added two new member functions to class Font. Updated BitmapFontCreator and added new fonts to the graphics system.

GTEngine.*.vcxproj*
Font.{h,cpp}
Tools/BitmapFontCreator/BitMapFontCreator.cpp
GTGraphicsShared.h
FontArialW400H18.{h,cpp}
FontArialW400H12.{h,cpp}
FontArialW400H14.{h,cpp}
FontArialW400H16.{h,cpp}
FontArialW700H12.{h,cpp}
FontArialW700H14.{h,cpp}
FontArialW700H16.{h,cpp}
FontArialW700H18.{h,cpp}

Replace occurrences of 0 by nullptr in calls to GetAdjDet.

IntelSSE.h

Initialized all the members of Result at the beginning of the query code.

IntrArc2Arc2.h
IntrSegment2Arc.h

Exposed the epsilon and tolerance parameters. Removed the LogAssert about tv being in the correct interval, replacing the code by clamping tv to the interval.

Minimize{1,N}.h

At the request of a user, added a working-data structure to improve performance by avoiding multiple std::vector objects.

NaturalSplineCurve.h

At the request of a user, modified the class to track some more information and to have a specialized find-neighbors query when the maximum number of neighbors is 1.

NearestNeighborQuery.h

Added the ability to derive from OverlayEffect. Added a shader parameter to specify the z-depth to allow multiple overlays to be ordered on the screen.

OverlayEffect.{h,cpp}

Added support for multithreaded picking of triangle primitives.

Picker.{h,cpp}

Added more information when SetOffset or SetNumActiveElements triggers an assertion. The assertion is now changed to a warning.

Resource.cpp

Added the ability to create vectors with all 1-valued components.

Vector.h

Added a Load function that takes an HMODULE object to locate resources within the application.

WICFileIO.{h,cpp}

July 30, 2019. Redesigned ApprQuery to replace the Curiously Recurring Template pattern by a simple virtual function system. Eliminated the need to implement two nearly equal versions of the fitters.

ApprQuery.h
ApprGaussian{2,3}.h
ApprHeightLine2.h
ApprHeightPlane3.h
ApprOrthogonalLine{2,3}.h
ApprOrthogonalPlane3.h
ApprPolynomial{2,3,4}.h
ApprPolynomialSpecial{2,3,4}.h

Added comments about the matrices automatically initialized to zero by the Matrix constructor. Reformatted the code.

ApprQuadratic{2,3}.h

Replace the C-style array of endpoints by std::array-based members.

Arc2.h

Added an adapter function Evaluate that takes an input of type Real* and reinterprets it to use the evaluation function that takes an input of type Vector<Real,N>.

ParametricCurve.h

Added a function operator bool() and a member mConstructed to notify the caller whether or not the object was created successfully. If the linear system solver fails, the object is not constructed. Added a function Evaluate that allows you to compute derivatives as well as the position (which was already supported by GetPosition).

BSplineCurveFit.h

Added an adapter functions to compute the bounding object containing points whose input is std::vector.

ContCircle2.h
ContSphere3.h
ContOrientedBox{2,3}.h

Added a conditional define to allow the LogAssert call to occur in GetTangent. The assertion is disabled by default.

ConvexHull2.h

Restored the old algorithm code that used constrained quadratic minimization. This will be used in the upcoming GTL distribution.

DistPointTriangle3.h

Added std::ostringstream code to format the HRESULT as a hexadecimal number for better readability.

DX11Include.h

Added new file for some simple string utilities, including converting between narrow and wide strings.

StringUtility.h

Added const to GetVariable. Set the default constructor and destructor using the new C++ mechanism.

Environment.{h,cpp}

July 29, 2019. Renamed the operator() functions to FitUsingLengths and added added an epsilon parameter used to terminate the iterations. Added an implementation FitUsingSquaredLengths that is bounded in time and involves solving a linear system. It is much faster than FitUsingLengths. The new algorithm is described in the least-squares fitting PDF at the website.

ApprCircle2.h
ApprSphere3.h

July 10, 2019. Implemented std::ldexp for BSNumber and BSRational. Added internal-public member functions to support this.

BSNumber.h
BSRational.h

June 22, 2019. Reimplemented the intersection code for line/ray/segment and cone based on the cone redesign. Added extensive unit tests that cover all nonbranching blocks in the code to verify correctness. The new implementations support arbitrary precision arithmetic. The line-cone PDF document was rewritten to describe the new code.

IntrLine3Cone3.h
IntrRay3Cone3.h
IntrSegment3Cone3.h

June 22, 2019. Added code that allows intersection of intervals of any of the types $[t_0, t_1]$, $[t_0, +\infty)$ or $(-\infty, t_1]$. Boolean inputs are used to specify the type of semiinfinite intervals passed to the queries.

IntrIntervals.h

Revised the cone primitive so that std::numeric_limits<Real>::max() no longer represents infinite height. Instead, the setting of height extremes is now controlled by the constructors and new member functions MakeInfiniteCone, MakeInfiniteTruncatedCone, MakeFiniteCone and MakeConeFrustum. Internally, the maximum height is represented by -1 and used by new member functions IsFinite and IsInfinite to report whether the maximum height of the code is finite or infinite. A new member function HeightInRange allows you to test whether a specified height is in the height range of the cone. These changes allow for geometric operations on cones when the underlying type is arbitrary precision BSNumber and BSRational that do not have representations of infinities.

Cone.h

Modified the containment query to use the new HeightInRange and to avoid using Length so that arbitrary precision BSNumber and BSRational can be used for exact containment queries.

ContCone.h

Modifications based on the cone redesign.

IntrAlignedBox3Cones.h
IntrOrientedBox3Cone3.h
IntrSphere3Cone3.h
Samples/Mathematics/IntersectBoxCone/IntersectBoxConeWindow.cpp
Samples/Mathematics/IntersectSphereCone/IntersectSphereConeWindow.cpp

Fixed a comment.

IntpBSplineUniform.h

June 13, 2019. A C-style array was declared with 3 elements when it should have been 2.

IntrHalfspace3Segment3.h

The B vector was uninitialized but needed to be set to zero. This was the behavior when the code was written, but apparently the array-based vector class used to initialize its members to zero, which it currently does not do.

ApprParaboloid3.h

Removed the use of std::numeric_limits<max> to allow the code to be used with rational arithmetic such as BSNumber and BSRational. The code comments explain the change in semantics.

IntrIntervals.h

Added a new file that replaces GteQuadraticField.h. The new class QFElement implements the necessary arithmetic interfaces to be used as-is in the mathematics code of GTEngine. The QuadraticField class had wrappers to allow sharing of the d-value of the field. I removed the sharing, allowing each quadratic field element to have its own copy.

GTMathematics.h
QFElement.h
IntrSphere3Triangle3.h
QuadraticField.h

June 4, 2019. Changes in the trackball mechanism broke the application because the world box and world triangle were not computed correctly for the intersection queries. I missed this during end-to-end testing. The goal of the sample is to illustrate that the intersection queries work, and the previous sample code was way too complicated for this purpose. Now I use the same paradigm as in other object-object intersection samples. The triangle does not move. The box can be translated and rotated in its local coordinate system.

Samples/Mathematics/IntersectTriangleBox/IntersectBoxTriangle.{h,cpp}

June 2, 2019. Fixed a bug I recently introduced. The DrawPixel functor was setting the pixels to 0 but needed to be 0xFF000000 to produce opaque pixels.

Samples/Physics/SimplePendulum/SimplePendulum.cpp

Removed the single-step mode because the time step was extremely small. Modified the time step to be the same whether debug or release configuration.

Samples/Physics/SimplePendulumFriction/SimplePendulumFrictionWindow.{h,cpp}

Added LogReporter support to be consistent with other sample applications.

Samples/Imagics/AdaptiveSkeletonClimbing3/AdaptiveSkeletonClimbing3.cpp

The assertions for matrix-times-vector and vector-times-matrix were incorrect. They were not caught by unit tests that exposed the assertions.

GMatrix.h

May 30, 2019. The increase in spot angle had a copy-and-paste error that led to decreasing the spot angle. The lighting constant buffer is shared by the vertex and pixel shaders for a specific light type and a specific geometry type. The key processing was modified to ensure that the relevant lighting member is incremented or decremented only once.

Samples/Graphics/Lights/LightsWindow.cpp

Some of the projects had a data-file folder but the contents were files from the Wild Magic 5 distribution. Modified these to contain the files that the current application uses. Other projects did not have data-file folders, so I added them. And the annoying bug in Visual Studio (multiple major versions) that causes HLSL files to be moved out of my shader-file folders showed up again. I set the HLSL files to "Does not participate in build."

Samples/Graphics/Terrain/Terrain.*.vcxproj*

Removed class-static data that is never used.

Font.h

May 29, 2019. Removed unused variables.

Samples/Graphics/BlendedTerrain/BlendedTerrainWindow.cpp

May 19, 2019. The class was missing implementations of three inline functions.

Controller.h

May 18, 2019. Added a test for the Hygon Dhyana processor.

CPUQueryInstructions.cpp

May 17, 2019. The implementation of GetIndex was missing.

TextureCubeArray.h

May 16, 2019. The members were declared as float instead of Real.

GradientAnisotropic2.h

Renamed function input variable from position to velocity.

MassSpringSurface.h

May 7, 2019. Corrected the comments.

ApprPolynomialSpecial{3,4}.h
ContEllipse2.h
ContEllipsoid3.h

May 6, 2019. The Vector3 members had hard-coded float which needed to be Real.

Torus3.h

A relative path for GteCubicRootsQR.h was used rather than the global path.

QuarticRootsQR.h

May 4, 2019. Restored the default null pointer parameters to the Inverse functions.

Matrix2x2.h
Matrix3x3.h

May 3, 2019. Removed GteWrapper.h and GteWrapper.cpp from the engine. These provided Memcpy, a wrapper to deal with the Microsoft security versions of memory copies. Replaced Memcpy by memcpy in other files.

GTEngine.*.vcxproj*
GTLowLevel.h
GaussianElimination.h
GenerateMeshUV.h
LinearSystem.h
MinimizeN.h
SingularValueDecompostion.h
SymmetricEigensolver.h
UnsymmetricEigenvalues.h
DX11Buffer.cpp
DX11Texture.cpp
HLSLShader.cpp
HLSLShaderFactory.cpp
HLSLShaderVariable.cpp
Font.cpp
MarchingCubes.cpp
Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow.cpp
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.cpp
Samples/Imagics/MedianFiltering/MedianFilteringWindow.cpp

Replaced memcpy by std::memcpy.

GaussianElimination.h
GenerateMeshUV.h

LinearSystem.h
MinimizeN.h
SingularValueDecompostion.h
SymmetricEigensolver.h
UnsymmetricEigenvalues.h
DX11Buffer.cpp
DX11Texture.cpp
HLSLShader.cpp
HLSLShaderFactory.cpp
HLSLShaderVariable.cpp
Font.cpp
CPUQueryInstructions.cpp
Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow.cpp
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.cpp
Samples/Geometrics/SplitMeshByPlane/SplitMeshByPlaneWindow.cpp
Samples/Geometrics/VertexCollapseMesh/VertexCollapseMeshWindow.cpp
Samples/Graphics/Castle/LoadData.cpp
Samples/Graphics/CubeMaps/CubeMapsWindow.cpp
Samples/Graphics/GeometryShaders/GeometryShadersWindow.cpp
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargetsWindow.cpp
Samples/Graphics/TextureArrays/TextureArraysWindow.cpp
Samples/Graphics/TextureUpdating/TextureUpdatingWindow.cpp
Samples/Imagics/AdaptiveSkeletonClimbing2/AdaptiveSkeletonClimbing2Window.cpp
Samples/Imagics/ExtractLevelCurves/ExtractLevelCurvesWindow.cpp
Samples/Imagics/ExtractLevelSurfaces/ExtractLevelSurfacesWindow.cpp
Samples/Imagics/ExtractRidges/ExtractRidges.cpp
Samples/Imagics/MedianFiltering/MedianFilteringWindow.cpp
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp
Samples/Mathematics/DistancePointConvexPolyhedron/DistancePointConvexPolyhedronWindow.cpp
Samples/Mathematics/MovingSphereBox/MovingSphereBoxWindow.cpp
Samples/Mathematics/NURBSSphere/NURBSSphereWindow.cpp
Samples/Physics/BouncingBall/DeformableBall.cpp

Replaced `memset` by `std::memset`.

BasisFunction.h
BSplineCurveFit.h
BSplineSurfaceFit.h
GaussianElimination.h
AkimaUniform2.h
AkimaUniform3.h
LinearSystem.h
NURBSSurface.h
NURBSVolume.h
SingularValueDecompostion.h
SymmetricEigensolver.h
MassSpringArbitrary.h

DX11InputLayout.cpp
GL4InputLayout.cpp
WGLEngine.cpp
ConstantBuffer.cpp
TextureBuffer.cpp
CPUQueryInstructions.cpp
Fluid2InitializeSource.cpp
Fluid2InitializeState.cpp
Fluid3InitializeSource.cpp
Fluid3InitializeState.cpp

Samples/Graphics/StructuredBuffers/StructuredBuffersWindow.cpp
Samples/Graphics/TextureUpdating/TextureUpdatingWindow.cpp
Samples/Imagics/BSplineInterpolation/BSplineInterpolation.cpp
Samples/Imagics/GpuGaussianBlur3/GpuGaussianBlur3Window.cpp
Samples/Imagics/VideoStreams/VideoStreamsWindow.cpp
Samples/Mathematics/GeodesicHeightFields/GeodesicHeightFieldsWindow.cpp
Samples/Mathematics/MovingSphereBox/MovingSphereBoxWindow.cpp
Samples/Mathematics/NURBSSphere/NURBSSphereWindow.cpp
Samples/Mathematics/PartialSums/PartialSums.cpp
Samples/Mathematics/ShortestPath/GpuShortestPath.cpp
Samples/Physics/BallHill/BallHillWindow.cpp
Samples/Physics/FoucaultPendulum/FoucaultPendulumWindow.cpp
Samples/Physics/SimplePendulum/SimplePendulum.cpp

May 2, 2019. Replaced GTEngine.h by only those headers necessary to compile the application. The remaining reasons vary with sample application. Eliminated explicit references to GTE_USE_MAT_VEC. Eliminated explicit references to GTE_DEV_OPENGL. Eliminated explicit references to ProgramFactory::PF_* enumerates. Split the HLSL files that contained both a vertex shader into files, each containing a single shader. Renamed the GLSL shader files.

Samples/Basics/AppendConsumeBuffers/AppendConsumeBuffers.*.vcxproj*
Samples/Basics/AppendConsumeBuffers/AppendConsumeBuffers.cpp
Samples/Basics/AppendConsumeBuffers/Shaders/AppendConsume.{glsl,hlsl}
Samples/Basics/AppendConsumeBuffers/Shaders/AppendConsume.cs.{glsl,hlsl}
Samples/Basics/IEEEFloatingPoint/IEEEFloatingPoint.*.vcxproj*
Samples/Basics/IEEEFloatingPoint/IEEEFloatingPoint.cpp
Samples/Basics/IEEEFloatingPoint/Shaders/TestSubnormals.{glsl,hlsl}
Samples/Basics/IEEEFloatingPoint/Shaders/TestSubnormals.cs.{glsl,hlsl}
Samples/Geometrics/CLODPolyline/CLODPolylineWindow.{h,cpp}
Samples/Geometrics/ConformalMapping/ConformalMappingWindow.{h,cpp}
Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2DWindow.{h,cpp}
Samples/Geometrics/ConvexHull2D/ConvexHull2DWindow.{h,cpp}
Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow.{h,cpp}
Samples/Geometrics/Delaunay2D/Delaunay2DWindow.{h,cpp}
Samples/Geometrics/Delaunay3D/Delaunay3DWindow.{h,cpp}
Samples/Geometrics/DisjointIntervalsRectangles/DisjointIntervalsRectangles.cpp

Samples/Geometrics/GenerateMeshUVs/GenerateMeshUVsWindow.{h,cpp}
Samples/Geometrics/MinimalCycleBasis/MinimalCycleBasisWindow.{h,cpp}
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.{h,cpp}
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2DWindow.{h,cpp}
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.{h,cpp}
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3DWindow.{h,cpp}
Samples/Geometrics/PolygonBooleanOperations/PolygonBooleanOperationsWindow.{h,cpp}
Samples/Geometrics/SplitMeshByPlane/SplitMeshByPlaneWindow.{h,cpp}
Samples/Geometrics/TriangulateCDT/TriangulateCDTWindow.{h,cpp}
Samples/Geometrics/TriangulateEC/TriangulateECWindow.{h,cpp}
Samples/Geometrics/VertexCollapseMesh/VertexCollapseMeshWindow.{h,cpp}
Samples/Graphics/AreaLights/AreaLights.*.vcxproj*
Samples/Graphics/AreaLights/AreaLightEffect.{h,cpp}
Samples/Graphics/AreaLights/AreaLightsWindow.{h,cpp}
Samples/Graphics/AreaLights/Shaders/AreaLight.hlsl
Samples/Graphics/AreaLights/Shaders/AreaLight{VS,PS}.glsl
Samples/Graphics/AreaLights/Shaders/AreaLight.{vs,ps}.{hlsl,glsl}
Samples/Graphics/BillboardNodes/BillboardNodesWindow.{h,cpp}
Samples/Graphics/BlendedAnimations/BlendedAnimationsWindow.{h,cpp}
Samples/Graphics/BlendedTerrain/BlendedTerrain.*.vcxproj*
Samples/Graphics/BlendedTerrain/BlendedTerrainEffect.{h,cpp}
Samples/Graphics/BlendedTerrain/BlendedTerrainWindow.{h,cpp}
Samples/Graphics/BlendedTerrain/Shaders/BlendedTerrain.hlsl
Samples/Graphics/BlendedTerrain/Shaders/BlendedTerrain{VS,PS}.glsl
Samples/Graphics/BlendedTerrain/Shaders/BlendedTerrain.{vs,ps}.{hlsl,glsl}
Samples/Graphics/BlownGlass/BlownGlass.*.vcxproj*
Samples/Graphics/BlownGlass/BlownGlassWindow.{h,cpp}
Samples/Graphics/BlownGlass/Shaders/VolumeRender.hlsl
Samples/Graphics/BlownGlass/Shaders/VolumeRender{VS,PS}.glsl
Samples/Graphics/BlownGlass/Shaders/VolumeRender.{vs,ps}.{hlsl,glsl}
Samples/Graphics/BspNodes/BspNodesWindow.{h,cpp}
Samples/Graphics/BufferUpdating/BufferUpdatingWindow.{h,cpp}
Samples/Graphics/BumpMaps/BumpMaps.*.vcxproj*
Samples/Graphics/BumpMaps/SimpleBumpMapEffect.cpp
Samples/Graphics/BumpMaps/BumpMapsWindow.{h,cpp}
Samples/Graphics/BumpMaps/Shaders/SimpleBumpMap.hlsl
Samples/Graphics/BumpMaps/Shaders/SimpleBumpMap{VS,PS}.glsl
Samples/Graphics/BumpMaps/Shaders/SimpleBumpMap.{vs,ps}.{hlsl,glsl}
Samples/Graphics/CameraAndLightNodes/CameraAndLightNodesWindow.{h,cpp}
Samples/Graphics/Castle/CastleWindow.{h,cpp}
Samples/Graphics/Castle/CreateMeshes.cpp
Samples/Graphics/Castle/TexturePNT1Effect.cpp
Samples/Graphics/CubeMaps/CubeMaps.*.vcxproj*
Samples/Graphics/CubeMaps/CubeMapEffect.cpp
Samples/Graphics/CubeMaps/CubeMapsWindow.{h,cpp}
Samples/Graphics/CubeMaps/Shaders/CubeMap.hlsl
Samples/Graphics/CubeMaps/Shaders/CubeMap{VS,PS}.glsl
Samples/Graphics/CubeMaps/Shaders/CubeMap.{vs,ps}.{hlsl,glsl}

Samples/Graphics/GeometryShaders/GeometryShaders.*.vcxproj*
Samples/Graphics/GeometryShaders/GeometryShadersWindow.{h,cpp}
Samples/Graphics/GeometryShaders/RandomSquares.hlsl
Samples/Graphics/GeometryShaders/RandomSquaresIndirect.hlsl
Samples/Graphics/GeometryShaders/RandomSquaresDirect{VS,GS}.glsl
Samples/Graphics/GeometryShaders/RandomSquaresIndirect{VS,GS}.glsl
Samples/Graphics/GeometryShaders/RandomSquaresPS.glsl
Samples/Graphics/GeometryShaders/RandomSquaresDirect.{vs,gs,ps}.{hlsl,glsl}
Samples/Graphics/GeometryShaders/RandomSquaresIndirect.{vs,gs,ps}.{hlsl,glsl}
Samples/Graphics/GlossMaps/GlossMaps.*.vcxproj*
Samples/Graphics/GlossMaps/GlossMapEffect.{h,cpp}
Samples/Graphics/GlossMaps/GlossMapsWindow.{h,cpp}
Samples/Graphics/GlossMaps/Shaders/GlossMap.hlsl
Samples/Graphics/GlossMaps/Shaders/GlossMap.{vs,ps}.{hlsl,glsl}
Samples/Graphics/IKControllers/IKControllersWindow.{h,cpp}
Samples/Graphics/Lights/LightsWindow.{h,cpp}
Samples/Graphics/LightTexture/LightTextureWindow.{h,cpp}
Samples/Graphics/MorphControllers/MorphControllersWindow.{h,cpp}
Samples/Graphics/MorphFaces/MorphFaces.*.vcxproj*
Samples/Graphics/MorphFaces/CubicInterpolator.h
Samples/Graphics/MorphFaces/MorphFacesWindow.{h,cpp}
Samples/Graphics/MorphFaces/Shaders/Texture2PNT{VS,PS}.glsl
Samples/Graphics/MorphFaces/Shaders/Texture2PNT.{vs,ps}.{hlsl,glsl}
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargets.*.vcxproj*
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargetsWindow.{h,cpp}
Samples/Graphics/MultipleRenderTargets/Shaders/MultipleRenderTargets.hlsl
Samples/Graphics/MultipleRenderTargets/Shaders/MultipleRenderTargets{Vertex,Pixel}.glsl
Samples/Graphics/MultipleRenderTargets/Shaders/MultipleRenderTargets.{vs,ps}.{hlsl,glsl}
Samples/Graphics/ParticleControllers/ParticleControllersWindow.{h,cpp}
Samples/Graphics/Picking/PickingWindow.{h,cpp}
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersection.*.vcxproj*
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersectionWindow.{h,cpp}
Samples/Graphics/PlaneMeshIntersection/Shaders/DrawIntersections.{hlsl,glsl}
Samples/Graphics/PlaneMeshIntersection/Shaders/PlaneMeshIntersection.hlsl
Samples/Graphics/PlaneMeshIntersection/Shaders/PlaneMeshIntersection{Vertex,Pixel}.glsl
Samples/Graphics/PlaneMeshIntersection/Shaders/DrawIntersections.cs.{hlsl,glsl}
Samples/Graphics/PlaneMeshIntersection/Shaders/PlaneMeshIntersection.{vs,ps}.{hlsl,glsl}
Samples/Graphics/PointControllers/PointControllersWindow.{h,cpp}
Samples/Graphics/ProjectedTextures/ProjectedTextures.*.vcxproj*
Samples/Graphics/ProjectedTextures/ProjectedTexturesWindow.{h,cpp}
Samples/Graphics/ProjectedTextures/Shaders/ProjectedTextureEffect.cpp
Samples/Graphics/ProjectedTextures/Shaders/ProjectedTexture.hlsl
Samples/Graphics/ProjectedTextures/Shaders/ProjectedTexture{VS,PS}.glsl
Samples/Graphics/ProjectedTextures/Shaders/ProjectedTexture.{vs,ps}.{hlsl,glsl}
Samples/Graphics/SphereMaps/SphereMaps.*.vcxproj*
Samples/Graphics/SphereMaps/SphereMapsWindow.{h,cpp}
Samples/Graphics/SphereMaps/Shaders/SphereMapEffect.cpp
Samples/Graphics/SphereMaps/Shaders/SphereMap.hlsl

Samples/Graphics/SphereMaps/Shaders/SphereMap{VS,PS}.glsl
Samples/Graphics/SphereMaps/Shaders/SphereMap.{vs,ps}.{hlsl,glsl}
Samples/Graphics/StructuredBuffers/StructuredBuffers.*.vcxproj*
Samples/Graphics/StructuredBuffers/StructuredBuffersWindow.{h,cpp}
Samples/Graphics/StructuredBuffers/Shaders/StructuredBuffers.hlsl
Samples/Graphics/StructuredBuffers/Shaders/StructuredBuffers{VS,PS}.glsl
Samples/Graphics/StructuredBuffers/Shaders/StructuredBuffers.{vs,ps}.{hlsl,glsl}
Samples/Graphics/Terrain/Terrain.*.vcxproj*
Samples/Graphics/Terrain/TerrainWindow.{h,cpp}
Samples/Graphics/Terrain/Shaders/TerrainEffect.cpp
Samples/Graphics/Terrain/Shaders/Terrain.hlsl
Samples/Graphics/Terrain/Shaders/Terrain{VS,PS}.glsl
Samples/Graphics/Terrain/Shaders/Terrain.{vs,ps}.{hlsl,glsl}
Samples/Graphics/TextureArrays/TextureArrays.*.vcxproj*
Samples/Graphics/TextureArrays/TextureArraysWindow.{h,cpp}
Samples/Graphics/TextureArrays/Shaders/TextureArrays.hlsl
Samples/Graphics/TextureArrays/Shaders/TextureArrays{Vertex,Pixel}.glsl
Samples/Graphics/TextureArrays/Shaders/TextureArrays.{vs,ps}.{hlsl,glsl}
Samples/Graphics/TextureUpdating/TextureUpdatingWindow.{h,cpp}
Samples/Graphics/Texturing/TexturingWindow.{h,cpp}
Samples/Graphics/VertexColoring/VertexColoringWindow.{h,cpp}
Samples/Graphics/VertexTextures/VertexTextures.*.vcxproj*
Samples/Graphics/VertexTextures/VertexTexturesWindow.{h,cpp}
Samples/Graphics/VertexTextures/Shaders/DisplacementEffect.{h,cpp}
Samples/Graphics/VertexTextures/Shaders/Displacement.hlsl
Samples/Graphics/VertexTextures/Shaders/Displacement{VS,PS}.glsl
Samples/Graphics/VertexTextures/Shaders/Displacement.{vs,ps}.{hlsl,glsl}
Samples/Graphics/WireMesh/WireMesh.*.vcxproj*
Samples/Graphics/WireMesh/WireMeshWindow.{h,cpp}
Samples/Graphics/WireMesh/Shaders/WireMesh.hlsl
Samples/Graphics/WireMesh/Shaders/WireMesh{VS,PS,GS}.glsl
Samples/Graphics/WireMesh/Shaders/WireMesh.{vs,ps,gs}.{hlsl,glsl}
Samples/Imagics/AdaptiveSkeletonClimbing2/AdaptiveSkeletonClimbing2.cpp
Samples/Imagics/AdaptiveSkeletonClimbing3/AdaptiveSkeletonClimbing3.cpp
Samples/Imagics/BSplineInterpolation/BSplineInterpolation.cpp
Samples/Imagics/Convolution/Convolution.*.vcxproj*
Samples/Imagics/Convolution/ConvolutionWindows.{h,cpp}
Samples/Imagics/Convolution/Shaders/Convolve*.{hlsl,glsl}
Samples/Imagics/Convolution/Shaders/Convolve*.cs.{hlsl,glsl}
Samples/Imagics/ExtractLevelCurves/ExtractLevelCurvesWindow.{h,cpp}
Samples/Imagics/ExtractLevelSurfaces/ExtractLevelSurfacesWindow.{h,cpp}
Samples/Imagics/ExtractRidges/ExtractRidges.cpp
Samples/Imagics/GaussianBlurring/GaussianBlurring.*.vcxproj*
Samples/Imagics/GaussianBlurring/GaussianBlurringWindow.{h,cpp}
Samples/Imagics/GaussianBlurring/Shaders/GaussianBlur3x3.{hlsl,glsl}
Samples/Imagics/GaussianBlurring/Shaders/GaussianBlur3x3.cs.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur2/GpuGaussianBlur2.*.vcxproj*
Samples/Imagics/GpuGaussianBlur2/GpuGaussianBlur2Window.{h,cpp}

Samples/Imagics/GpuGaussianBlur2/Shaders/BoundaryDirichlet.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/BoundaryNeumann.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/GaussianBlur.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/BoundaryDirichlet.cs.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/BoundaryNeumann.cs.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/GaussianBlur.cs.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur3/GpuGaussianBlur3.*.vcxproj*
Samples/Imagics/GpuGaussianBlur3/GpuGaussianBlur3Window.{h,cpp}
Samples/Imagics/GpuGaussianBlur3/Shaders/BoundaryDirichlet.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/DrawImage.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/BoundaryNeumann.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/GaussianBlur.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/BoundaryDirichlet.cs.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/BoundaryNeumann.cs.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/DrawImage.ps.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/GaussianBlur.cs.{hlsl,glsl}
Samples/Imagics/MedianFiltering/MedianFiltering.*.vcxproj*
Samples/Imagics/MedianFiltering/MedianFilteringWindow.{h,cpp}
Samples/Imagics/MedianFiltering/Shaders/Median3x3.{hlsl,glsl}
Samples/Imagics/MedianFiltering/Shaders/Median5x5.{hlsl,glsl}
Samples/Imagics/MedianFiltering/Shaders/MedianBySort.{hlsl,glsl}
Samples/Imagics/MedianFiltering/Shaders/MedianShared.hlsli
Samples/Imagics/MedianFiltering/Shaders/Median3x3.cs.{hlsl,glsl}
Samples/Imagics/MedianFiltering/Shaders/Median5x5.cs.{hlsl,glsl}
Samples/Imagics/MedianFiltering/Shaders/MedianBySort.cs.{hlsl,glsl}
Samples/Imagics/MedianFiltering/Shaders/MedianShared.cs.hlsli
Samples/Imagics/SurfaceExtraction/SurfaceExtraction.*.vcxproj*
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.{h,cpp}
Samples/Imagics/SurfaceExtraction/Shaders/DrawSurfaceIndirect.hlsl
Samples/Imagics/SurfaceExtraction/Shaders/DrawSurfaceIndirect{VS,GS,PS}.glsl
Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurface.{hlsl,glsl}
Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurfaceIndirect.{hlsl,glsl}
Samples/Imagics/SurfaceExtraction/Shaders/DrawSurfaceIndirect.vs,gs,ps.hlsl
Samples/Imagics/SurfaceExtraction/Shaders/DrawSurfaceIndirect.{vs,gs,ps}.glsl
Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurface.cs.{hlsl,glsl}
Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurfaceIndirect.cs.{hlsl,glsl}
Samples/Imagics/VideoStreams/VideoStreamsWindow.{h,cpp}
Samples/Mathematics/AllPairsTriangles/AllPairsTriangles.*.vcxproj*
Samples/Mathematics/AllPairsTriangles/AllPairsTrianglesWindow.{h,cpp}
Samples/Mathematics/AllPairsTriangles/TriangleIntersection.h
Samples/Mathematics/AllPairsTriangles/Shaders/DrawUsingVertexID.hlsl
Samples/Mathematics/AllPairsTriangles/Shaders/DrawUsingVertexID{VS,PS}.glsl
Samples/Mathematics/AllPairsTriangles/Shaders/InitializeColors.{hlsl,glsl}
Samples/Mathematics/AllPairsTriangles/Shaders/TriangleIntersection.{hlsl,glsl}
Samples/Mathematics/AllPairsTriangles/Shaders/VertexColorIndexed.hlsl
Samples/Mathematics/AllPairsTriangles/Shaders/VertexColorIndexed{VS,PS}.glsl
Samples/Mathematics/AllPairsTriangles/Shaders/DrawUsingVertexID.{vs,ps}.{hlsl,glsl}
Samples/Mathematics/AllPairsTriangles/Shaders/InitializeColors.cs.{hlsl,glsl}

Samples/Mathematics/AllPairsTriangles/Shaders/TriangleIntersection.cs.{hlsl,glsl}
Samples/Mathematics/AllPairsTriangles/Shaders/VertexColorIndexed.{vs,ps}.{hlsl,glsl}
Samples/Mathematics/ApproximateEllipseByArcs/ApproximateEllipseByArcs.{h,cpp}
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitterWindow.{h,cpp}
Samples/Mathematics/BSplineCurveReduction/BSplineCurveReductionWindow.{h,cpp}
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitterWindow.{h,cpp}
Samples/Mathematics/DistanceAlignedBoxes/DistanceAlignedBoxesWindow.{h,cpp}
Samples/Mathematics/DistanceAlignedBoxOrientedBox/DistanceAlignedBoxOrientedBoxWindow.{h,cpp}
Samples/Mathematics/DistanceOrientedBoxes/DistanceOrientedBoxesWindow.{h,cpp}
Samples/Mathematics/DistancePointConvexPolyhedron/DistancePointConvexPolyhedronWindow.{h,cpp}
Samples/Mathematics/DistancePointHyperellipsoid/DistancePointHyperellipsoid.cpp
Samples/Mathematics/DistancePointConvexPolyhedron/DistancePointConvexPolyhedronWindow.{h,cpp}
Samples/Mathematics/DistanceRectangleBox/DistanceRectangleBoxWindow.{h,cpp}
Samples/Mathematics/DistanceSegments3/DistanceSegments3.*.vcxproj*
Samples/Mathematics/DistanceSegments3/DistanceSegments3.cpp
Samples/Mathematics/DistanceSegments3/Shaders/DistanceSeg3Seg3.{hlsl,glsl}
Samples/Mathematics/DistanceSegments3/Shaders/DistanceSeg3Seg3.cs.{hlsl,glsl}
Samples/Mathematics/DistanceTriangleBox/DistanceTriangleBoxWindow.{h,cpp}
Samples/Mathematics/FitCone/FitConeWindow.{h,cpp}
Samples/Mathematics/FitCylinder/FitCylinderWindow.{h,cpp}
Samples/Mathematics/FitTorus/FitTorusWindow.{h,cpp}
Samples/Mathematics/GeodesicEllipsoid/GeodesicEllipsoidWindow.{h,cpp}
Samples/Mathematics/GeodesicHeightField/GeodesicHeightFieldWindow.{h,cpp}
Samples/Mathematics/Interpolation2D/Interpolation2DWindow.{h,cpp}
Samples/Mathematics/IntersectBoxCone/IntersectBoxConeWindow.{h,cpp}
Samples/Mathematics/IntersectBoxCylinder/IntersectBoxCylinderWindow.{h,cpp}
Samples/Mathematics/IntersectBoxSphere/IntersectBoxSphereWindow.{h,cpp}
Samples/Mathematics/IntersectConvexPolyhedra/IntersectConvexPolyhedraWindow.{h,cpp}
Samples/Mathematics/IntersectInfiniteCylinders/IntersectInfiniteCylindersWindow.{h,cpp}
Samples/Mathematics/IntersectSphereCone/IntersectSphereConeWindow.{h,cpp}
Samples/Mathematics/IntersectTriangleBox/IntersectTriangleBoxWindow.{h,cpp}
Samples/Mathematics/IntersectTriangles2D/IntersectTriangles2DWindow.{h,cpp}
Samples/Mathematics/MovingCircleRectangle/MovingCircleRectangleWindow.{h,cpp}
Samples/Mathematics/MovingSphereBox/MovingSphereBoxWindow.{h,cpp}
Samples/Mathematics/NURBSCircle/NURBSCircleWindow.{h,cpp}
Samples/Mathematics/NURBSCurveExample/NURBSCurveExampleWindow.{h,cpp}
Samples/Mathematics/NURBSSphere/NURBSSphereWindow.{h,cpp}
Samples/Mathematics/PartialSums/PartialSums.*.vcxproj*
Samples/Mathematics/PartialSums/PartialSums.cpp
Samples/Mathematics/PartialSums/Shaders/PartialSums.{hlsl,glsl}
Samples/Mathematics/PartialSums/Shaders/PartialSums.cs.{hlsl,glsl}
Samples/Mathematics/PlaneEstimation/PlaneEstimation.*.vcxproj*
Samples/Mathematics/PlaneEstimation/PlaneEstimationWindow.{h,cpp}
Samples/Mathematics/PlaneEstimation/Shaders/EvaluateBezier.{hlsl,glsl}
Samples/Mathematics/PlaneEstimation/Shaders/PlaneEstimation.{hlsl,glsl}
Samples/Mathematics/PlaneEstimation/Shaders/PlaneVisualize.{hlsl,glsl}
Samples/Mathematics/PlaneEstimation/Shaders/PositionVisualize.{hlsl,glsl}
Samples/Mathematics/PlaneEstimation/Shaders/EvaluateBezier.cs.{hlsl,glsl}

Samples/Mathematics/PlaneEstimation/Shaders/PlaneEstimation.cs.{hlsl,glsl}
Samples/Mathematics/PlaneEstimation/Shaders/PlaneVisualize.ps.{hlsl,glsl}
Samples/Mathematics/PlaneEstimation/Shaders/PositionVisualize.ps.{hlsl,glsl}
Samples/Mathematics/RootFinding/RootFinding.*.vcxproj*
Samples/Mathematics/RootFinding/RootFinding.cpp
Samples/Mathematics/RootFinding/Shaders/RootFinder.{hlsl,glsl}
Samples/Mathematics/RootFinding/Shaders/RootFinder.cs.{hlsl,glsl}
Samples/Mathematics/ShortestPath/ShortestPath.*.vcxproj*
Samples/Mathematics/ShortestPath/ShortestPathWindow.{h,cpp}
Samples/Mathematics/ShortestPath/CpuShortestPath.{h,cpp}
Samples/Mathematics/ShortestPath/GpuShortestPath.{h,cpp}
Samples/Mathematics/ShortestPath/Shaders/InitializeDiagToCol.{hlsl,glsl}
Samples/Mathematics/ShortestPath/Shaders/InitializeDiagToRow.{hlsl,glsl}
Samples/Mathematics/ShortestPath/Shaders/PartialSumsDiagToCol.{hlsl,glsl}
Samples/Mathematics/ShortestPath/Shaders/PartialSumsDiagToRow.{hlsl,glsl}
Samples/Mathematics/ShortestPath/Shaders/UpdateShader.{hlsl,glsl}
Samples/Mathematics/ShortestPath/Shaders/WeightsShader.{hlsl,glsl}
Samples/Mathematics/ShortestPath/Shaders/InitializeDiagToCol.cs.{hlsl,glsl}
Samples/Mathematics/ShortestPath/Shaders/InitializeDiagToRow.cs.{hlsl,glsl}
Samples/Mathematics/ShortestPath/Shaders/PartialSumsDiagToCol.cs.{hlsl,glsl}
Samples/Mathematics/ShortestPath/Shaders/PartialSumsDiagToRow.cs.{hlsl,glsl}
Samples/Mathematics/ShortestPath/Shaders/UpdateShader.cs.{hlsl,glsl}
Samples/Mathematics/ShortestPath/Shaders/WeightsShader.cs.{hlsl,glsl}
Samples/Mathematics/SymmetricEigensolver3x3/SymmetricEigensolver3x3.cpp
Samples/Mathematics/ThinPlateSplines/ThinPlateSplines.cpp
Samples/Physics/BallHill/BallHillWindow.{h,cpp}
Samples/Physics/BallRubberBand/BallRubberBandWindow.{h,cpp}
Samples/Physics/BouncingBall/BouncingBallWindow.{h,cpp}
Samples/Physics/Cloth/ClothWindow.{h,cpp}
Samples/Physics/DoublePendulum/DoublePendulumWindow.{h,cpp}
Samples/Physics/ExtremalQuery/ExtremalQueryWindow.{h,cpp}
Samples/Physics/FlowingSkirt/FlowingSkirtWindow.{h,cpp}
Samples/Physics/Fluids2D/Fluids2D.*.vcxproj*
Samples/Physics/Fluids2D/Fluids2DWindow.{h,cpp}
Samples/Physics/Fluids2D/Shaders/DrawDensity.{hlsl,glsl}
Samples/Physics/Fluids2D/Shaders/DrawDensity.ps.{hlsl,glsl}
Samples/Physics/Fluids3D/Fluids3D.*.vcxproj*
Samples/Physics/Fluids3D/Fluids3DWindow.{h,cpp}
Samples/Physics/Fluids3D/Shaders/VolumeRender.hlsl
Samples/Physics/Fluids3D/Shaders/VolumeRenderVS.glsl
Samples/Physics/Fluids3D/Shaders/VolumeRenderPS.glsl
Samples/Physics/Fluids3D/Shaders/VolumeRender.vs.{hlsl,glsl}
Samples/Physics/Fluids3D/Shaders/VolumeRender.ps.{hlsl,glsl}
Samples/Physics/FoucaultPendulum/FoucaultPendulumWindow.{h,cpp}
Samples/Physics/FreeFormDeformation/FreeFormDeformationWindow.{h,cpp}
Samples/Physics/FreeTopFixedTip/FreeTopFixedTipWindow.{h,cpp}
Samples/Physics/GelatinBlob/GelatinBlobWindow.{h,cpp}
Samples/Physics/GelatinCube/GelatinCubeWindow.{h,cpp}

Samples/Physics/HelixTubeSurface/HelixTubeSurfaceWindow.{h,cpp}
Samples/Physics/IntersectingBoxes/IntersectingBoxesWindow.{h,cpp}
Samples/Physics/IntersectingRectangles/IntersectingRectanglesWindow.{h,cpp}
Samples/Physics/KeplerPolarForm/KeplerPolarFormWindow.{h,cpp}
Samples/Physics/MassPulleySpring/MassPulleySpringWindow.{h,cpp}
Samples/Physics/MassSprings3D/MassSprings3D.*.vcxproj*
Samples/Physics/MassSprings3D/MassSprings3DWindow.{h,cpp}
Samples/Physics/MassSprings3D/CpuMassSpringVolume.{h,cpp}
Samples/Physics/MassSprings3D/GpuMassSpringVolume.{h,cpp}
Samples/Physics/MassSprings3D/Shaders/DrawUsingVertexID.hlsl
Samples/Physics/MassSprings3D/Shaders/DrawUsingVertexID{VS,PS}.glsl
Samples/Physics/MassSprings3D/Shaders/RungeKutta.hlsl
Samples/Physics/MassSprings3D/Shaders/RungeKutta{1a,1b,2a,2b,3a,3b,4a,4b}.glsl
Samples/Physics/MassSprings3D/Shaders/DrawUsingVertexID.{vs,ps}.{hlsl,glsl}
Samples/Physics/MassSprings3D/Shaders/RungeKutta.cs.hlsli
Samples/Physics/MassSprings3D/Shaders/RungeKutta{1a,1b,2a,2b,3a,3b,4a,4b}.cs.hlsl
Samples/Physics/MassSprings3D/Shaders/RungeKutta{1a,1b,2a,2b,3a,3b,4a,4b}.cs.glsl
Samples/Physics/Rope/RopeWindow.{h,cpp}
Samples/Physics/SimplePendulum/SimplePendulum.cpp
Samples/Physics/SimplePendulumFriction/SimplePendulumFrictionWindow.{h,cpp}
Tools/GenerateApproximations/GenerateApproximations.cpp
Tools/GenerateApproximations/FitASin.h
Tools/GenerateApproximations/FitATan.h
Tools/GenerateApproximations/FitCos.h
Tools/GenerateApproximations/FitExp2.h
Tools/GenerateApproximations/FitInvSqrt.h
Tools/GenerateApproximations/FitLog2.h
Tools/GenerateApproximations/FitReciprocal.h
Tools/GenerateApproximations/FitSin.h
Tools/GenerateApproximations/FitSqrt.h
Tools/GenerateApproximations/FitTan.h
Tools/PrecisionCalculator/PrecisionCalculator.cpp

Here are some notes about fixes to some bugs that were discovered during testing after the aforementioned changes.

- The BumpMaps sample show incorrect results when OpenGL is used and the torus is textured rather than bump mapped. The problem is that the GLSL location values for SimpleBumpMap.vs.glsl are hard-coded for the vertex data structure used in bump mapping. The channels in order are position, normal, light direction, base texture coordinate and normal texture coordinate. When the 'b' key is pressed to toggle to texturing only, the GLSL shader embedded in Texture2Effect has hard-coded location values for a vertex with position and texture coordinate, which conflicts with the vertex used in bump mapping. I rewrote the code to have two torii, one for bump mapping and one for texturing. Longer term, I need to modify the GLSL processing code to allow the location to use DX-like semantics which are replaced by actual locations based on the attached vertex buffer.

- The GeometryShaders sample triggers an assertion when using DX11-Debug and USE_DRAW_DIRECT commented out. The assertion code occurs in the DX11 block but needed to be in the OpenGL block. Also fixed Shader for OpenGL to deal with vendor-specific ordering of structured buffer layout information.

- The AllPairsTriangles sample was not rendering correctly in the debug or release configurations using OpenGL and when USE_CPU_FIND_INTERSECTIONS is active. The vertex format was a position as a 3-tuple of float and a color index of type unsigned int. Thinking the 3-tuple is expanded to a 4-tuple in GLSL, I modified the CPU position to be a 4-tuple, but the rendering was still not correct. The color index is type DF_R32_UINT, and I verified that the OpenGL vertex array object corresponding to the color index is GL_UINT. I then modified the vertex format to be a 4-tuple of float with vertex (x, y, z, colorIndex), and I modified the shaders accordingly. The rendering is now correct. I do not know whether I am incorrectly setting up the buffers or whether the graphics driver has a bug.

- The ApproximateEllipseByArcs for debug or release OpenGL configurations showed a black screen. This is a result of two calls to mEngine-¿DisplayColorBuffer(0), one in the Window2::OnDisplay call and one after the message drawing in ApproximateEllipseByArcs::OnDisplay. I added an override of the virtual DrawScreenOverlay and moved the message drawing to it so that only one swap buffers occurs.

- I modified the DistanceSegment3 project CPU code and shader code to make the struct packing simpler. The random number generation for the segment components was replaced by reading from input files. This allows for reproducibility of the results when using different compilers. I added the output data as comments in the code.

- Fixed the trackball motion in IntersectTriangleBox by replacing the old code (that was broken by a previous update to the trackball design) with new code that is a lot simpler.

- Fixed a compiler error in MassSprings3D. The CPU-based mass-spring code had an input ProgramFactory& that had been changed to a smart-pointer type in the GPU-based code.

Eliminated the explicit use of GTE_DEV_OPENGL conditional compilation.

DirectionalLightTextureEffect.cpp
OverlayEffect.cpp
PointLightTextureEffect.cpp
TextEffect.cpp
Texture2Effect.cpp
Texture3Effect.cpp
Fluid2UpdateState.cpp
Fluid3UpdateState.cpp

Added a Set function that has a texture-sampler pair. This is designed to avoid having explicit GTE_DEV_OPENGL conditional compilation in the sample applications. I also fixed a problem with the structured buffer layouts when compiling for OpenGL. The GL reports the members of the layout, but the order can vary for different OpenGL implementations. For example, the ordering might be by member offset or it might be by member name in alphabetical order. I modified the OpenGL-based Shader constructor to sort each layout by member offset.

Shader.{h,cpp}

Added DoTransform functions to transform a vector v by a matrix M according to whether the user has selected the convention GTE_USE_MAT_VEC (function returns Mv) or GTE_USE_VEC_MAT (function returns vM). Added SetBasis and GetBasis functions to treat the columns of a matrix as a basis when GTE_USE_MAT_VEC is active or to treat the rows of a matrix as a basis when GTE_USE_VEC_MAT is active. This allows the sample applications to hide the conditional compilation for transforming.

Matrix2x2.h
Matrix3x3.h
Matrix4x4.h

Fixed a compiler error when GTE_USE_VEC_MAT was enabled.

Trackball.cpp

April 13, 2019. Exposed the include of fstream in the header file so that callers of CreateFromFiles do not have to include explicitly that header. Before the change, fstream was exposed indirectly by GTGraphicsDX11.h but not GTGraphicsGL4.h.

ProgramFactory.{h,cpp}

Modified the top-level headers to encapsulate better the graphics, window and application layers. The goal is to replace GTEngine.h in the sample applications by only those header files needed for compilation, avoiding the slow compile times for the BuildAll solutions in Microsoft Visual Studio. Also, a goal is to avoid exposing explicitly the symbol GTE_DEV_OPENGL for conditional compilation in the applications. The initial test case is AppendConsumeBuffers which had a significant amount of explicit conditional compilation for DX11, WGL, GLX when creating the graphics engine and program factory and for loading the shader file. Made all the parameters default values for theWGLEngine and GLXEngine constructors that take the useDepth24Stencil8 inputs. This allows for generic engine construction without exposing explicitly the GTE_DEV_OPENGL or __LINUX__ preprocessor symbols. The generic definitions for DefaultEngine, DefaultProgramFactory and DefaultShaderName live in GTGraphics.h. Added an include of GTGraphics.h because any application deriving from Window3 will need a graphics engine.

GTEngine.{v12,v14,v15,v16}.{vcxproj,vcxproj.filters}
GTGraphicsShared.h
GTApplication.h
GTEngine.h
GTGraphics.h
WGLEngine.h
GLXEngine.h
Window3.h

Added the GLX-based Linux files to the project but disabled them from compiling in any Windows-based compilation. This is convenient for searching through Visual Studio and for editing GLX-based files without having to drag them from Windows Explorer into the Visual Studio IDE.

GTEngine.{v12,v14,v15,v16}.{vcxproj,vcxproj.filters}

Added makefiles with the -Werror flag set. This allows me to compile on a Linux box and have the compiler terminate as soon as a warning occurs (which I then fix). Too many warnings have been missed when the compiler does not stop.

GTEngine/makeengine_werror.gte
GTEngine/Samples/makesample_werror.gte
GTEngine/Samples/makesamples_werror.gte
GTEngine/Samples/makeallsamples_werror.gte

43 Updates to Version 3.24

April 29, 2019. MSVS 2019 default settings for cpp and h files is to use hard-coded tabs rather than spaces. I forgot to modify this to use-spaces after installing MSVS 2019. Some files had hard-coded tabs that were removed.

DisjointIntervals.h
TCBSplineCurve.h
Samples/DX11/LowLevel/LowLevel.cpp
Samples/DX11/LowLevelStream/LowLevelStream.cpp
Samples/Mathematics/IntersectConvexPolyhedra/ConvexPolyhedron.h
Samples/Mathematics/IntersectConvexPolyhedra/MTMesh.h
Samples/Mathematics/IntersectConvexPolyhedra/UnorderedSet.h
Tools/GenerateProjects/GenerateProjects.cpp
Tools/GenerateProjects/ProjectTemplate.v16.{h,cpp}

After end-to-end testing of the sample applications, I moved several projects from the Geometrics folder to the Mathematics folder but forgot to modify the mEnvironment path setting.

Samples/Mathematics/AllPairsTriangles/AllPairsTrianglesWindow.cpp
Samples/Mathematics/DistanceSegments3/DistanceSegments3.cpp
Samples/Mathematics/ShortestPath/ShortestPathWindow.cpp

44 Updates to Version 3.23

April 12, 2019. Created projects and solutions for Microsoft Visual Studio 2019.

This is a list too large to include here.

Added support for generating MSVS 2019 sample projects and solutions

Tools/GenerateProjects/GenerateProjects.cpp
ProjectTemplate.v16.{h,cpp}

Updated the comments about Microsoft Visual Studio versions. GTEngine has projects for MSVS 2013, 2015, 2017 and 2019.

GTEngineDEF.h

The main programs had a void return. No-permissive mode for C++ requires the return value to be int.

Samples/DX11/LowLevel/LowLevel.cpp
Samples/DX11/LowLevelStream/LowLevelStream.cpp

Made changes in no-permissive mode to avoid MSVS 2019 complaints.

Tools/BitmapFontCreator/BitmapFontCreator.cpp

The function GetInterval is not expected to fail, but the code is written as if it could. To avoid potentially unused variable warnings, set xmin and xmax to zero in the failure case.

DisjointIntervals.h

Removed the minIndex variable from the Refine function. It is effectively unused and was included for debugging purposes.

RiemannianGeodesic.h

Removed unused variable found in function AssignDirichletMaskBorder.

PdeFilter1.h

Modified the logic of GetKeyInfo to avoid a gcc warning about potentially uninitialized key and dt. The code is correct, but the compiler cannot determine this metaknowledge.

TCBSplineCurve.h

When the function values at the interval endpoints have the same sign, the root finder returns 0 as the number of iterations indicating it is unknown whether the interval has a root. To avoid warnings about potentially uninitialized values, the root value is set anyway (to an interval endpoint). The local variable gmin in Bisect of the line-circle distance query was there for debugging. It has been removed.

RootBisection.h
DistLine3Circle3.h

The House function declared V as a std::array but did not initialize its elements to zero, which might lead to a result that is uninitialized in all but its first element.

CubicRootsQR.h
QuarticRootsQR.h

April 10, 2019. Reorganized the sample applications so that they live in the folders suggested by the engine code they illustrate.

GTBuildAll.{v12,v14,v15}.sln

Moved to Geometrics from Mathematics.

CLODPolyline
DisjointIntervalsRectangles
GenerateMeshUVs
PolygonBooleanOperations
SplitMeshByPlane

Moved to Geometrics from Imagics.

ConformalMapping

Moved to Mathematics from Geometrics.

AllPairsTriangles
DistanceAlignedBoxes
DistanceAlignedBoxOrientedBox
DistanceOrientedBoxes
DistancePointConvexPolyhedron
DistancePointHyperellipsoid
DistanceRectangleBox
DistanceSegments3
DistanceTriangleBox
IntersectBoxCone
IntersectBoxCylinder
IntersectBoxSphere
IntersectConvexPolyhedra
IntersectSphereCone
IntersectTriangleBox
IntersectTriangles2D
ShortestPath

Ported the WM5 sample IntersectInfiniteCylinders.

GTBuildAll.{v12,v14,v15}.sln
Samples/Mathematics/IntersectInfiniteCylinders/IntersectInfiniteCylinders.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/IntersectInfiniteCylinders/IntersectInfiniteCylinders.cpp

Ported the WM5 sample ThinPlateSplines.

GTBuildAll.{v12,v14,v15}.sln
Samples/Mathematics/ThinPlateSplines/ThinPlateSplines.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/ThinPlateSplines/ThinPlateSplines.cpp

Ported the WM5 sample IntersectConvexPolyhedra. The sample has its own triangle mesh data structures, but I left them mainly intact just to get the sample running. The GTEngine version of the sample needs to be rewritten to use GTEngine mesh data structures and to merge the sample's ConvexPolyhedron class with GTEngine's ConvexPolyhedron3 class.

> GTBuildAll.{v12,v14,v15}.sln
> Samples/Mathematics/IntersectConvexPolyhedra/IntersectConvexPolyhedra.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
> Samples/Mathematics/IntersectConvexPolyhedra/IntersectConvexPolyhedra.{h,cpp}
> Samples/Mathematics/IntersectConvexPolyhedra/ConvexPolyhedron.h
> Samples/Mathematics/IntersectConvexPolyhedra/MTEdge.h
> Samples/Mathematics/IntersectConvexPolyhedra/MTMesh.h
> Samples/Mathematics/IntersectConvexPolyhedra/MTVertex.h
> Samples/Mathematics/IntersectConvexPolyhedra/MTTriangle.h
> Samples/Mathematics/IntersectConvexPolyhedra/UnorderedSet.h

Changed the mRoot object from a Node to a Spatial. This allows attaching a Visual object to the tracking device. Before this modification, you had to create a Node parent of a Visual and attach the parent to the tracking device.

> TrackingObject.{h,cpp}

April 9, 2019. Ported the WM5 sample DistancePointEllipseEllipsoid. This project had the distance queries implemented for point-ellipse, point-ellipsoid, and point-hyperellipsoid. The queries already exist in DistPointHyperellipsoid.h, so the ported project has some code that acts as a unit test to verify the query results. The project was also renamed.

> GTBuildAll.{v12,v14,v15}.sln
> GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
> Samples/Mathematics/DistancePointHyperellipsoid/DistancePointHyperellipsoid.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
> Samples/Mathematics/DistancePointHyperellipsoid/DistancePointHyperellipsoid.cpp

Refactored the WM5 sample ClodPolyline that implemented continuous level of detail of a polyline in 3 dimensions using vertex collapses. The new engine class is CLODPolyline and supports n-dimensional polylines. The sample application itself was ported but renamed.

> GTBuildAll.{v12,v14,v15}.sln
> GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
> GTMathematics.h
> CLODPolyline.h
> Samples/Geometrics/CLODPolyline/CLODPolyline.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
> Samples/Geometrics/CLODPolyline/CLODPolylineWindow.{h,cpp}

April 8, 2019. Refactored the WM5 sample ClipMesh that implemented splitting a mesh by a plane. The new engine class is SplitPlaneByMesh. The sample application itself was ported but renamed.

> GTBuildAll.{v12,v14,v15}.sln
> GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}

GTMathematics.h
SplitMeshByPlane.h
Samples/Geometrics/SplitPlaneByMesh/SplitPlaneByMesh.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Geometrics/SplitPlaneByMesh/SplitPlaneByMeshWindow.{h,cpp}

Refactored the WM5 sample Boolean2D that implemented Boolean operations on polygons in 2D using BSP trees. The new engine class is BSPPolygon2 and has nested classes that were in separate files in the WM5 sample. The sample application itself was ported but renamed.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
BSPPolygon2.h
Samples/Geometrics/PolygonBooleanOperations/PolygonBooleanOperations.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Geometrics/PolygonBooleanOperations/PolygonBooleanOperationsWindow.{h,cpp}

April 5, 2019. Ported the remaining WM5 curves/surfaces/volumes code to GTEngine. Modified the signature for Evaluate both for parametric curves and parametric surfaces. The quadric surface code uses exact rational arithmetic, and it was convenient to add constructors to BSRational that take numerators and denominators that are integers (signed or unsigned, 32-bit or 64-bit). Collapsed the two constructors for Polynomial1 into one constructor with default parameters. Reformatted all the curves/surfaces/volumes code for the new format where template function bodies occur inside the class declaration.

PolynomialCurve.h
QuadricSurface.h
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
BasisFunction.h
BezierCurve.h
BSplineCurve.h
BSplineCurveFit.h
BSplineSurface.h
BSplineSurfaceFit.h
BSplineVolume.h
BSRational.h
DarbouxFrame.h
FrenetFrame.h
IndexAttribute.h
Mesh.h
NaturalSplineCurve.h
NURBSCurve.h
NURBSSurface.h
NURBSVolume.h
ParametricCurve.h
ParametricSurface.h
Polynomial1.h
RectangleMesh.h

RectanglePatchMesh.h
RevolutionMesh.h
TCBSplineCurve.h
TubeMesh.h
VertexAttribute.h

45 Updates to Version 3.22

April 4, 2019. Fixed compiler error in permissive mode on Ubuntu 18.04, gcc 7.3.0, where a standard header file needed to be included.

MinimumAreaCircle2.h
MinimumVolumeSphere3.h

Reformatted the file by moving function bodies into the class.

BandedMatrix.h

March 26, 2019. Refactored the WM5 sample BooleanIntervalRectangle to create engine classes DisjointIntervals and DisjointRectangles that encapsulate Boolean operations on intervals $[x_0, x_1)$ and on rectangles $[x_0, x_1) \times [y_0, y_1)$. The sample application itself was ported but renamed.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
DisjointIntervals.h
DisjointRectangles.h
Samples/Mathematics/DisjointIntervalsRectangles/DisjointIntervalsRectangles.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/DisjointIntervalsRectangles/DisjointIntervalsRectangles.{h,cpp}

March 23, 2019. Ported the WM5 class BSplineReduction. Ported the sample application BSplineFitContinuous (with a renaming).

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
BSplineReduction.h
Samples/Mathematics/BSplineCurveReduction/BSplineCurveReduction.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/BSplineCurveReduction/BSplineCurveReductionWindow.{h,cpp}

March 22, 2019. Ported the WM5 class EllipsoidGeodesic. Ported the sample application GeodesicPaths (with a renaming).

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}

GTMathematics.h
EllipsoidGeodesic.h
Samples/Mathematics/GeodesicEllipsoid/GeodesicEllipsoid.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/GeodesicEllipsoid/GeodesicEllipsoidWindow.{h,cpp}

Ported the WM5 classes RiemannianGeodesic and BSplineGeodesic. Ported the sample application GeodesicHeight-Field.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
RiemannianGeodesic.h
BSplineGeodesic.h
Samples/Mathematics/GeodesicHeightField/GeodesicHeightField.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/GeodesicHeightField/GeodesicHeightFieldWindow.{h,cpp}
Samples/Mathematics/GeodesicHeightField/Data/ControlPoints.txt

Ported the WM5 class Lozenge3 and queries ContLozenge3 to GTEngine.

GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
Lozenge3.h
ContLozenge3.h

March 21, 2019. Ported the WM5 sample application GpuGaussianBlur3 to GTEngine. Added new images to the application data.

GTBuildAll.{v12,v14,v15}.sln
Samples/Imagics/GpuGaussianBlur3/GpuGaussianBlur3.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/GpuGaussianBlur3/GpuGaussianBlur3Window.{h,cpp}
Samples/Imagics/GpuGaussianBlur3/Shaders/GaussianBlur.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/BoundaryDirichlet.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/BoundaryNeumann.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/DrawImage.{hlsl,glsl}
Samples/Data/Head_U16_X128_Y128_Z64.binary
Samples/Data/Head_U16_S128x128_T8x8.binary
Samples/Data/Head_U8_S128x128_T8x8.binary

Changed the loaded 3D image to be the newly resized 3D x-ray crystallography data set.

Samples/Imagics/BSplineInterpolation/BSplineInterpolation.cpp

March 20, 2019. Ported the WM5 sample application GpuGaussianBlur2 to GTEngine.

GTBuildAll.{v12,v14,v15}.sln
Samples/Imagics/GpuGaussianBlur2/GpuGaussianBlur2.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}

Samples/Imagics/GpuGaussianBlur2/GpuGaussianBlur2Window.{h,cpp}
Samples/Imagics/GpuGaussianBlur2/Shaders/GaussianBlur.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/BoundaryDirichlet.{hlsl,glsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/BoundaryNeumann.{hlsl,glsl}

The comments in WICFileIO indicated only a couple of texture formats are supported. There are quite a few of them, so the comments were updated to reflect this.

GTEngine/Include/Applications/MSW/WICFileIO.h

March 19, 2019. Refactored the WM5 sample AdaptiveSkeletonClimbing3 to create an engine class AdaptiveSkeletonClimbing3 that encapsulates the surface extraction. The sample application itself was ported.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTImagics.h
AdaptiveSkeletonClimbing3.h
Samples/Imagics/AdaptiveSkeletonClimbing3/AdaptiveSkeletonClimbing3.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/AdaptiveSkeletonClimbing3/AdaptiveSkeletonClimbing3.cpp

Ported the WM5 sample ExtractRidges to GTEngine.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
Samples/Imagics/ExtractRidges/ExtractRidges.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/ExtractRidges/ExtractRidges.cpp

March 18, 2019. Refactored the WM5 sample AdaptiveSkeletonClimbing2 to create an engine class AdaptiveSkeletonClimbing2 that encapsulates the curve extraction. The sample application itself was ported.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTImagics.h
AdaptiveSkeletonClimbing2.h
Samples/Imagics/AdaptiveSkeletonClimbing2/AdaptiveSkeletonClimbing2.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/AdaptiveSkeletonClimbing2/AdaptiveSkeletonClimbing2.cpp

Ported the WM5 classes FastMarch, FastMarch2 and FastMarch3 to GTEngine. These classes are used for sophisticated image segmentation based on level set and fast marching methods.

GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTImagics.h
FastMarch.h
FastMarch2.h
FastMarch3.h

Ported the WM5 classes MeshCurvature and ConformalMap to GTEngine. The latter class is now named ConformalMapGenus0 to make it clear the topology of the input and output meshes are that of a sphere. The WM5 sample application for conformal mappings was also ported.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
ConformalMapGenus0.h
MeshCurvature.h
Samples/Imagics/ConformalMapping/ConformalMapping.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/ConformalMapping/ConformalMappingWindow.{h,cpp}
Samples/Data/Brain_V4098_T8192.binary

Factored out common code in Trackball and Trackcylinder to a new base class TrackObject. WM5 had the ability for the trackball to modify the local rotation of any node in a scene, but that capability had not been ported to GTEngine. This capability has now been added to the GTEngine code.

Trackball.{h,cpp}
Trackcylinder.{h,cpp}
TrackObject.{h,cpp}

March 12, 2019. Ported the WM5 PDE-based filters to GTEngine.

GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTImagics.h
CurvatureFlow2.h
CurvatureFlow3.h
FastGaussianBlur1.h
FastGaussianBlur2.h
FastGaussianBlur3.h
GaussianBlur2.h
GaussianBlur3.h
GradientAnisotropic2.h
GradientAnisotropic3.h
PdeFilter.h
PdeFilter1.h
PdeFilter2.h
PdeFilter3.h

March 11, 2019. Ported the WM5 ExtractSurfaceCubes and ExtractSurfaceTetra classes to GTE SurfaceExtractorCubes and SurfaceExtractorTetrahedra. The classes are now templated on image type that must be signed or unsigned integers using 1, 2 or 4 bytes. An abstract base class was added for sharing code, SurfaceExtractor. A new sample application was added to illustrate and test the ported code.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTImagics.h

SurfaceExtractor.h
SurfaceExtractorCubes.h
SurfaceExtractorTetrahedra.h
Samples/Imagics/ExtractLevelSurfaces/ExtractLevelSurfaces.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/ExtractLevelSurfaces/ExtractLevelSurfacesWindow.{h,cpp}
Samples/Data/Molecule_U8_X100_Y100_Z120.binary

March 8, 2019. Renaming old surface extractor to one that indicates Marching Cubes is used. This allows porting the other flavors of surface extraction from Wild Magic 5 to GTEngine.

GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTImagics.h
SurfaceExtractor.h
SurfaceExtractorMC.h
Samples/Physics/BouncingBall/DeformableBall.cpp

March 8, 2019. Ported the WM5 ExtractCurveSquares and ExtractCurveTriangles classes to GTE CurveExtractorSquares and CurveExtractorTriangles (the names to be consistent with the already existing class SurfaceExtractor). The classes are now templated on image type that must be signed or unsigned integers using 1, 2 or 4 bytes. An abstract base class was added for sharing code, CurveExtractor. A new sample application was added to illustrate and test the ported code. Also, the file organization in the Imagics folder of the Visual Studio projects was modified to add subfolders Extraction, Images and Utilities.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTImagics.h
CurveExtractor.h
CurveExtractorSquares.h
CurveExtractorTriangles.h
Samples/Imagics/ExtractLevelCurves/ExtractLevelCurves.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/ExtractLevelCurves/ExtractLevelCurvesWindow.{h,cpp}
Samples/Data/Head_U16_X256_Y256.binary
Samples/Data/Head_U16_X256_Y256.png
Samples/Data/Head_U8_X256_Y256.binary

March 7, 2019. Added specialized type trait std::is_signed for BSNumber and BSRational.

BSNumber.h
BSRational.h

March 6, 2019. Fixed compile breaks when GTE_IMAGICS_ASSERT_ON_INVALID_INDEX was enabled. The internal tests trapped problems for Image but not for Image2 and Image3; fixed those tests. Reformatted the files for the new coding rules and fixed an incorrect comment in Image.

Image.h
Image2.h
Image3.h

March 6, 2019. Converted the graphics class BoundingSphere to a template class in order to support the pending port of the Wild Magic 5 collision detection system. The ported code will be templatized and support float and double. The idea is to separate the physics code from the graphics system data structures. The class CullingPlane had to be converted to a template class because BoundingSphere uses that class.

GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
BoundingSphere.h
CullingPlane.h
Culler.{h,cpp}
Spatial.h
Visual.h
BoundingSphere.cpp
CullingPlane.cpp

March 6, 2019. Ported from Wild Magic 5 to GTEngine the test-intersection and find-intersection queries for stationary 2D triangles. Added a sample application to illustrate usage and for testing.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
IntrTriangle2Triangle2.h
Samples/Geometrics/IntersectTriangles2D/IntersectTriangles2D.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Geometrics/IntersectTriangles2D/IntersectTriangles2DWindow.{h,cpp}

46 Updates to Version 3.21

March 4, 2019. Modified the sphere-cone test-intersection query to handle four types of cones: infinite, infinite truncated, finite and frusta. Added a sample application that was used for testing. The Intersection-SphereCone.pdf document has been updated to describe the algorithms.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
Cone.h
IntrSphere3Cone3.h
Samples/Geometrics/IntersectSphereCone/IntersectSphereCone.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Geometrics/IntersectSphereCone/IntersectSphereConeWindow.{h,cpp}

The Window::OnCharPress calls are replaced by the functions Window2::OnCharPress or Window3::OnCharPress. The Window2 class does not override OnCharPress, so there is no changes in the behavior of applications that use Window2. However, the Window3 class overrides Window::OnCharPress to allow the derived application to control key-generated translation and rotation speed.

Samples/Geometrics/AllPairsTriangles/AllPairsTrianglesWindow.cpp
Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2DWindow.cpp
Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow.cpp

Samples/Geometrics/Delaunay3D/Delaunay3DWindow.cpp
Samples/Geometrics/DistanceAlignedBoxes/DistanceAlignedBoxesWindow.cpp
Samples/Geometrics/DistanceAlignedBoxOrientedBox/DistanceAlignedBoxOrientedBoxWindow.cpp
Samples/Geometrics/DistanceOrientedBoxes/DistanceOrientedBoxesWindow.cpp
Samples/Geometrics/DistancePointConvexPolyhedron/DistancePointConvexPolyhedronWindow.cpp
Samples/Geometrics/IntersectBoxCylinder/IntersectBoxCylinderWindow.cpp
Samples/Geometrics/IntersectBoxSphere/IntersectBoxSphereWindow.cpp
Samples/Geometrics/IntersectRectangleBox/IntersectRectangleBoxWindow.cpp
Samples/Geometrics/IntersectTriangleBox/IntersectTriangleBoxWindow.cpp
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2DWindow.cpp
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3DWindow.cpp
Samples/Geometrics/TriangulationCDT/TriangulationCDTWindow.cpp
Samples/Geometrics/TriangulationEC/TriangulationECWindow.cpp
Samples/Graphics/BillboardNodes/BillboardNodesWindow.cpp
Samples/Graphics/BlendedAnimations/BlendedAnimationsWindow.cpp
Samples/Graphics/BlendedTerrain/BlendedTerrainWindow.cpp
Samples/Graphics/BumpMaps/BumpMapsWindow.cpp
Samples/Graphics/Lights/LightsWindow.cpp
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargetsWindow.cpp
Samples/Graphics/TextureUpdating/TextureUpdatingWindow.cpp
Samples/Graphics/VertexCollapseMesh/VertexCollapseMeshWindow.cpp
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitterWindow.cpp
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitterWindow.cpp
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.cpp
Samples/Mathematics/Interpolation2D/Interpolation2DWindow.cpp
Samples/Physics/Cloth/ClothWindow.cpp
Samples/Physics/Fluids3D/Fluids3DWindow.cpp
Samples/Physics/MassSprings3D/MassSprings3DWindow.cpp
Samples/Physics/Rope/RopeWindow.cpp

February 14, 2019. Modified the interfaces to use Vector3<Real> rather than Vector4<Real>. This is in preparation for porting the Wild Magic 5 collision detection system to GTEngine.

BoundingSphere.{h,cpp}
Picker.cpp

February 13, 2019. Wrote a new document for clipping a convex polygon against a hyperplane. The implementation is a new file that replaces an old file for 3D only but can be used as well for 2D.

GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
IntrConvexPolygonHyperplane.h
IntrConvexPolygonPlane.h
IntrTriangle3OrientedBox3.h

February 12, 2019. Moved the experimental GTGraphicsDX12.h to an unpublished development folder. The

DX12 engine is not yet fully featured.

GTGraphicsDX12.h
GTEngine.h

February 6, 2019. Ported the Wild Magic 5 IKController, IKJoint and IKGoal classes to GTEngine. The joint and goal classes are now nested structures inside IKController that are managed by the controller itself. A sample application has been added to GTEngine.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTGraphics.h
IKController.{h,cpp}
Samples/Graphics/IKControllers/IKControllers.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/IKControllers/IKControllersWindow.{h,cpp}

February 5, 2019. Ported the Wild Magic 5 Particles and ParticleController classes to GTEngine. A sample application has been added to GTEngine.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTGraphics.h
Particles.{h,cpp}
ParticleController.{h,cpp}
Samples/Graphics/ParticleControllers/ParticleControllers.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/ParticleControllers/BloodCellController.{h,cpp}
Samples/Graphics/ParticleControllers/ParticleControllersWindow.{h,cpp}

February 3, 2019. Ported the Wild Magic 5 PointController class to GTEngine. A sample application has been added to GTEngine.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTGraphics.h
PointController.{h,cpp}
Samples/Graphics/PointControllers/PointControllers.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/PointControllers/PointControllersWindow.{h,cpp}

February 2, 2019. Ported the Wild Magic 5 MorphController class to GTEngine. A sample application has been added to GTEngine.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTGraphics.h
MorphController.{h,cpp}
Samples/Graphics/MorphControllers/MorphControllers.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/MorphControllers/MorphControllersWindow.{h,cpp}

January 31, 2019. Ported the Wild Magic 5 terrain system to GTEngine. The ported code has a Terrain class with a private nested class Page. WM5 terrain forced the user to store data in files, and internally the terrain system loaded those files. The GTEngine terrain eliminates that constraint, and now the user passes arrays of heights to initialize the terrain. This allows heights to be procedurally generated during execution. The terrain API had been simplified. A sample application has been added to GTEngine that duplicates the one in WM5.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTGraphics.h
Terrain.{h,cpp}
Samples/Graphics/Terrain/Terrain.{v12,v14,v15}.sln
Samples/Graphics/Terrain/Terrain.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/Terrain/TerrainWindow.{h,cpp}
Samples/Graphics/Terrain/TerrainEffect.{h,cpp}
Samples/Graphics/Terrain/Shaders/BaseMulDetailFogExpSqr.hlsl
Samples/Graphics/Terrain/Shaders/BaseMulDetailFogExpSqrVS.glsl
Samples/Graphics/Terrain/Shaders/BaseMulDetailFogExpSqrPS.glsl

The DX11/HLSL engine properly maps channels of a position-normal-tcoord vertex to the inputs of Texture2Effect vertex shaders (using semantics). The GL4/GLSL engine does not properly map the channels because the location for texture coordinates is listed as 1 in the GLSL code but the normal vector occurs in location 1 for PNT1 objects. Added a class to patch this and allow the GL4/GLSL version of Castle to display the textures correctly. The GL4 engine will be revised in GTL to include a better system for mapping vertex formats to GLSL shaders.

GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
Castle.{v12,v14,v15}.{vcxproj,vcxproj.filters}
CastleWindow.h
CreateMeshes.cpp
TexturePNT1Effect.{h,cpp}

After creating the index buffer in CreateRectangle, the test for a null pointer (low-probability failure to allocate the index buffer) used the vertex buffer pointer rather than the index buffer.

MeshFactory.cpp

January 22, 2019. The projects were missing the ContAlignedBox.h file.

GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}

47 Updates to Version 3.20

January 17, 2019. Added new files. Two type traits were created in GteMath.h, one is is_arbitrary_precisionto support conditional compilation via enable_if for selection between floating-point or arbitrary precision arithmetic (BSNumber, BSRational). The other is has_division_operator to support conditional compilation via enable_if for selection between BSNumber (has no divison operator) and BSRational (has division operator). The implementations for float, double and long double are in GteMath.h. The files GteBSNumber.h and GteBSRational.h have implementations for their respective classes. The file GteArbitraryPrecision.h includes all the headers to support GTEngine arbitrary precision arithmetic. The file GteQuadraticField.h supports arithmetic for numbers of the form $x + y\sqrt{d}$, where x, y and d are real or rational with $d > 0$. When rational, the class supports exact arithmetic for geometric queries involving square roots (vector normalization, roots of quadratic polynomials) until the very end when the output of the queries need floating-point-valued results.

GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
Math.h
BSNumber.h
BSRational.h
ArbitraryPrecision.h
QuadraticField.h

January 16, 2019. Added new files that implement finding the first time of contact and the corresponding contact point between a sphere and a triangle, both moving with constant linear velocity. A sample application was added for testing the code.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
GteIntrSphere3Triangle3.h
Samples/Mathematics/MovingSphereTriangle/MovingSphereTriangle.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/MovingSphereTriangle/MovingSphereTriangleWindow.{h,cpp}

January 11, 2019. The test-intersection query for two 2D line segments had a bug when the segments are on the same line. The result.intersect member was set to true always when it should have been set to iiResult.intersect.

IntrSegment2Segment2.h

48 Updates to Version 3.19

January 9, 2019. Added new files for representing nonmanifold triangle meshes.

GTLowLevel.h
GTMathematics.h
SharedPtrCompare.h
WeakPtrCompare.h
ETNonmanifoldMesh.{h,cpp}
VETNonmanifoldMesh.{h,cpp}

December 11, 2018. Quaternion<Real> was derived from Vector<4, Real> in order to share the unary and binary operators for linear algebraic operations and to share Dot, Length and Normalize for geometric operations.

Unfortunately, implicit conversions hide the fact that there is a conflict between quaternion multiplication Quaternion<Real>::operator* and multiplication operators Vector<4, Real>::operator*. Similar conflicts exist for the negation unary operator. The vector-based operations return Vector<4, Real> objects, and the multiplications resolve to the component-wise multiplication of vectors instead of the group-algebraic multiplication of quaternions. Expressions such as 2.0 * q0 * q1 and -q0 * q1 do not compile to the mathematical expressions one expects. For example, 2.0 * q0 * q1 is parsed as (2.0 * q0) * q1. The product 2.0 * q0 is compiled to use the vector-scalar multiplication (which is what we want) and returns a vector object, say v. The compiler then has to generate code for v * q1, which unfortunately resolves to the component-wise vector product rather than the quaternion product. Also for example, -q0 * q1 is parsed so that -q0 is computed by the vector unary negation and returns a vector object, say v. Once again the compiler uses the component-wise vector product for v * q1 rather than quaternion multiplication. As a result, I removed the dependency of Quaternion<Real> on Vector<4, Real> as a base class. Added unary and binary operators for linear algebraic operations. Added Dot, Length and Normalize functions for geometric operations.

Quaternion.h

Reformatted the files, revised the SLERP comments and added SlerpRPH to the Quaternion class because the SLERP estimation class has a function to estimate SlerpRPH.

Quaternion.h
SlerpEstimate.h

Removed unused variables. The compiler complaints about unused variables occurred only after the Quaternion changes. The constructor for the quaternion class was modified from an empty body to Quaternion() = default, so my guess is that this change caused the compiler to realize that the variables were unused.

Samples/Geometrics/DistanceAlignedBoxes/DistanceAlignedBoxesWindow.cpp
Samples/Geometrics/DistanceAlignedBoxOrientedBox/DistanceAlignedBoxOrientedBoxWindow.cpp
Samples/Geometrics/DistanceOrientedBoxes/DistanceOrientedBoxesWindow.cpp
Samples/Geometrics/DistancePointConvexPolyhedron/DistancePointConvexPolyhedronWindow.cpp
Samples/Geometrics/DistanceRectangleBox/DistanceRectangleBoxWindow.cpp
Samples/Geometrics/DistanceTriangleBox/DistanceTriangleBoxWindow.cpp
Samples/Geometrics/IntersectBoxCone/IntersectBoxConeWindow.cpp
Samples/Geometrics/IntersectBoxCylinder/IntersectBoxCylinderWindow.cpp
Samples/Geometrics/IntersectBoxSphere/IntersectBoxSphereWindow.cpp

December 6, 2018. Reformatted the file in preparation for porting to GTL.

Polynomial1.h

49 Updates to Version 3.18

November 30, 2018. Added test for point-in-cone (in any dimension 2 or larger).

ContCone.h
GTMathematics.h

November 30, 2018. In the Samples folder, renamed the top-level folder CSharpCppManaged to CSharpCppManaged_MSVS2017. Renamed the project files and filter files with appropriate modifications to the XML to include the v14 version number of MSVS 2017. Added similar wrappers for MSVS 2013 in the top-level folder CSharpCppManaged_MSVS2013 and in the top-level folder CSharpCppManaged_MSVS2015 for MSVS 2015.

November 29, 2018. Extended the test-intersection queries to support aligned boxes as well as oriented boxes and to support cone frusta as well as infinite cones. Modified the sample application to test all cases. The Cone<N,Real> class was modified to contain minHeight and maxHeight members (the previous version had only a maximum height with the assumption that the minimum height is zero).

Cone3.h
IntrAlignedBox3Cone3.h
IntrOrientedBox3Cone3.h
IntrLine3Cone3.h
Samples/Geometrics/IntersectBoxCone/IntersectBoxConeWindow.{h,cpp}

Reformatted the files so the member functions occur within the header. Added comments that refer to a new PDF document which contains the algorithm details for test-intersection and find-intersection queries between linear components and boxes.

IntrLine3AlignedBox3.h
IntrLine3OrientedBox3.h
IntrRay3AlignedBox3.h
IntrRay3OrientedBox3.h
IntrSegment3AlignedBox3.h
IntrSegment3OrientedBox3.h

50 Updates to Version 3.17

October 30, 2018. I used to have sections ordered alphabetically by the names of the subfolders described in the comments. Two phase name lookups for template matching by the MSVS 2017 compiler had no problems with the ordering, but the Linux g++ compiler does. This occurred when trying to match std::sqrt(...) and other math functions when the inputs are based on BSNumber or BSRational. The *Arithmetic* section has been moved before all other headers, and the UInteger* files have been moved before the BS* files.

GTMathematics.h

October 30, 2018. Modified the cone fitting code to allow the user to specify whether the cone input is the initial guess or whether to compute the cone initial guess internally. The least-squares fitting document has been updated to describe the algorithm for computing the initial guess. A new sample application has been added to illustrate.

ApprCone3.h
GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
Samples/Mathematics/FitCone/FitCone.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/FitCone/FitConeWindow.{h,cpp}

October 30, 2018. Modified the torus fitting code to allow the user to specify whether the torus input is the initial guess or whether to compute the torus initial guess internally. A new sample application has been added to illustrate. Linux g++ on Ubuntu 16.04 complained again about ambiguous lookup of std::sqrt when the input is BSRational. I had to modify the order of the header includes in GTMathematics.h to avoid this problem when compiling FitTorusWindow. It appears two-phase name lookup for template matching has the unpleasant consequence of forcing order dependence of header files.

ApprTorus3.h
GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
Samples/Mathematics/FitTorus/FitTorus.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/FitTorus/FitTorusWindow.{h,cpp}

October 28, 2018. When using ComputeType of float or double, floating-point rounding errors can cause the UpdateSupport functions to fail to find a bounding circle/sphere. There was a trap in the code for this failure, but the handler was choosing the bounding sphere from an array of spheres computed in the function, but it used an out-of-range-index. Now the UpdateSupport functions report an error condition. When the top-level logic for point processing finds an error, it computes a bounding circle/sphere using the algorithm in GetContainer of GteContCircle2.h or GteContSphere3.h. This circle/sphere is generally not minimal. The operator() reports success or failure. On failure you can always try to call the function again because the random shuffle is most likely to be different from the previous one. A change in ordering of points might lead to a successful construction. To be certain that you will always obtain the minimal bound, you really need to use an exact rational type for ComputeType.

MinimumAreaCircle2.h
MinimumAreaSphere3.h

October 22, 2018. Added implementations for NURBS curves that represent quarter, half or full circles and NURBS surfaces that represent eighth, half or full spheres. Added a sample applications to illustrate. Reformatted the base NURBS classes and replace the arrays BasisFunctionInput by individual items in order to allow the derived classes to construct the base classes.

GTBuildAll.{v12,v14,v15}.sln
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
NURBSCurve.h
NURBSSurface.h
NURBSVolume.h
NURBSCircle.h
NURBSSphere.h
Samples/Mathematics/NURBSCircle/NURBSCircle.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/NURBSCircle/NURBSCircleWindow.{h,cpp} Samples/Mathematics/NURBSSphere/NURBSSphere.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/NURBSSphere/NURBSSphereWindow.{h,cpp}

October 26, 2018. Factored the $G(W)$ function so that the summations involving only sample point components can be precomputed. This leads to a massive performance increase when evaluating G over a dense set of vectors W on the unit hemisphere.

ApprCylinder3.h

Added member functions for drawing thick lines, rectangles, circles and ellipses.

Window2.{h,cpp}

October 20, 2018. DXEngine tries to create a device for feature levels in the order 11.1, 11.0, 10.1, 10.0, 9.3, 9.2 and 9.1. The member featureLevel of the DX11 version of Window::Parameter was intended to be the least capable feature level attempted during the creation. The default was 11.0, which means if you have a graphics card that supports DX10 but not DX11, the device creation fails and the application aborts (with a LogError dialog indicating the failure). To execute with DX10, you needed to set the featureLevel to something less capable than 11.0. I had modified the device creation in DXEngine some time ago to create a device by iterating over the feature levels, and if one is found it sets the shader model accordingly for the D3DCompile call during shader compilation. Because of this design for device creation, there is no need for the featureLevel member or for clamping the search to a limited number of feature levels. This member and the support that uses it has been removed.

Include/Applications/MSW/DX11/GteWindow.h
Source/Applications/MSW/DX11/GteWindow.cpp
Source/Applications/MSW/DX11/GteWindowSystem.cpp
Include/Applications/Graphics/DX11/GteDX11Engine.h
Source/Applications/Graphics/DX11/GteDX11Engine.cpp
Samples/Basics/AppendConsumeBuffers/AppendConsumeBuffers.cpp
Samples/Mathematics/PartialSums/PartialSums.cpp
Samples/Mathematics/RootFinding/RootFinding.cpp
Samples/DX11/RawBuffers/RawBuffers.cpp

51 Updates to Version 3.16

October 19, 2018. With /permissive- in Microsoft Visual Studio 2017 (and with no-permissive on Linux), the two-phase name lookup for template functions was failing because of the order of the header file inclusion. The order must be GteUIntegerAP32.h, GteBSRational.h and GteRootsPolynomial.h in order for the compiler to find a match for std::sqrt and other math function calls in RootsPolynomial that take inputs from BSRational.

ApprTorus3.h

October 17, 2018. An MSVS 2017 internal compiler error occurs for the SCPolynomial member function definitions when the /permissive- option is used. I moved the definitions to the point of declaration in the subclass to eliminate the error. I also set the /permissive- option in the engine project (only for MSVS 2017, the previous compilers do not recognize the option).

GTEngine.v15.vcxproj
DistCircle3Circle3.h

October 16, 2018. The GetContainer functions did not initialize the radius of the output circle or sphere. The default constructors set the radius to 1, so if the points are all a distance less than 1 from their average, the bounding circle or sphere is larger than need be. I added to GetContainer initialization of the radius to 0.

ContCircle2.h
ContSphere3.h

Added more comments to the file to describe the differences between representation of vectors by a basis, representation of points by an affine basis and Cartesian coordinates of points.

ConvertCoordinates.h

October 9, 2018. Renamed some variables to use the same names as in the PDF that describes the noniterative solver. Added more comments to explain the cryptic block of code in the if (norm > (Real)0) clause.

SymmetricEigensolver3x3.h

October 5, 2018. Eliminated the Function$< T >$ classes. Eliminated the header file GteConstants.h file, moving the contents into the new file GteMath.h. The new file also contains implementations for some convenient mathematics functions not found in cmath, all implemented in the std namespace. The Function<T> implementations that were in the deleted file are now in GteBSNumber.h, GteBSRational.h and GteIEEEBinary16.h but implemented within the std namespace. This breaks the dependencies of many of the geometric queries on Function$< T >$ so that if you do not use exact precision or 16-bit floating-point arithmetic, you are not forced to use the Function<T> wrapper. The namespace qualifier std:: was added to any math function call. For example, sin was replaced by std::sin.

GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
GteMath.h
GteConstants.h
GteFunctions.h
GteFunctionsBSNumber.h
GteFunctionsBSRational.h
GteFunctionsIEEEBinary16.h
ACosEstimate.h
ApprCone3.h
ApprCylinder3.h
ApprEllipse2.h
ApprEllipseByArcs.h
ApprEllipsoid3.h
ApprGreatCircle3.h
ApprQuadratic2.h
ApprQuadratic3.h
ApprTorus3.h
ATanEstimate.h
BandedMatrix.h

BasisFunction.h
BSNumber.h
BSRational.h
ChebyshevRatio.h
CholeskyDecomposition.h
Cone.h
ContCapsule3.h
ContCircle2.h
ContCylinder3.h
ContEllipse2.h
ContEllipsoid3.h
ContEllipsoid3MinCR.h
ContSphere3.h
CosEstimate.h
CubicRootsQR.h
DarbouxFrame.h
DCPQuery.h
DistAlignedBox3OrientedBox3.h
DistAlignedBoxAlignedBox.h
DistCircle3Circle3.h
DistLine3AlignedBox3.h
DistLine3Circle3.h
DistLineLine.h
DistLineRay.h
DistLineSegment.h
DistOrientedBox3OrientedBox3.h
DistPoint3Circle3.h
DistPoint3ConvexPolyhedron3.h
DistPoint3Cylinder3.h
DistPoint3Frustum3.h
DistPoint3Rectangle3.h
DistPoint3Tetrahedron3.h
DistPointAlignedBox.h
DistPointHyperellipsoid.h
DistPointLine.h
DistPointRay.h
DistPointSegment.h
DistPointTriangle.h
DistRayRay.h
DistRaySegment.h
DistRectangle3AlignedBox3.h
DistRectangle3OrientedBox3.h
DistSegmentSegment.h
DistTriangle3AlignedBox3.h
DistTriangle3OrientedBox3.h
Exp2Estimate.h
ExtremalQuery3BSP.h
FIQuery.h

FrenetFrame.h
GenerateMeshUV.h
GMatrix.h
GVector.h
Hyperellipsoid.h
IEEEBinary16.h
IntpBSplineUniform.h
IntpSphere2.h
IntpThinPlateSpline2.h
IntpThinPlateSpline3.h
IntrAlignedBox2Circle2.h
IntrAlignedBox3Sphere3.h
IntrCircle2Circle2.h
IntrDisk2Sector2.h
IntrEllipse2Ellipse2.h
IntrEllipsoid3Ellipsoid3.h
IntrHalfspace3Cylinder3.h
IntrHalfspace3Ellipsoid3.h
IntrLine2Circle2.h
IntrLine3Capsule3.h
IntrLine3Cone3.h
IntrLine3Cylinder3.h
IntrLine3Ellipsoid3.h
IntrLine3Sphere3.h
IntrPlane3Circle3.h
IntrPlane3Cylinder3.h
IntrPlane3Ellipsoid3.h
IntrPlane3Sphere3.h
IntrSphere3Cone3.h
IntrSphere3Sphere3.h
InvSqrtEstimate.h
LinearSystem.h
Log2Estimate.h
Math.h
Matrix.h
Matrix2x2.h
Mesh.h
MinimumAreaBox2.h
MinimumAreaCircle2.h
MinimumVolumeBox3.h
MinimumVolumeSphere3.h
Projection.h
QuarticRootsQR.h
RevolutionMesh.h
RootsPolynomial.h
Rotation.h
Sector2.h
SinEstimate.h

SingularValueDecomposition.h
SqrtEstimate.h
SurfaceExtractor.h
SymmetricEigensolver.h
SymmetricEigensolver2x2.h
SymmetricEigensolver3x3.h
TanEstimate.h
TIQuery.h
Torus3.h
TubeMesh.h
UnsymmetricEigenvalues.h
Vector.h
ViewVolume.cpp
BlendTransformController.cpp
Controller.cpp
MeshFactory.cpp
CullingPlane.cpp
Culler.cpp
Trackball.cpp
BoundingSphere.cpp
BillboardNode.cpp
Lighting.cpp
Trackcylinder.cpp
ImageUtility2.cpp
Timer.cpp
IEEEBinary16.cpp
IntelSSE.cpp
Samples/Geometrics/DistanceSegments3/DistanceSegments3.cpp
Samples/Geometrics/IntersectBoxCone/IntersectBoxConeWindow.cpp
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.cpp
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.cpp
Samples/Geometrics/TriangulationCDT/TriangulationCDTWindow.cpp
Samples/Graphics/BspNodes/BspNodesWindow.cpp
Samples/Graphics/Castle/CastleWindow.cpp
Samples/Graphics/Lights/LightsWindow.cpp
Samples/Imagics/Convolution/ConvolutionWindow.cpp
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp
Samples/Mathematics/ApproximateEllipsesByArcs/ApproximateEllipsesByArcsWindow.cpp
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitterWindow.cpp
Samples/Mathematics/FitCylinder/FitCylinderWindow.cpp
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.cpp
Samples/Mathematics/MovingCircleRectangle/MovingCircleRectangleWindow.cpp
Samples/Mathematics/MovingSphereBox/MovingSphereBoxWindow.cpp
Samples/Mathematics/NURBSCurveExample/NURBSCurveExampleWindow.cpp
Samples/Physics//BallHill/BallHillWindow.cpp
Samples/Physics/BallRubberBand/PhysicsModule.cpp
Samples/Physics/BouncingBall/BouncingBallWindow.cpp
Samples/Physics/BouncingBall/PhysicsModule.cpp

Samples/Physics/Cloth/PhysicsModule.cpp
Samples/Physics/DoublePendulum/PhysicsModule.cpp
Samples/Physics/FlowingSkirt/FlowingSkirtWindow.cpp
Samples/Physics/FoucaultPendulum/FoucaultPendulumWindow.cpp
Samples/Physics/FoucaultPendulum/PhysicsModule.cpp
Samples/Physics/FreeTopFixedTip/FreeTopFixedTipWindow.cpp
Samples/Physics/FreeTopFixedTip/PhysicsModule.cpp
Samples/Physics/HelixTubeSurface/HelixTubeSurfaceWindow.cpp
Samples/Physics/KeplerPolarForm/KeplerPolarFormWindow.cpp
Samples/Physics/KeplerPolarForm/PhysicsModule.{h,cpp}
Samples/Physics/MassPulleySpringSystem/MassPulleySpringSystemWindow.cpp
Samples/Physics/MassPulleySpringSystem/PhysicsModule.cpp
Samples/Physics/SimplePendulum/SimplePendulum.cpp
Samples/Physics/SimplePendulumFriction/SimplePendulumFrictionWindow.cpp
Samples/Physics/SimplePendulumFriction/PhysicsModule.cpp
Tools/GenerateApproximations/FitASin.h
Tools/GenerateApproximations/FitATan.h
Tools/GenerateApproximations/FitCos.h
Tools/GenerateApproximations/FitExp2.h
Tools/GenerateApproximations/FitInvSqrt.h
Tools/GenerateApproximations/FitLog2.h
Tools/GenerateApproximations/FitReciprocal.h
Tools/GenerateApproximations/FitSin.h
Tools/GenerateApproximations/FitSqrt.h
Tools/GenerateApproximations/FitTan.h

October 4, 2018. Replaced fabs and Function<T>::FAbs by std::abs. This is in preparation for eliminating the Function class, implementing std::somefunction in BSNumber and BSRational as needed. The idea is not to force users to include GteFunctions.h, instead just imposing a policy that Real in template classes must have various functions implemented for the template to compile and run.

ApprEllipseByArcs.h
ApprTorus3.h
Functions.h
FunctionsIEEEBinary16.h
GVector.h
IntrEllipse2Ellipse2.h
IntrConvexPolygonPlane.h
IntrOrientedBox2Sector2.h
IntrSegment2AlignedBox2.h
IntrTriangle3OrientedBox3.h
IntpAkimaUniform2.h
Polygon2.h
Polyhedron3.h
SymmetricEigensolver3x3.h
VertexCollapseMesh.h
Vector.h
Vector2.h

Vector3.h
Vector4.h
BlendTransformController.cpp
BoundingSphere.cpp
ViewVolume.cpp
Transform.cpp
Samples/Graphics/BumpMaps/SimpleBumpMapEffect.cpp
Samples/Graphics/Castle/CastleWindow.cpp
Samples/Physics/BouncingBall/DeformableBall.cpp
Samples/Physics/KeplerPolarForm/PhysicsModule.cpp
Samples/Physics/FlowingSkirt/FlowingSkirtWindow.cpp
Samples/Physics/FoucaultPendulum/PhysicsModule.cpp
Samples/Physics/FreeTopFixedTip/PhysicsModule.cpp

October 4, 2018. LLVM introduced the identifier _Float16 in apple-clang 10.0. I had defined this as an interal type in IEEEBinary16. The naming rules for the compiler state that identifiers starting with one or two underscores followed by a capital letter are reserved names. Renamed mine to GTEFloat16.

52 Updates to Version 3.15

October 3, 2018. Added an implementation of the intersection query between a sphere and a box (considered as solids), each moving with a constant linear velocity. A PDF describing the algorithm is at the website, Interesection of Moving Sphere and Box. A sample application has been added to serve as a unit tests and to show how the code is used. The oriented-box code was modified to share the query code for axis-aligned boxes.

GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTBuildAll.{v12,v14,v15}.sln
IntrAlignedBox2Circle2.h
IntrAlignedBox3Sphere3.h
IntrOrientedBox3Sphere3.h
Samples/Mathematics/MovingSphereBox/MovingSphereBox.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/MovingSphereBox/MovingSphereBoxWindow.{h,cpp}

October 3, 2018. The Debug-build frame rate for BlendedAnimations is low, making it difficult to debug. The problem is that the Debug builds do not inline functions, so the operator[] function calls of Vector4, Matrix4x4, std::array and std::vector are slow when called billions of times as they are in the inner loop of the vertex-bone double loop. In a Release build, the matrix-vector multiplies in the inner loop are the bottleneck, taking about 25% of CPU time. To improve performance (both for Debug and Release), I replaced Array2 members by std::vector. To avoid performance hit with the lock() called on the std::weak_ptr bones in the inner loop, I store the world transforms in a std::vector container before executing the double loop. Finally, I use typecasting to raw float pointers to avoid the no-inline problem and to bypass all the Vector4, Matrix4x4, std::array and std::vector framework. On my Intel i7-6700 3.40 GHz machine, the Debug build before the changes runs at 16 fps and after the changes runs at 223 fps. The Release build before the changes runs at 2390 fps (sync to vertical retrace disabled) and after the changes runs at 4170 fps. I also tried to partition the data and added multithreading (via std::thread), but the overhead for a simple approach (launch all threads, wait for completion via join) was significant enough that the multithreaded approach performed worse than the single-threaded approach.

SkinController.{h,cpp}
Samples/Graphics/BlendedAnimations/BipedManager.cpp

September 27, 2018. The texture coordinate at the south pole of the sphere generated by CreateSphere was $(1/2, 1/2)$ but needed to be $(1/2, 0)$. to be consistent with the texture coordinate assignments at other sphere locations.

MeshFactory.cpp

When enabling and disabling the OpenGL state when independent blending is active, the glEnable(GL_BLEND) and glDisable(GL_BLEND) needed to be glEnablei(GL_BLEND, i) and glDisablei(GL_BLEND, i).

GL4BlendState.cpp

September 26, 2018. Removed the include of Functions.h from classes implementing the distance queries via DCPQuery.h because the latter file already includes it. Added the include of Functions.h to TIQuery.h and FIQuery.h because it is invariably used by classes implementing intersection queries.

TIQuery.h
FIQuery.h
DistAlignedBoxAlignedBox.h
DistAlignedBox3OrientedBox3.h
DistCircle3Circle3.h
DistLine3Circle3.h
DistOrientedBox3OrientedBox3.h
DistPointHyperellipsoid.h
DistPoint3Circle3.h
DistRectangle3AlignedBox3.h
DistRectangle3OrientedBox3.h
DistTriangle3AlignedBox3.h
DistTriangle3OrientedBox3.h
IntrEllipse2Ellipse2.h

September 21, 2018. Made some minor changes to InRegionEdgeOverlap to be consistent with the new moving sphere-box find-intersection code.

IntrAlignedBox2Circle2.h

September 18, 2018. The FindVertexRegionIntersection has a block of code that calls GetVertexIntersection. The latter function returns the contact time when the sphere intersects a vertex of the box. However, the contact point was incorrectly computed to be the line-sphere intersection for the line of motion and the octant of a sphere at the vertex of the box (which is part of the Minkowski sum of box and sphere). The contact point is simply the vector of extents.

IntrAlignedBox3Sphere3.h

The FIQuery::operator() method needed to transform the contact point back to the original coordinate system.

IntrOrientedBox3Sphere3.h

Added a new sample application for testing the find-intersection queries between a sphere and a box (aligned or oriented), both moving with constant linear velocities.

GTBuildAll.{v12,v14,v15}.sln
Samples/Mathematics/MovingSphereBox/MovingSphereBox.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/MovingSphereBox/MovingSphereBoxWindow.{h,cpp}

53 Updates to Version 3.14

September 8, 2018. The main function enables a LogReporter for which the message-box logging is disabled. This was necessary because of an unexpected error generated by glUniformi for OpenGL builds. The error only occurs on the first drawing pass, after which the scene is drawn correctly. At one time I modified the order of the LogReporter parameters, which causes the message-box error to occur. Fixed this.

Samples/Graphics/CubeMaps/CubeMapsWindow.cpp

Added a pvw-matrix constant buffer to VisualEffect including set-get accessors. The derived classes all declared their own buffers and duplicated the interfaces. The design of VisualEffect is to support effects that get attached to Visual objects, so the common case is that these effects will require a pvw-matrix that combines the projection and view matrices of a camera and the world matrix of the Visual object.

VisualEffect.{h,cpp}
ConstantColorEffect.{h,cpp}
LightingEffect.{h,cpp}
Texture2Effect.{h,cpp}
Texture3Effect.{h,cpp}
VertexColorEffect.{h,cpp}
DirectionalLightEffect.cpp
DirectionalLightTextureEffect.cpp
PointLightEffect.cpp
PointLightTextureEffect.cpp
SpotLightEffect.cpp
Samples/Graphics/AreaLights/AreaLightEffect.{h,cpp}
Samples/Graphics/BlendedTerrain/BlendedTerrainEffect.{h,cpp}
Samples/Graphics/BumpMaps/SimpleBumpMapEffect.{h,cpp}
Samples/Graphics/CubeMaps/CubeMapEffect.{h,cpp}
Samples/Graphics/GlossMaps/GlossMapEffect.{h,cpp}
Samples/Graphics/ProjectedTextures/ProjectedTextureEffect.{h,cpp}
Samples/Graphics/SphereMaps/SphereMapEffect.{h,cpp}
Samples/Graphics/VertexTextures/DisplacementEffect.{h,cpp}

The VisualEffect modifications are motivated by a feature request to use a spatial hierarchy update pass to determine potentially visible sets of Visual objects and have their pvw-matrices updated. Currently, if you use PVWUpdater to update the GPU memory corresponding to the CPU memory that stores the pvw matrices, you have to dynamically subscribe the matrix-cbuffer pairs for the potentially visible Visual objects to avoids synchronizing the GPU memory for all matrix-cbuffer pairs. The design of PVWUpdater did not include information about whether a matrix-cbuffer pair can be rejected because of nonvisibility. Now PVWUpdater has an update function whose input is the potentially visible set obtained from a Culler object. The BspNodes sample application was modified to illustrate how to use the new feature of PVWUpdater.

PVWUpdater.{h,cpp}
Samples/Graphics/BspNodes/BspNodesWindow.{h,cpp}

August 20, 2018. The eigenvector matrix type used an incorrect formula. Replaced mIsRotation and function bool IsRotation() by mEigenvectorMatrixType and int GetEigenvectorMatrixType(). Modified comments to GetEigenvectors to make it clear that the 1-dimensional array is filled in to represent an $n \times n$ matrix that is stored in row-major order and whose columns are the eigenvectors.

SymmetricEigensolver.h
Hyperellipsoid.h

July 31, 2018. Added an implementation of the intersection query between a circle and a rectangle (considered as solids), each moving with a constant linear velocity. A PDF describing the algorithm is at the website, Interesection of Moving Circle and Rectangle. A sample application has been added to serve as a unit tests and to show how the code is used. The oriented-box code was modified to share the query code for axis-aligned boxes.

GTMathematics.h
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTBuildAll.{v12,v14,v15}.sln
IntrOrientedBox2Circle2.h
IntrAlignedBox2Circle2.h
Samples/Mathematics/MovingCircleRectangle/MovingCircleRectangle.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/MovingCircleRectangle/MovingCircleRectangleWindow.{h,cpp}

July 25, 2018. Added an implementation of a virtual track cylinder.

GTGraphics.h
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
Trackcylinder.{h,cpp}

54 Updates to Version 3.13

July 18, 2018. Added implementations for Cholesky decomposition of symmetric matrices, for minimization using the Gauss-Newton algorithm and for minimization using the Levenberg-Marquardt algorithm. Added a least-squares algorithm for fitting a cone to a set of points in 3D. Added a least-squares algorithm for fitting a torus to a set of points in 3D.

GTMathematics.h
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTBuildAll.{v12,v14,v15}.sln
CholeskyDecomposition.h
GaussNewtonMinimizer.h
LevenbergMarquardtMinimizer.h
ApprCone3.h
ApprTorus3.h

Fixed compiler warnings in MSVS 2013 about not being able to automatically generated the assignment operator. One of the base class members is tagged with const. Added the delete modified for copy constructors and copy assignments and for move constructors and move assignments. Also fixed a compiler warning about a signed-unsigned mismatch when in a 32-bit build.

IntpBSplineUniform.h

Fixed a bug where the extern declaration of wglSwapIntervalEXT was different from that of GteOpenGL.h. This bug was introduced when I got GTEngine compiling under a MinGW environment.

Graphics/GL4/WGL/GteWGLEngine.cpp

Added libEGL to the linker line to support headless rendering.

GeometricTools/GTEngine/Samples/makesample.gte

July 13, 2018. Fixed a bug in the Divide algorithm when a constant divisor is used. Added MakeMonic to divide through by the leading coefficient, after which the highest-degree term is 1. Added GreatestCommon-Divisor and SquareFreeFactorization. The latter provides a list of polynomials, each having simple real-valued roots. The factorization is a preprocessing step for using Vincent's Theorem for computing root-bounding intervals for the real-valued roots.

Polynomial1.h

July 2, 2018. The picking system is based on intersection of lines, rays or segments with triangle primitives (triangle meshes, triangle strips). However, it does not correctly handle triangle primitives that have adjacency information. Added support for triangle meshes with adjacency. The other types will be implemented at a later date.

IndexBuffer.cpp

June 22, 2018. The result.intersect was uninitialized when the segment intersects the circle of the arc but the intersection points are not on the arc. Fixed this to ensure the result.intersect is set in all cases.

IntrSegment2Arc2.h

June 21, 2018. Added support for selecting at run time whether to use GLX or EGL for function pointer lookups on Linux distributions. If you want EGL, you must set the global variable gUseEGLGetProcAddress to true (default value is false) before calling InitializeOpenGL. In particular, you should do so before creating a GL4Engine object.

GTEngine/Source/Graphics/GL4/GLX/GteGLXExtensions.cpp
GTEngine/Tools/GenerateOpenGLWrapper/GteOpenGL.h

55 Updates to Version 3.12

June 7, 2018. Added implementations for B-spline interpolation of data defined on lattices in any dimensions Wild Magic 5 had implementations based on the BSplineInterpolation.pdf document, but that is quite old and has a translation bias. The document has been rewritten and the code in GTEngine is an implementation of the algorithm presented in the revised document. Also added a new sample application to illustrate how to use the interpolators and to act as unit tests.

IntpBSplineUniform.h
Samples/Imagics/BSplineInterpolation/BSplineInterpolation.{v12,v14,v15}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/BSplineInterpolation/BSplineInterpolation.cpp
Data/Molecule97x97x116.png
GTMathematics.h
Polynomial1.h
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTBuildAll.{v12,v14,v15}.sln

May 22, 2018. Added default constructors. Removed the default parameters from the current constructors and added the explicit modifier.

EdgeKey.{h,cpp}
TriangleKey.{h,cpp}
TetrahedronKey.{h,cpp}

With the old code, C++ rule *13.3.1.7 Initialization by list-initialization* allows the following code.

```
std::set<EdgeKey<false>> keys;
int i0 = 1, i1 = 2;

// The initializer list is implicitly converted by choosing its first
// element as the argument to the EdgeKey constructor and then using
// the default parameter of -1 for the second argument, which is not
// the desired behavior.
keys.insert({ i0, i1 });  // acts like EdgeKey<false>(1,-1)

// If you add a default constructor and remove the default parameters
// for the current constructor, this code compiles because of the match
// rule 13.3.1.7.
keys.insert({ i0, i1 });  // acts like EdgeKeys<false>(1, 2);

// If the non-default constructor additionally is tagged as explicit,
// this code fails to compile.
```

```
keys.insert({ i0, i1 });  // fails to compile

// The constructor call with a 2-element initializer list still succeeds
// however.
EdgeKey<false> someKey{ i0, i1 };  // acts like EdgeKey<false>(i0, i1)
```

May 19, 2018. Modified the comments for defining a torus.

Torus3.h

May 15, 2018. Added a new folder that has a simple example for how to call GTEngine C++ code from C# code using a managed wrapper (C++/CLI). The example computes the mininum-volume bounding box for a set of points. You should run this in a Release build for speed (it is slow in a Debug build). The configuration manager preference is set for x64, so if you have only a 32-bit machine, you will want to change the configuration for x86. The framework is based on the nicely written article Creating a C++/CLI Wrapper by Mircea Oprea. If you read this article, the native library originally was created as a dynamic library, but that has an overstrike and is replaced by a static library. This is a consequence of the discussion at the end of the article where Maxence points out that this avoids DLL export problems. However, as Maxence says, you can add `__declspec(dllexport)` tags to the functions you want exported so that the DLL is actually generated by the native library (in this case, the CppLibrary).

GeometricTools/GTEngine/Samples/CSharpCppManaged
GeometricTools/GTEngine/Samples/CSharpCppManaged/CppLibrary
GeometricTools/GTEngine/Samples/CSharpCppManaged/CppLibrary/CppLibrary.h
GeometricTools/GTEngine/Samples/CSharpCppManaged/CppLibrary/CppLibrary.vcxproj
GeometricTools/GTEngine/Samples/CSharpCppManaged/CppLibrary/CppLibrary.vcxproj.filters
GeometricTools/GTEngine/Samples/CSharpCppManaged/CppLibrary/MinimumVolumeBox.cpp
GeometricTools/GTEngine/Samples/CSharpCppManaged/CppLibrary/MinimumVolumeBox.h
GeometricTools/GTEngine/Samples/CSharpCppManaged/ManagedLibrary
GeometricTools/GTEngine/Samples/CSharpCppManaged/ManagedLibrary/AssemblyInfo.cpp
GeometricTools/GTEngine/Samples/CSharpCppManaged/ManagedLibrary/ManagedLibrary.vcxproj
GeometricTools/GTEngine/Samples/CSharpCppManaged/ManagedLibrary/ManagedLibrary.vcxproj.filters
GeometricTools/GTEngine/Samples/CSharpCppManaged/ManagedLibrary/ManagedObject.h
GeometricTools/GTEngine/Samples/CSharpCppManaged/ManagedLibrary/MinimumVolumeBox.cpp
GeometricTools/GTEngine/Samples/CSharpCppManaged/ManagedLibrary/MinimumVolumeBox.h
GeometricTools/GTEngine/Samples/CSharpCppManaged/CSharpApplication
GeometricTools/GTEngine/Samples/CSharpCppManaged/CSharpApplication/Properties
GeometricTools/GTEngine/Samples/CSharpCppManaged/CSharpApplication/Properties/AssemblyInfo.cs
GeometricTools/GTEngine/Samples/CSharpCppManaged/CSharpApplication/App.config
GeometricTools/GTEngine/Samples/CSharpCppManaged/CSharpApplication/CSharpApplication.csproj
GeometricTools/GTEngine/Samples/CSharpCppManaged/CSharpApplication/Program.cs

April 28, 2018. Fixed a typographical error in the comments. For open uniform splines, the number of unique knots is $s = n - d + 1$, not $s = n - d - 1$. Removed the comment about the *floating uniform* definition not occuring in the book by Riesenfeld et al. In fact the definition occurred on pages where I had not expected it.

BasisFunction.h

April 15, 2018. Added a new file for the containment queries for axis-aligned boxes.

ContAlignedBox.h
GTMathematics.h

April 2, 2018. Added new files to query a CPU for SIMD capabilities and related support.

Mathematics/MSW/GteCPUQueryInstructions.h
Source/MSW/GteCPUQueryInstructions.cpp
GTMathematics.h
GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}

March 11, 2018. Refactored the application and window system so that the code compiles with MinGW on a Windows PC without an MSYS environment. This is available at http://mingw-w64.org/doku.php/download, the package MingW-W64-builds, GCC version 7.2.0 and Mingw-w64 version 5.0.3.

GTEngine/GTEngine.{v12,v14,v15}.{vcxproj,vcxproj.filters}
GTMathematics.h
GTWindows.h
Include/Applications/MSW/GteMSWindowSystem.h
Include/Applications/MSW/DX11/GteWindowSystem.h
Include/Applications/MSW/WGL/GteWindowSystem.h
Source/Applications/GteEnvironment.cpp
Source/Applications/MSW/GteMSWindowSystem.cpp
Source/Applications/MSW/DX11/GteWindowSystem.cpp
Source/Applications/MSW/WGL/GteWindowSystem.cpp
Source/Applications/MSW/GteWICFileIO.cpp
Include/Graphics/GL4/GteOpenGL.h
Source/Graphics/GL4/GteOpenGL.cpp
Source/Graphics/GL4/WGL/GteWGLEngine.cpp
Source/Graphics/GL4/WGL/GteWGLExtensions.cpp

56 Updates to Version 3.11

February 19, 2018. This update is based solely on resolving warnings that occur with GCC 8.0.1 when compiling in the Fedora 28 (rawhide) environment.

Added -Wall to the make files that build the engine and samples.

GTEngine/makeengine.gte
GTEngine/Samples/makesample.gte

GCC 8.0.1 triggered a warning for Vector<N,Real>::Vector(std::initializer_lists<Real>), claiming potential undefined behavior. The warning is enabled using –Waggressive-loop-optimizations. The code is provably correct. The unit tests generate correct results no matter how many elements are in the initialier list. I believe the warning is incorrect, but decided to modify the code to use std::copy and std::fill instead.

Vector.h

February 17, 2018. Fixed a typecast from int to unsigned int in order that a comparison occurs between two unsigned integers.

DrawTarget.cpp

Commented out several lines of computation where the results are not used later in the algorithm. The lines were left intact for instructive purposes.

Camera.cpp

The name member initialization was moved to its proper location in the constructor initializer list to reflect its order of declaration in the header file.

Spatial.cpp

Replaced a memset call for an array of Vertex by a loop over the array, setting the individual members of Vertex to the zero 2-tuple. The locally defined struct Vertex is plain-old-data (POD), but GCC 8.0.1 complains that cannot be initialized via memset. The compiler warning is incorrect.

Font.cpp

The GL_INVALID_INDEX in glcorearb.h is declared as an unsigned int constant and the examples for GLSL reflection make comparisons between this constant and int members of an array. Typecast the members in the comparison to avoid a GCC 8.0.1 signed-unsigned comparison warning.

GL4Engine.cpp
GLSLReflection.cpp

Removed two unused variables and added a typecast to avoid a signed-unsigned comparison.

Applications/GLX/GteWindow.cpp

Initialized pmin and pmax to zero in DrawTriangulation for passing to ComputeExtremes in order to avoid a GCC 8.0.1 maybe-uninitialized warning when using the vectors later.

Samples/Geometrics/TriangulationCDTWindow.cpp

Reordered the initialization of the members of TriangulateEC::Vertex to the correct order as determined by the declarations in that nested class.

TriangulateEC.h

Removed computation of a direction vector because the result was not used in the function.

Samples/Graphics/CameraAndLightNodes/CameraAndLightNodesWindow.cpp

Reordered the initialization of the members in the constructor to the correct order as determined by the declarations in the class. Removed the local variable currDistance that existed for debugging but is not used in the code otherwise.

GenerateMeshUV.h

Replaced memset calls for std::vector members in order to zero the memory, avoiding a warning from GCC 8.0.1. The compilier complains that Vector<N,Real> is a nontrivial type, but in fact it is plain-old-data (POD). The compiler warning is incorrect.

BSplineCurve.h
BSplineSurface.h
BSplineVolume.h
NURBSCurve.h

Modified the code to use union of a float and unsigned int to avoid breaking the strict aliasing rules of C++.

Samples/Mathematics/RootFinding/RootFinding.cpp

Modified main to return a value dependent on iters to avoid GCC 8.0.1 warning about unused variable. The iters variable was included for debugging/inspection just to see how many iterations the eigensolver uses.

Samples/Mathematics/SymmetricEigensolver3x3/SymmetricEigensolver3x3.cpp

Reordered the initialization of the members in the constructor to the correct order as determined by the declarations in the class.

Samples/Physics/FreeFormDeformationWindow.cpp

Initialized key and dt to avoid a GCC 8.0.1 compiler warning about potential use of uninitialized variables.

NaturalSplineCurve.h

57 Updates to Version 3.10

February 8, 2018. Implemented the virtual function SetTitle of X-Windows and GLX.

Include/Applications/GLX/GteWindow.h
Source/Applications/GLX/GteWindow.cpp

The projects needed to include the template files for v15.

Tools/GenerateProject/GenerateProject.v12.vcxproj
Tools/GenerateProject/GenerateProject.v12.vcxproj.filters

February 6, 2018. The Picker class queried for the total number of primitives to examine and assumed that all primitives are to be searched; that is, the picker used GetNumPrimitives and the search loop started at 0. Instead, only the number of active number of primitives need to be searched and the first primitive is not necessarily at index 0. That is, the picker needed to use GetNumActivePrimitives and GetFirstPrimitive.

Picker.cpp

February 5, 2018. Added support for mouse wheel events.

Source/Applications/GLX/GteWindow.cpp

November 26, 2017. Fixed the spelling of the function name to access the denominator of the rational number. It is now correctly called GetDenominator.

BSRational.h

November 6, 2017. The convenience header file was missing include statements for GteFunctionsBSNumberh.h, GteFunctionsBSRational.h and GteFunctionsIEEEBinary16.h.

Mathematics.h

October 22, 2017. The functions GetTrailingBit for 64-bit inputs had a bug where an input of zero returned 32 when it should have been zero.

BitHacks.cpp

58 Updates to Version 3.9

September 16, 2017. Discontinued support for the Macintosh with OS X. Please read the installation and release notes for comments about this decision. The Xcode projects have been removed from the GTEngine distribution.

GTEngine.xcodeproj/project.pbxproj

MSVS 2013 does not support initializing static const members in the class declaration. Removed the static member, using instead its constant value in the small number of locations at which it occurred.

OBBTreeOfPoints.h

The OnCharPress override was calling the incorrect base-class function.

Samples/Graphics/CubeMaps/CubeMapsWindow.cpp

September 5, 2017. Broke the code during the upgrade of the Matrix class that adds support for initializer lists. Diagonal matrices that used to be constructed correctly now were initialized incorrectly. The ellipse-ellipse code had already been modified correctly.

IntrEllipsoid3Ellipsoide.h

Added an equality sign for consistency (older compilers might not support the code without it).

HLSLResource.cpp

September 1, 2017. Removed the include of <initializer_list> from the file because it is not needed by GVector.

GVector.h

Added the missing header file GteOBBTreeOfPoints.h to the projects.

GTEngine.{v12,v14,v15}.vcxproj

Added a filter subfolder named MSW to the Logger filter and moved the MSWindows-specific files to it.

GTEngine.{v12,v14,v15}.vcxproj.filters

August 10, 2017. The attempt to use an initializer list for Vector3 in the constructor initializer statement does not compile using Microsoft Visual Studio 2013. Moved the initialization to the constructor body.

ApprCylinder3.h

August 3, 2017. The eigenvectors returned by the non-iterative solver were supposed to form a right-handed set. A branch statement was causing the set to be left handed. Modified the code to return always a right-handed set. The ComputeEigenvector0 and ComputeEigenvector1 functions were passing the known eigenvalues by non-const reference. These are now passed by value. The Subtract function has a hard-coded float instead of Real.

SymmetricEigensolver3x3.h

July 26, 2017. Added code for computing a bounding-volume tree of oriented bounding boxes of a set of points in 3D. This is a port of Wild Magic 5 code but for points instead of triangles. The code for computing an OBB tree for triangles will be posted soon.

OBBTreeForPoints.h
GTMathematics.h

The MaxNeighbors template parameter is used only by the FindNeighbors member function. This parameter was removed from the class and the FindNeighbors function now is a template member function whose parameter is MaxNeighbors. Removed the helper class SortFunctor, using instead a simple lambda function in the place where the comparison is needed.

NearestNeighborQuery.h

July 25, 2017. Added const modifiers to the cone arguments.

IntrAlignedBox3Cone.h
IntrOrientedBox3Cone.h

July 6, 2017. The code was added to generate MSVS 2017 projects and solutions but the main function did not call the new code.

Tools/GenerateProject/GenerateProject.cpp

July 4, 2017. Removed the GTE_* bit-hack macros. Some of them produce incorrect results because of sign extension issues with integer types.

BitHacks.{h,cpp}
BSNumber.h
UIntegerALU32.h
UIntegerFP32.h
UIntegerAP32.cpp

July 2, 2017. The partitioning of the angle samples for multithreading was incorrect. Added two more constructors. One constructor allows you to choose the cylinder axis to be any of the eigenvectors of the covariance matrix. The other constructor allows you to choose any cylinder axis. Using these constructors, the least-squares fit uses the specified axis and minimizes the error over centers and radii.

ApprCylinder3.h

The vertex shaders compute z/w for perspective depth. This quantity needs to be linearly interpolated, not perspectively interpolated, so the parameter need to be modified with noperspective. The comments were improved in the pixel shaders. The application created depth textures with 24-bit depth and 8-bit stencil, but the OpenGL version was failing on a call to glGenBuffers when creating the staging buffer for the depth texture. The depth texture is now created using 32-bit depth and no stencil bits and the overlay pixel shaders were modified to handle the floating-point depth values.

Samples/Graphics/MultipleRenderTargets/MultipleRenderTargetsWindow.cpp
Samples/Graphics/MultipleRenderTargets/Shaders/MultipleRenderTargets.hlsl
Samples/Graphics/MultipleRenderTargets/Shaders/MultipleRenderTargetsVertex.glsl
Samples/Graphics/MultipleRenderTargets/Shaders/MultipleRenderTargetsPixel.glsl

June 29, 2017. Added the ability to specify whether the graphics engine uses a 24-bit depth and 8-bit stencil buffer or a 32-bit depth buffer. Previously, only the 24-8 format was supported. These changes are for GLX and for the applications that create non-graphics WGLEngine or GLXEngine objects.

GTEngine/Include/Graphics/GL4/GLX/GLXEngine.h
GTEngine/Source/Graphics/GL4/GLX/GLXEngine.cpp
GTEngine/Include/Applications/GLX/WindowSystem.h
GTEngine/Source/Applications/GLX/WindowSystem.cpp
Samples/Basics/AppendConsumeBuffers/AppendConsumeBuffers.cpp
Samples/Basics/IEEEFloatingPoint/IEEEFloatingPoint.cpp
Samples/Basics/ShaderReflection/ShaderReflection.cpp
Samples/Basics/DistanceSegments3/DistanceSegments3.cpp
Samples/Basics/PartialSums/PartialSums.cpp
Samples/Basics/RootFinding/RootFinding.cpp

Added the ability to reset a VertexFormat object to the state produced by the default constructor. This allows reusing the object within a scope.

VertexFormat.{h,cpp}

June 23, 2017. Fixed the comment in the shader about the depth equation. The code had said the equation is for OpenGL, but in fact the equation is for Direct3D.

Samples/Graphics/MultipleRenderTargets/Shaders/MultipleRenderTargets.hlsl

Added the ability to specify whether the graphics engine uses a 24-bit depth and 8-bit stencil buffer or a 32-bit depth buffer. Previously, only the 24-8 format was supported. The GL4DrawTarget::Enable call now needs to select whether the framebuffer attachment is depth-stencil or depth-only.

GTEngine/Include/Applications/WindowBase.h
GTEngine/Include/Applications/MSW/WGL/Window.h
GTEngine/Include/Graphics/DX11/DX11Engine.h
GTEngine/Include/Graphics/GL4/GL4Engine.h
GTEngine/Include/Graphics/GL4/WGL/WGLEngine.h
GTEngine/Source/Applications/WindowBase.cpp
GTEngine/Source/Applications/MSW/DX11/WindowSystem.cpp
GTEngine/Source/Applications/MSW/WGL/WindowSystem.cpp
GTEngine/Source/Graphics/DX11/DX11Engine.cpp
GTEngine/Source/Graphics/GL4/GL4Engine.cpp
GTEngine/Source/Graphics/GL4/WGL/WGLEngine.cpp
GTEngine/Source/Graphics/GL4/GL4DrawTarget.cpp

Added a new channel type, DF_UINT_24_8 to represent 24-bit depth and 8-bit stencil buffer values.

GTEngine/Include/Graphics/DataFormat.{h,cpp}
GTEngine/Source/Graphics/GL4/GL4Texture.cpp

59 Updates to Version 3.8

June 18, 2017. Removed the Arithmetic class that used the tag-dispatch pattern. The class forces applications to include the classes BSNumber, BSRational and IEEEBinary16 even though these classes are not used by the applications. Applications that have a need for functions using BSNumber, BSRational or IEEEBinary16 will include the new specific header files.

Arithmetic.h
GTMathematics.h
Functions.h
FunctionsBSNumber.h
FunctionsBSRational.h
FunctionsIEEEBinary16.h
GTEngine.v12.{vcxproj,vcxproj.filters}
GTEngine.v14.{vcxproj,vcxproj.filters}
GTEngine.v15.{vcxproj,vcxproj.filters}

Added explicit typecasts to avoid double-to-float conversion warnings on Ubuntu 16.04. The cmath file is exposed to the source file, so the compiler warnings are in error.

Samples/HelixTubeSurface/HelixTubeSurfaceWindow.cpp

Removed the ContourEdges graphics sample from source control. This does not yet have a GLSL implementation, so the makefile fails the build on a Linux box.

May 23, 2017. The segments are transformed to box coordinates and the intersection points (if any) are computed in box coordinates. They needed to be transformed back to the original coordinate space.

IntrSegment3AlignedBox3.h
IntrSegment3OrientedBox3.h

April 18, 2017. Replace the template typecasts in a couple of functions with the correct types.

Functions.h

April 17, 2017. Removed unnecessary include statements.

IntrEllipse2Ellipse2.h

60 Updates to Version 3.7

April 1, 2017. Added an Environment object to access the shader files so that the application can run from any folder. Added a console-window output when running FitCylinder in a Debug build using the mesh points; the program is very slow due to range and iterator checking for std::array and std::vector.

DistanceSegments3.cpp
MinimalCycleBasisWindow{h,cpp}
VideoStreamsWindow.cpp
FitCylinderWindow.cpp

March 19, 2017. Added projects and solution files for Microsoft Visual Studio 2017. Updated the GenerateProject tool to generate the skeleton files for MSVS 2017. Replaced the TargetPlatformVersion XML element with WindowsTargetPlatformVersion; the former prevented the correct retargeting of projects in a solution.

GenerateProject.v15.{vcxproj, vcxproj.filters}
ProjectTemplate.v15.{h,cpp}
GenerateProject.v14.{vcxproj, vcxproj.filters}
ProjectTemplate.v14.cpp

February 23, 2017. The arc-arc intersection code did not handle the case when the two arcs are the same. The code was fixed and the unit tests were updated to include a test for the equal-arcs case.

IntrArc2Arc2.h

61 Updates to Version 3.6

February 6, 2017. These files generated compiler errors with GCC-7.0.1 in Fedora Rawhide. They needed an include of <functional>.

ImageUtility{2,3}.h

Removed pragma to avoid warning with GCC-7.0.1 in Fedora Rawhide. Modified the code to eliminate the need to negate an unsigned integer (for a bit hack).

BitHacks.cpp

Removed the file because the implementation is not yet complete.

DualQuaternion.h

Removed the project because the port from Wild Magic 5 is not yet working.

WaterDropFormation.{v12,v14}.{sln,vcxproj,vcxproj.filters}
WaterDropFormation.{h,cpp}

62 Updates to Version 3.5

January 22, 2017. The conversion from 32-bit float to 16-bit float was incorrect because of an unnecessary shift of the trailing significand.

IEEEBinary16.cpp

January 2, 2017. The ETManifoldMesh::Insert function inserts the triangle into the triangle map mTMap early in the function, but if a nonmanifold condition is encountered, the function exits early and the (bad) triangle is in the map. Moved the map insertion to the end of the function. Also, the AssertOnNonmanifoldInsertion function now returns the previous value of the internal state mAssertOnNonmanifoldInsertion.

ETManifoldMesh.{h,cpp}

December 9, 2016. Added support for the topology types involving adjacent primitives: line-list-adjacent, line-strip-adjacent, triangle-list-adjacent, and triangle-strip-adjacent.

IndexFormat.h
IndexBuffer.{h,cpp}

November 28, 2016. Added another constructor to VertexBuffer to allow vertex-id for the vertex shaders but without a vertex buffer or structured buffer. The typical scenario is when the vertex shader itself uses the vertex-ids to generate the positions (a screen-space quad, for example).

Resource.cpp
VertexBuffer.{h,cpp}
DX11Engine.cpp
GL4Engine.cpp

63 Updates to Version 3.4

November 27, 2016. Added a new distance query for oriented boxes. Added a new sample application to test the query.

DistOrientedBox3OrientedBox3.h
Samples/Geometrics/DistanceOrientedBoxes/DistanceOrientedBoxes.{v12,v14}.{sln, vcxproj, vcxproj.filters}
Samples/Geometrics/DistanceOrientedBoxes/DistanceOrientedBoxes.{h,cpp}

November 26, 2016. Added a new distance query for aligned box and oriented boxes. Added a new sample application to test the query.

DistAlignedBox3OrientedBox3.h
Samples/Geometrics/DistanceAlignedBox3OrientedBox3/DistanceAlignedBox3OrientedBox3.{v12,v14}.{sln, vcxproj, vcxproj.filters}
Samples/Geometrics/DistanceAlignedBox3OrientedBox3/DistanceAlignedBox3OrientedBox3Window.{h,cpp}

Added a new distance query for aligned boxes. Added a new sample application to test the query.

DistAlignedBoxAlignedBox.h
Samples/Geometrics/DistanceAlignedBoxes/DistanceAlignedBoxes.{v12,v14}.{sln, vcxproj, vcx-proj.filters}
Samples/Geometrics/DistanceAlignedBoxes/DistanceAlignedBoxesWindow.{h,cpp}

November 25, 2016. Added a new distance query for point and convex polyhedron. Added a new sample application to test the query.

DistPoint3ConvexPolyhedron3.h
Samples/Geometrics/DistancePointConvexPolyhedron/DistancePointConvexPolyhedron.{v12,v14}.{sln, vcxproj, vcxproj.filters}
Samples/Geometrics/DistancePointConvexPolyhedron/DistancePointConvexPolyhedronWindow.{h,cpp}

Fixed a typographical error in the signature of a constructor.

ContPointInPolyhedron3.h

November 24, 2016. Modified the LCP solver to be templated and support dimensions known at compile time or at run time.

LCPSolver.h

November 23, 2016. Added new distance queries for rectangles and boxes. Added a new sample application to test the query.

DistRectangle3AlignedBox3.h
DistRectangle3OrientedBox3.h
Samples/Geometrics/DistanceRectangleBox/DistanceRectangleBox.{v12,v14}.{sln, vcxproj, vcx-proj.filters}
Samples/Geometrics/DistanceRectangleBox/DistanceRectangleBoxWindow.{h,cpp}

Modified the mesh loading code to eliminate redundancy in the data sets. Many of the meshes can be reduced to much smaller vertex and index buffers.

Samples/Graphics/Castle/CastleWindow.h
Samples/Graphics/Castle/LoadData.cpp

November 20, 2016. Added new distance queries for triangles and boxes. The LCP solver had a hard-coded maximum number of iterations: $n+1$ where n is the integer template parameter. This turns out to be insufficient; the maximum number depends on the query type and is effectively unknown. Added the ability to specify the maximum number of iterations; the default is currently n^2. Added a new sample application to test the query.

LCPSolver.h
DistTriangle3AlignedBox3.h
DistTriangle3OrientedBox3.h
Samples/Geometrics/DistanceTriangleBox/DistanceTriangleBox.{v12,v14}.{sln, vcxproj, vcxproj.filters}
Samples/Geometrics/DistanceTriangleBox/DistanceTriangleBoxWindow.{h,cpp}

Added a header dependency (when building without precompiled headers).

ApprCylinder3.h

November 17, 2016. The last occurrence of `result.segmentParameter` needed to be assigned `-segExtent`.

DistSegment3AlignedBox3.h

64 Updates to Version 3.3

November 13, 2016. Added a new implementation for the linear complementarity problem (LCP), which is a much simpler implementation than that of Wild Magic 5. It is templated and allows for exact rational arithmetic as well as floating-point types.

LCPSolver.h

Added new code for test-intersection query between boxes and cylinders. This uses the new implementation of the LCP solver. The test application verifies that the code works correctly.

IntrAlignedBox3Cylinder3.h
IntrOrientedBox3Cylinder3.h
Samples/Geometrics/IntersectBoxCylinder/IntersectBoxCylinder.{v12,v14}.{sln, vcxproj, vcxproj.filters}
Samples/Geometrics/IntersectBoxCylinder/IntersectBoxCylinderWindow.{h,cpp}

Refactored the box-sphere intersection testing to obtain separate implementations for axis-aligned bounding boxes and oriented bounding boxes. The test application verifies that the code works correctly.

IntrAlignedBox3Sphere3.h
IntrOrientedBox3Sphere3.h
Samples/Geometrics/IntersectBoxSphere/IntersectBoxSphere.{v12,v14}.{sln, vcxproj, vcxproj.filters}
Samples/Geometrics/IntersectBoxSphere/IntersectBoxSphereWindow.{h,cpp}

Added a bind call whose inputs are Direct3D 11 interfaces. This allows for binding DDS textures. The engine needs a DDS loaded, but for now you can use code that is available with the DirectX distribution.

DX11Engine.{h,cpp}
DX11Texture2.{h,cpp}

Modified the depth-texture code to correctly created resources and shader resource views that allow depth-textures to be used as inputs to shader programs.

TextureDS.{h,cpp}
DX11TextureDS.{h,cpp}

Modified the camera rig to process all active motions, which was the semantics of camera control in Wild Magic 5.

CameraRig.cpp

Added member accessors to set PVW matrices. This was needed in a Direct3D 12 sample where two sets of PVW matrices was needed per effect, one set for a draw call and the other set for an additional draw call. The matrix management was better handled by the application than internally in the effect classes.

ConstantColorEffect.{h,cpp}
LightingEffect.{h,cpp}
Texture2Effect.{h,cpp}
Texture3Effect.{h,cpp}
VertexColorEffect.{h,cpp}

Added a header file include statement to satisfy a dependency on maps.

MSWWindowSystem.h

Removed unused code.

Samples/Graphics/WireMesh/Shaders/WireMesh.hlsl

October 31, 2016. The Get template function that takes a handle as input needed to apply a static pointer cast to the returned value.

Shader.h

October 12, 2016. The OnClose callback was added to the window classes, but the call to the callback in the WNDPROC was missing.

MSWWindowSystem.cpp

Modified the device creation code when using the DX11Engine constructors that use null pointers for the adapters are called. The code now attempts to fall back to a version of Direct3D that the graphics card supports, and in the last try attempts to create a WARP (software) renderer. The default version strings of HLSLProgramFactory are also set accordingly, so the application writer does not need to hard-code attempts to select different versions of Direct3D until one is found that works.

DX11Engine.{h,cpp}

65 Updates to Version 3.2

September 24, 2016. Fixed some errors in the comments.

Resource.h

September 12, 2016. Starting support for Direct3D 12. The engine is not yet ready to support all the sample applications. It will be developed as time permits.

Files that have been modified to support DX12. Wrapped the code in GTE_USE_DX12 preprocessor blocks. Hiding the dependencies on DX11/DX12 by using macros.

Include/GTEngine.h
Include/GTWindows.h
Include/Graphics/DX11/GteHLSLBaseBuffer.h
Include/Graphics/DX11/GteHLSLByteAddressBuffer.h
Include/Graphics/DX11/GteHLSLConstantBuffer.h
Include/Graphics/DX11/GteHLSLParameter.h
Include/Graphics/DX11/GteHLSLResource.h
Include/Graphics/DX11/GteHLSLResourceBindInfo.h
Include/Graphics/DX11/GteHLSLSamplerState.h
Include/Graphics/DX11/GteHLSLShader.h
Include/Graphics/DX11/GteHLSLShaderFactory.h
Include/Graphics/DX11/GteHLSLShaderType.h

Include/Graphics/DX11/GteHLSLShaderVariable.h
Include/Graphics/DX11/GteHLSLStructuredBuffer.h
Include/Graphics/DX11/GteHLSLTexture.h
Include/Graphics/DX11/GteHLSLTextureArray.h
Include/Graphics/DX11/GteHLSLTextureBuffer.h
Source/Graphics/DX11/GteHLSLBaseBuffer.cpp
Source/Graphics/DX11/GteHLSLByteAddressBuffer.cpp
Source/Graphics/DX11/GteHLSLConstantBuffer.cpp
Source/Graphics/DX11/GteHLSLParameter.cpp
Source/Graphics/DX11/GteHLSLResource.cpp
Source/Graphics/DX11/GteHLSLResourceBindInfo.cpp
Source/Graphics/DX11/GteHLSLSamplerState.cpp
Source/Graphics/DX11/GteHLSLShader.cpp
Source/Graphics/DX11/GteHLSLShaderFactory.cpp
Source/Graphics/DX11/GteHLSLShaderType.cpp
Source/Graphics/DX11/GteHLSLShaderVariable.cpp
Source/Graphics/DX11/GteHLSLStructuredBuffer.cpp
Source/Graphics/DX11/GteHLSLTexture.cpp
Source/Graphics/DX11/GteHLSLTextureArray.cpp
Source/Graphics/DX11/GteHLSLTextureBuffer.cpp

New files in GeometricTools/GTEngine folder.

GTEngineDX12.v14.sln
GTEngineDX12.v14.vcxproj
GTEngineDX12.v14.vcxproj.filters
Include/GTGraphicsDX12.h
Include/Applications/MSW/DX12/GteWindow.h
Include/Applications/MSW/DX12/GteWindowSystem.h
Source/Applications/MSW/DX12/GteWindow.cpp
Source/Applications/MSW/DX12/GteWindowSystem.cpp
Include/Graphics/DX12/d3dx12.h
Include/Graphics/DX12/GteDX12Buffer.h
Include/Graphics/DX12/GteDX12CompileShader.h
Include/Graphics/DX12/GteDX12ConstantBuffer.h
Include/Graphics/DX12/GteDX12DescriptorHeap.h
Include/Graphics/DX12/GteDX12Engine.h
Include/Graphics/DX12/GteDX12Exception.h
Include/Graphics/DX12/GteDX12GraphicsObject.h
Include/Graphics/DX12/GteDX12Include.h
Include/Graphics/DX12/GteDX12IndexBuffer.h
Include/Graphics/DX12/GteDX12InputLayout.h
Include/Graphics/DX12/GteDX12InputLayoutManager.h
Include/Graphics/DX12/GteDX12PipelineState.h
Include/Graphics/DX12/GteDX12Resource.h
Include/Graphics/DX12/GteDX12SamplerState.h
Include/Graphics/DX12/GteDX12Texture2.h

Include/Graphics/DX12/GteDX12VertexBuffer.h
Source/Graphics/DX12/GteDX12CompileShader.cpp
Source/Graphics/DX12/GteDX12ConstantBuffer.cpp
Source/Graphics/DX12/GteDX12DescriptorHeap.cpp
Source/Graphics/DX12/GteDX12Engine.cpp
Source/Graphics/DX12/GteDX12Exception.cpp
Source/Graphics/DX12/GteDX12GraphicsObject.cpp
Source/Graphics/DX12/GteDX12IndexBuffer.cpp
Source/Graphics/DX12/GteDX12InputLayout.cpp
Source/Graphics/DX12/GteDX12InputLayoutManager.cpp
Source/Graphics/DX12/GteDX12PipelineState.cpp
Source/Graphics/DX12/GteDX12Resource.cpp
Source/Graphics/DX12/GteDX12SamplerState.cpp
Source/Graphics/DX12/GteDX12Texture2.cpp
Source/Graphics/DX12/GteDX12VertexBuffer.cpp
Samples/DX12/Texturing/Texturing.sln
Samples/DX12/Texturing/Texturing.vcxproj
Samples/DX12/Texturing/Texturing.vcxproj.filters
Samples/DX12/Texturing/Texturing.cpp
Samples/DX12/Texturing/Texturing.h
Samples/DX12/VertexColoring/VertexColoring.sln
Samples/DX12/VertexColoring/VertexColoring.vcxproj
Samples/DX12/VertexColoring/VertexColoring.vcxproj.filters
Samples/DX12/VertexColoring/VertexColoring.cpp
Samples/DX12/VertexColoring/VertexColoring.h

September 10, 2016. Ported the NURBS 2D curve sample from Wild Magic 5 to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Mathematics/NURBSCurveExample/NURBSCurveExample.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/NURBSCurveExample/NURBSCurveExample.{h,cpp}

Latest development on mesh refactoring. Added an input to Mesh that allows the derived classes to specify which topologies are valid in the MeshDescription constructor input. Added to ParametricCurve the ability to set a subinterval of t for the domain of the curve.

Mesh.h
ParametricCurve.h
RectangleMesh.h
RectanglePatchMesh.h
RevolutionMesh.h
TubeMesh.h

August 29, 2016. Added member function SetLayout. The Shader class is no longer a friend and can call the new member function directly. This also allows for development and testing of a Direct3D 12 engine. In

D3D12, the constant buffer must be a multiple of 256 bytes, so `ConstantBuffer` has conditional compilation to enforce this constraint.

ConstantBuffer.{h,cpp}
TextureBuffer.h
Shader.h

August 26, 2016. Added convenience constructor for uniform nonperiodic splines.

BasisFunction.h
Samples/Physics/Cloth/ClothWindow.cpp
Samples/Physics/FlowingSkirt/FlowingSkirtWindow.cpp
Samples/Physics/FreeFormDeformation/FreeFormDeformationWindow.cpp
Samples/Physics/GelatinCube/GelatinCubeWindow.cpp
Samples/Physics/MassPulleySpringSystem/MassPulleySpringSystemWindow.cpp
Samples/Physics/Rope/RopeWindow.cpp

More refactoring to provide meshing outside the graphics subsystem.

GTMathematics.h
RevolutionMesh.h
TubeMesh.h

August 20, 2016. Fixed some logic errors in the clamping of the line-cone intersection to the plane that truncates the cone. Modified the unit tests for full code coverage and validation of results.

IntrLine3Cone3.h

August 17, 2016. Started the refactoring of code to build meshes from curves and surfaces. The idea is to have code independent of the graphics subsystem but that can be used by graphics applications.

Mesh.h
RectanglePatchMesh.h
VertexAttribute.h
TubeMesh.h
RectangleMesh.h
MeshFactory.{h,cpp}
Samples/Physics/Cloth/ClothWindow.{h,cpp}
Samples/Physics/GelatinCube/GelatinCubeWindow.{h,cpp}
Samples/Physics/HelixTubeSurface/HelixTubeSurfaceWindow.cpp
Samples/Physics/MassPulleySpringSystem/MassPulleySpringSystemWindow.cpp
Samples/Physics/Rope/RopeWindow.cpp

Added a least-squares estimator for fitting a cylinder to points.

GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GteMathematics.h
ApprCylinder3.h
Samples/Mathematics/FitCylinder/FitCylinder.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/FitCylinder/FitCylinder.{h,cpp}
Samples/Mathematics/FitCylinder/mesh.txt

August 7, 2016. The Matrix and GMatrix constructors now guarantee that the matrices are initialized to zero in the cases where previously the matrices were uninitialized by the constructor.

Matrix.h
GMatrix.h

Added a Boolean input named robust that defaults to false to any functions involving Length or Normalize. The default uses the standard algorithm for normalizing a vector. The robust algorithms avoid floating-point overflow in the computation of length.

Vector.h
Vector{2,3,4}.h

July 28, 2016. The last Hyperplane constructor listed in the header file had an off-by-one indexing error that led to incorrect construction of the plane.

Hyperplane.h

July 25, 2016. The UpdateSupport function needed to wrap the j0 index from -1 to numVertices-1, but the test for negativity was incorrectly applied to j1.

MininumAreaBox2.h

July 23, 2016. Added a new document to describe robust computation of the eigenvalues and eigenvectors of a 2×2 symmetric matrix.

RobustEigenSymmetric2x2.pdf (*A Robust Eigensolver for* 2×2 *Symmetric Matrices*)

July 16, 2016. Ported the simple pendulum with friction sample application from Wild Magic 5 to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Physics/SimplePendulumFriction/SimplePendulumFriction.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/SimplePendulumFriction/SimplePendulumFriction.{h,cpp}
Samples/Physics/SimplePendulumFriction/PhysicsModule.{h,cpp}

Added ExitFullscreen member function that does not require a DXGIOutput class object. If a user decides to go fullscreen in a GTEngine application by pressing ALT+ENTER but then terminates the application before returning to windowed mode, the application triggers an assertion about not all DX11 resources being freed. Adding a call to ExitFullscreen in the Window destructor fixes the problem.

DX11Engine.{h,cpp}
Source/Applications/MSW/DX11/Window.cpp

Removed the copyright notices from the automatically generated h and cpp files.

Tools/GenerateProject/ProjectTemplate.{v12,v14}.cpp

July 14, 2016. Ported the simple pendulum sample application from Wild Magic 5 to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Physics/SimplePendulum/SimplePendulum.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/SimplePendulum/SimplePendulum.cpp

July 13, 2016. Ported the mass-pulley-spring sample application from Wild Magic 5 to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Physics/MassPulleySpringSystem/MassPulleySpringSystem.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/MassPulleySpringSystem/MassPulleySpringSystemWindow.{h,cpp}
Samples/Physics/MassPulleySpringSystem/PhysicsModule.{h,cpp}
Metal.png

July 12, 2016. Factored out common code in RectangleSurface, BoxSurface, and TubeSurface and moved it to base class MeshSurface. Ported the helix tube surface sample application from Wild Magic 5 to GTEngine.

GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GTMathematics.h
MeshSurface.h
RectangleSurface.h
BoxSurface.h
TubeSurface.h

Samples/Physics/Cloth/ClothWindow.{h,cpp}
Samples/Physics/Rope/RopeWindow.{h,cpp}
Samples/Physics/HelixTubeSurface/HelixTubeSurface.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/HelixTubeSurface/HelixTubeSurfaceWindow.{h,cpp}

July 10, 2016. Ported the gelatin cube sample application from Wild Magic 5 to GTEngine. This includes porting the BoxSurface class, which has been redesigned similar to RectangleSurface which is independent of the graphics system. Renamed the Tessellate member to GetVertices, because the triangulation is fixed once parameter domain is specified (the user cannot retessellate without creating a new object). Updated the gelatin blob sample to use depth-stencil and culling state that the original sample used.

GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GTMathematics.h
RectangleSurface.h
BoxSurface.h
Samples/Physics/GelatinCube/GelatinCube.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/GelatinCube/GelatinCubeWindow.{h,cpp}
Samples/Physics/GelatinCube/PhysicsModule.{h,cpp}
Samples/Physics/GelatinBlob/GelatinBlobWindow.{h,cpp}

66 Updates to Version 3.1

July 6, 2016. Added Linux-specific includes to be consistent with what is done on the Microsoft Windows platform.

GTEngine.h
GTGraphicsGL4.h

Added the extension headers copied from the OpenGL Registry into our local directory structure and modified the includes of the extension headers to point to these.

Graphics/GL4/GL/glext.h
Graphics/GL4/GL/glxext.h
Graphics/GL4/GL/wglext.h
Graphics/GL4/GLX/GteGLXExtensions.cpp
Graphics/GL4/WGL/GteWGLExtensions.cpp

Fixed bugs introduced with the previous modifications.

NURBSSurface.h
NURBSVolume.h

July 3, 2016. Removed the properties from the reflection queries that are from OpenGL 4.4 or 4.5. GTEngine is designed to run with minimum version 4.3. Added several missing ENUM members from the map; these are associated with cube map arrays.

GLSLReflection.cpp

July 2, 2016. Replaced the call to the OpenGL 4.1 function glProgramUniform1i by a call to the OpenGL 2.0 function glUniform1i. We already have an active program, so the correct location's value is set. Cleaned up the Enable and Disable calls for textures and texture arrays.

GL4Engine.cpp

Removed unnecessary include. Commented out function that is not used.

GLXExtensions.cpp

July 1, 2016. Removed LogError messages that duplicate ones reported in child function calls.

MSW/MSWWindowSystem.h
GLX/WindowSystem.h

June 30, 2016. Modified the OpenGL subsystem to test for a required minimum version and to gracefully exit when that minimum is not met. Currently, the minimum version is OpenGL 4.3 to support compute shaders. The subsystem needs redesigning (and shaders rewritten) before we can allow the applications to specify the required minimum version. Also, Linux distributions might not ship the file glcorearb.h. We have provided this in a subfolder within the GTEngine distribution to guarantee that the code will compile. If your Linux distribution has glcorearb.h, read the release notes about how you can regenerate GteOpenGL.cpp for this file.

Graphics/GL4/GL/glcorearb.h
GL4Engine.{h,cpp}
GLX/GLXEngine.{h,cpp}
GLX/WindowSystem.cpp
WGL/WGLEngine.{h,cpp}
WGL/WindowSystem.cpp
Tools/GenerateOpenGLWrapper/GenerateOpenGLWrapper.cpp
Tools/GenerateOpenGLWrapper/GteOpenGL.{h,cpp}
Tools/GenerateOpenGLWrapper/Initialize.txt
Tools/GenerateOpenGLWrapper/Version.txt

Modified the make file for a sample to work around an apparent bug in Ubuntu 14.04 regarding the pthread library. Removed the link of libraries GLU because GTEngine does not use this.

Samples/makesample.gte

Changed the DarbouxFrame and FrenetFrame constructors to accept shared pointers rather than raw pointers to be consistent with the GTEngine design. Redesigned RectangleSurface to use an interface that will be ported to other surface-type objects that will be ported from Wild Magic 5 to GTEngine.

DarbouxFrame.h
FrenetFrame.h
RectangleSurface.h
TubeSurface.h
Samples/Physics/Cloth/ClothWindow.cpp

67 Updates to Version 3.0

June 28, 2016. Ported the free-top-fixed-tip sample application from Wild Magic 5 to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Physics/FreeTopFixedTip/FreeTopFixedTip.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/FreeTopFixedTip/FreeTopFixedTipWindow.{h,cpp}
Samples/Physics/FreeTopFixedTip/PhysicsModule.{h,cpp}
Samples/Data/TopTexture.png

Ported the gelatin blob sample application from Wild Magic 5 to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Physics/GelatinBlob/GelatinBlob.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/GelatinBlob/GelatinBlobWindow.{h,cpp}
Samples/Physics/GelatinBlob/PhysicsModule.{h,cpp}
Samples/Data/Water.png

These files were tested on a Linux machine, but some changes to them were not carried over to the SVN repository on a PC. Fixed the out-of-sync code.

Source/Applications/GLX/GteWICFileIO.cpp
Source/Applications/GLX/GteWindow.cpp

June 26, 2016. Fixed compiler errors on Fedora 24 with gcc-6.1.1.

Font.cpp
MeshFactory.cpp

Converted the raw pointer member of PickRecord to a shared pointer to be consistent with other subsystems in GTEngine. The Picker internal member functions now pass shared pointers.

Picker.{h,cpp}
PickRecord.{h,cpp}

The constructors required a nonnull pointer to data that is then copied into internal class members. Invariably applications have to allocate arrays, fill them, and then pass them to the constructors to be copied. The design is modified to avoid having to pass in data, but the internal members are still allocated. The application can then set the data after construction, thereby avoiding the double allocation.

BSplineCurve.h
BSplineSurface.h
BSplineVolume.h
NURBSCurve.h
NURBSSurface.h
NURBSVolume.h

Ported the free-form deformation sample application from Wild Magic 5 to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Physics/FreeFormDeformation/FreeFormDeformation.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/FreeFormDeformation/FreeFormDeformationWindow.{h,cpp}

June 25, 2016. The DX11-specific samples were moved to a new folder Samples/DX11. Modified the subfolder, project, solution, and file names to eliminate the redundant D3D11 in those names.

Samples/DX11/LowLevel/*
Samples/DX11/LowLevelStream/*
Samples/DX11/RawBuffers/*
Samples/DX11/SharedTextures/*
Samples/DX11/RawBuffers/RawBuffers.cpp

Modified the file preamble in the templates for generating the default application source files.

Tools/GenerateProjects/ProjectTemplate.{v12,v14}.cpp

The ODE function was using mState rather than input. The former quantity is updated in the ODE solver itself.

Samples/Physics/BallHill/PhysicsModule.cpp

Ported the Foucault pendulum sample application from Wild Magic 5 to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Physics/FoucaultPendulum/FoucaultPendulum.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/FoucaultPendulum/FoucaultPendulumWindow.{h,cpp}
Samples/Physics/FoucaultPendulum/PhysicsModule.{h,cpp}
Samples/Data/Wood.png

June 20, 2016. The query needed to use `line.direction` in the dot product.

DistLine3OrientedBox3.h

68 Updates to Version 2.5

June 19, 2016. The port of Wild Magic 5 OpenGL graphics for Linux and GLX to GTEngine is complete.

June 15, 2016. The toggle between windowed and fullscreen mode used the `IDXGIOutput` value queried from the `IDXGIAdapter` objects. If an output goes fullscreen, an entry was added to a `std::map` with the output address as the key. However, these addresses change from query to query, so the toggle did not work. The code was modified to use the output display name as the key, which does uniquely identify which window to toggle between windowed and fullscreen mode.

DX11Engine.{h,cpp}

June 13, 2016. Modifications due to Cygwin g++ compiling inline functions whether or not they are used, as compared to Microsoft Visual Studio where they do not. The Cygwin environment found several problems with no-PCH. MSVS also appears to allow several standard C library functions to be used without having to include their header files.

GTEngine.h
GenerateMeshUV.h
ConstantBuffer.{h,cpp}
TextureBuffer.{h,cpp}
Font.cpp
Visual.cpp
MeshFactory.cpp
GL4InputLayout.cpp
ImageUtility2.cpp
Wrapper.cpp
TetrahedronKey.cpp

Added a `std::flush` so that messages are printed before a segmentation fault and core dump when using Cygwin.

LogToStdout.cpp

June 12, 2016. The explicit specializations of Set were declared inside the class scope but C++ requires them to occur outside the class scope.

Shader.h

The source file accesses members of the StructuredBuffer, so the header file for that class must be included.

VertexBuffer.cpp

Replaced UINT by GLuint.

GL4Buffer.cpp

As of 12 June 2016, Cygwin's glcorearb.h is revision

```
** Khronos $Revision : 27684 $ on $Date : 2014 - 08 - 11 01:21 : 35
   - 0700 (Mon, 11 Aug 2014) $
```

but the glcorearb.h we use on Microsoft Windows is revision

```
** Khronos $Revision : 28299 $ on $Date : 2014 - 09 - 25 04:11 : 58
   - 0700 (Thu, 25 Sep 2014) $
```

The Cygwin file does not have declarations for the four functions

```
static PFNGLGETQUERYBUFFEROBJECTI64VPROC sglGetQueryBufferObjecti64v = nullptr;
static PFNGLGETQUERYBUFFEROBJECTIVPROC sglGetQueryBufferObjectiv = nullptr;
static PFNGLGETQUERYBUFFEROBJECTUI64VPROC sglGetQueryBufferObjectui64v = nullptr;
static PFNGLGETQUERYBUFFEROBJECTUIVPROC sglGetQueryBufferObjectuiv = nullptr;
```

There are some #define variations also, but these do not affect compilation of GTEngine. Added conditional compilation to avoid the compiler errors on Cygwin. Also, the GL extension string parsing for versions of OpenGL prior to 3.0 was incorrect and used string functions specific to Microsoft Windows. Replaced the parsing with std::string functions.

OpenGL.cpp
Tools/GenerateOpenGLWrapper/Initialize.txt

Moved WICFileIO files to the MSW subfolder, because they are dependent on Microsoft Windows.

GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
WICFileIO.{h,cpp}
Samples/Graphics/BlendedAnimations/BipedManager.cpp
Samples/Graphics/BlendedTerrain/BlendedTerrainEffect.cpp
Samples/Graphics/BumpMaps/SimpleBumpMapEffect.cpp

The class header was missing an include of cstdarg, which Microsoft Visual Studio appears not to complain about (but Cygwin g++ did). The SplitPath and FullPath functions are not used in the engine and are specific to Microsoft Windows, so they were removed. The CreateString* functions were used only in two sample applications, but they can be replaced by insteading using std::to_string for creating strings with unformatted data or std::stringstream for formatted data.

Environment.{h,cpp}
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp
Samples/Imagics/VideoStreams/VideoStreamsWindow.cpp

Added static constants for key identifiers, key modifiers, mouse buttons, mouse state, and mouse modifiers. Removed the enumerations that were previously used for the mouse. This mimics what Wild Magic 5 does and hides platform-specific values (each OS platform defines the static members with the appropriate values).

WindowBase.{h,cpp}
Window3.cpp
MSWWindow.cpp
MSWWindowSystem.cpp
GLX/Window.cpp

A modification was made in DX11TextureCube that indicated a cube map cannot be a dynamic-update resource. The same modification needed to be made in DX11TextureCubeArray.

DX11TextureCubeArray.cpp

June 11, 2016. Refactored the Window and WindowSystem classes into WindowBase, which is independent of operating system; MSWindow and MSWindowSystem, which are for dependent on the Microsoft Windows platform but are independent of graphics API, and platform-specific versions of Window and WindowSystem. The Application folder now has subfolders that partition the files based on operating system and graphics API. This change is part of the incorporation of GLX support for Linux into GTEngine.

GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
Include/GTApplications.h
Include/GTWindows.h

Include/Applications/WindowBase.h
Include/Applications/Window{2,3}.h
Include/Applications/MSW/MSWWindows.h
Include/Applications/MSW/MSWWindowSystem.h
Include/Applications/MSW/DX11/Window.h
Include/Applications/MSW/DX11/WindowSystem.h
Include/Applications/MSW/WGL/Window.h
Include/Applications/MSW/WGL/WindowSystem.h
Source/Applications/WindowBase.cpp
Source/Applications/MSW/MSWWindows.cpp
Source/Applications/MSW/MSWWindowSystem.cpp
Source/Applications/MSW/DX11/Window.cpp
Source/Applications/MSW/DX11/WindowSystem.cpp
Source/Applications/MSW/WGL/Window.cpp
Source/Applications/MSW/WGL/WindowSystem.cpp

Moved the Intel SSE code to a new subfolder because it is specific to Microsoft Windows.

GTMathematics.h
Include/Mathematics/MSW/IntelSSE.h
Source/Mathematics/MSW/IntelSSE.cpp

Cygwin's Linux shell defines WIN32, which exposes blocks of GTEngine code that are not intended for Linux. Revised the logic of testing the macros for underlying operating system. Also, Cygwin's include of glcorearb.h occurs without turning off the annoying min and max macros of Windows.h. Turned them off inside OpenGL.h before the include of glcorearb.h.

GTEngineDEF.h
OpenGL.h

An HLSL-dependent header file was included without conditional compilation for DX11 versus GL4. Moved it into a conditional compilation block, which led compiler complaints when GTE_DEV_OPENGL is defined and when the Cygwin min/max problem was not yet fixed. The Cygwin fixes eliminate the compiler complaints and the HLSL file is now protected. Also then had to include GteLogger.h that is used by GteShader.h. And finally, this broke a sample application that used both HLSL and GLSL related blocks of code, which is now fixed by conditional compilation.

Shader.h
BlendedTerrainEffect.cpp

June 5, 2016. Created new folders GLX and WGL to store OpenGL code specific to platforms. Moved the WGL-specific files to the WGL folder.

GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GTGraphicsGL4.h
WGLEngine.{h,cpp}
WGLExtensions.cpp
WindowSystem.cpp
Samples/Basics/ShaderReflection/ShaderReflection.cpp

Factored out Microsoft-Windows-specific files and code to new folders. Added new macro __MSWINDOWS__ that is enabled when WIN32 or _WIN64 is defined. This makes it clearer in the code when conditional compilation enables or disables OS-specific blocks of code.

GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GTPhysics.h
OpenGL.{h,cpp}
LogReporter.{h,cpp}
Timer.h
Wrapper.cpp
LogToMessageBox.{h,cpp} (MSWindows-specific)
LogToOutputWindow.{h,cpp} (MSWindows-specific)
TetrahedronKey.h
Samples/Basics/AppendConsumeBuffer/AppendConsumeBuffers.cpp
Samples/Basics/IEEEFloatingPoint/IEEEFloatingPoint.cpp
Samples/Basics/ShaderReflection/ShaderReflection.cpp
Samples/Geometrics/DistanceSegments3/DistanceSegments3.cpp
Samples/Graphics/Castle/CastleWindow.cpp
Samples/Mathematics/PartialSums/PartialSums.cpp
Samples/Mathematics/RootFinding/RootFinding.cpp
Tools/GenerateOpenGLWrapper/GenerateOpenGLWrapper.cpp
Tools/GenerateOpenGLWrapper/GteOpenGL.h
Tools/GenerateProject/ProjectTemplate.{v12,v14}.cpp

Cygwin complained about the -fPIC option specified as a CFLAGS option. Removed it; Wild Magic 5 does not have it in CFLAGS either.

GTEngine/makefile.gte

June 2, 2016. Added a custom visualizer file for the Vector and Matrix classes. We will add more visualizers as needed.

gte.natvis

June 1, 2016. Revised the VertexBuffer interface to require the structured buffer to be passed to the constructor when rendering by vertex-id is desired. This is a convenient encapsulation that allows you to access the vertex data through the vertex buffer.

VertexBuffer.{h,cpp}
DX11Engine.cpp
GL4Engine.cpp
Samples/Geometrics/AllPairsTriangles/AllPairsTrianglesWindow.cpp
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp
Samples/Physics/MassSprings3D/MassSprings3DWindow.cpp

May 31, 2016. A major revision of the document. The new version shows how to estimate vertex tangents and vertex normals for a parameterized mesh (or mesh with texture coordinates) at each vertex using the entire set of triangles that share the vertex. The algorithm is useful for constructing the vertex normals and tangents whose interpolated values are used in a pixel shader for tangent-space normal mapping.

MeshDifferentialGeometry.pdf (*Mesh Differential Geometry*)

The nonuniform Akima interpolation code needed to be ported along with the uniform interpolation code. The associated PDF had pseudocode that referenced Wild Magic 5 code, so it was modified for GTEngine (and tested with real code).

IntpAkimaNonuniform1.h
AkimaInterpolation.pdf (*Akima Interpolation for Nonuniform 1D Data*)

May 29, 2016. The Result structure now returns parameters for the t-interval $[0, 1]$ (the endpoint form of the segment) and for the s-interval $[-e, e]$ (for the center-direction-extent form of the segment).

IntrSegment2AlignedBox2.h
IntrSegment2OrientedBox2.h

69 Updates to Version 2.4

May 25, 2016. The find-intersection queries needed to translate its points of intersection back to the original coordinate system by adding the box center. The unit tests had only boxes with centers at the origin (updated these tests). Modified the result.parameter values to be t-values for the segment parameterization $(1-t)P_0 + tP_1$ for $t \in [0, 1]$. They had been relative to the center-direction-extent form which is computed internally in the queries, $C + sD$, where D is unit length and $s \in [-\varepsilon, \varepsilon]$ with ε the extent.

IntrSegment2AlignedBox2.h
IntrSegment2OrientedBox2.h

May 24, 2016. Removed the LogWarning from Shader::Get that returns a handle. It is reasonable for an application to use Shader::Get to determine whether a Shader::Set call will be successful for the specified resource name.

Shader.cpp

The test-intersection query for Ray2 and AlignedBox2 had an incorrect ordering of parameters in the DoQuery call.

IntrRay2AlignedBox2.h

May 22, 2016. Ported WM5 code for partitioning a convex polygon by a plane in 3D and made the code robust when the polygon is nearly parallel to the plane. Also ported test-intersection and find-intersection queries for triangles and oriented boxes in 3D. Added a sample application that serves as the test code.

GTMathematics.h
GteIntrConvexPolygonPlane.h
GteIntrTriangle3OrientedBox3.h
Samples/Geometrics/IntersectTriangleBox/IntersectTriangleBox.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Geometrics/IntersectTriangleBox/IntersectTriangleBoxWindow.{h,cpp}

May 18, 2016. The Window::SetTitle declaration had the inline modifier, but the body was not inline, which hides the actual definition. Removed the modifier.

Window.h

May 15, 2016. Added an Inverse member function that returns the inverse of a Transform as another Transform object.

Transform.{h,cpp}

Modified the generated code to include the next version number.

Tools/GenerateProject/ProjectTemplate.{v12,v14}.cpp

Added function HLift to embed a 3×3 matrix as the upper block of a 4×4 matrix. Added HProject to extract a 3×3 matrix from the upper block of a 4×4 matrix.

Matrix.h

Removed redundant and unused code.

Samples/Graphics/BumpMaps/BumpMapsWindow.cpp

May 13, 2016. When a D3D11 texture is created that shares another texture, if the original texture has a staging texture associated with it, one must also be created for the newly sharing texture.

DX11Texture2.cpp

May 11, 2016. Changed a LogError to LogWarning regarding an invalid object type encountered during picking. An application might very well have such a class that should not participate in picking, so you can tailor the logging system not to launch a dialog box when these objects are encountered during picking.

Picker.cpp

Fixed some comments (removed reference to obsolete class member functions).

CameraRig.h

May 5, 2016. The WICFileIO::Load function had two places where it was returning false instead of nullptr.

WICFileIO.cpp

May 1, 2016. New files that implement the QR algorithm for computing eigenvalues of unsymmetric matrices and for using the same algorithm for computing the roots of cubic and quartic polynomials.

UnsymmetricEigenvalues.h
CubicRootsQR.h
QuarticRootsQR.h

Fixed a compiler warning that now occurs with MSVS 2015 Update 2.

Integration.h

April 29, 2016. Reverted the changes to the buffer updating that was posted on April 1, 2016 for Version 2.3. The text drawing system of GTEngine appears not to work properly when the no-overwrite mode is used on different vendor graphics cards. If you need to update subbuffers, set up the buffer resource so that it is enabled for copies from CPU to GPU and use CopyCpuToGpu instead.

DX11ConstantBuffer.cpp
DX11IndexBuffer.cpp
DX11IndirectArgumentsBuffer.cpp
DX11RawBuffer.cpp

DX11StructuredBuffer.cpp
DX11TextureBuffer.cpp
DX11VertexBuffer.cpp
DX11Buffer.cpp

Instead of creating and destroying a query object on each call to WaitForFinish, the query is now created in the first call and saved as a class member. That object is released on destruction.

DX11Engine.{h,cpp}

April 27, 2016. Added functions to copy GPU to GPU directly. Only the DX11Engine versions have been implemented (we need them at the moment). GL4Engine versions will be implemented later.

GraphicsEngine.h
DX11Engine.{h,cpp}
DX11Resource.h
DX11Buffer.{h,cpp}
DX11Texture.{h,cpp}
GL4Engine.{h,cpp}
GL4Resource.h
GL4TextureSingle.h
GL4TextureArray.h

Exposed the ability to flush the GPU command buffer in order for shared textures to be updated on one device when another device modifies the data. The new base class function is called Flush().

GraphicsEngine.h
DX11Engine.{h,cpp}
GL4Engine.{h,cpp}
Samples/Graphics/SharedTexturesD3D11/SharedTexturesWindow.cpp

April 26, 2016. Added BindProgram to allow explicit binding before the first call to a compute shader.

GraphicsEngine.h
DX11Engine.{h,cpp}
GL4Engine.{h,cpp}

April 17, 2016. Replace two occurrences of hard-coded float with template parameter Real.

ApprEllipseByArcs.h

Added new 2D intersection queries.

GTMathematics.h
Sector2.h
IntrDisk2Sector2.h
IntrHalfspace2Polygon2.h
IntrOrientedBox2Sector2.h

April 16, 2016. The DestroyStorage function is intended to free up the CPU memory for the resource, but clear() does not do this; the capacity remains the same. Added an additional call to shrink_to_fit to free up the memory. Replaced &mStorage[0] by mStorage.data().

Resource.{h,cpp}

April 13, 2016. Ported the Wild Magic 5 code for approximating an axis-aligned ellipse by circular arcs. Ported and upgraded the sample application (that was only in Wild Magic 4) to illustrate the approximation. The PDF describing the algorithm had pseudocode and references to the Wild Magic 4 sample but did not have pseudocode for the implementation of the algorithm. The PDF has been updated to eliminate the WM4 reference and to include an implementation. Added new functions to ImageUtility2 and Window2 to draw axis-aligned ellipses using a Bresenham-style algorithm.

GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
ImageUtility2.{h,cpp}
Windows.{h,cpp}
ApprEllipseByArcs.h
Samples/Mathematics/ApproximateEllipseByArcs/ApproximateEllipseByArcs.{v12,v14}.{vcxproj,vcxproj.filters,sln}
Samples/Mathematics/ApproximateEllipseByArcs/ApproximateEllipseByArcsWindow.{h,cpp}
ApproximateEllipse.pdf (*Approximating an Ellipse by Circular Arcs*)

April 12, 2016. Removed the LogError code when Shader::Get cannot find a resource by name. It is reasonable that an application might have a generic shader system for which a resource might be attached to some shaders but not to others, and Shader::Get is a way to determine this.

Shader.h

April 10, 2016. The wrong flag was tested in WindowSystem::MessagePump when the user requests no idle loop.

WindowSystem.h

70 Updates to Version 2.3

March 12, 2016. Ported the Wild Magic 5 sample application MorphFaces.

GTBuildAll.{v12,v14}.sln
Samples/Graphics/MorphFaces/MorphFaces.{v12,v14}.{sln, vcxproj, vcxproj.filters}
Samples/Graphics/MorphFaces/MorphFaces.cpp
Samples/Graphics/MorphFaces/MorphFacesWindow.{h,cpp}
Samples/Graphics/MorphFaces/CubicInterpolator.h
Samples/Graphics/MorphFaces/Data/Eye.png
Samples/Graphics/MorphFaces/Data/LightColorSampler.txt
Samples/Graphics/MorphFaces/Data/SharedTexTri.txt
Samples/Graphics/MorphFaces/Data/M*.txt (25 morph targets)

March 13, 2016. Factored the WGL code out of GL4Engine into a new derived class WGLEngine. This makes GL4Engine independent of the windowing system in preparation for adding GLX and Linux support.

WGLEngine.{h,cpp}
GL4Engine.{h,cpp}
WindowSystem.cpp
Samples/Basics/AppendConsumeBuffers/AppendConsumeBuffers.cpp
Samples/Basics/IEEEFloatingPoint/IEEEFloatingPoint.cpp
Samples/Basics/ShaderReflection/ShaderReflection.cpp
Samples/Geometrics/DistanceSegments3/DistanceSegments3.cpp
Samples/Mathematics/PartialSums/PartialSums.cpp
Samples/Mathematics/RootFinding/RootFinding.cpp

Removed unused functions from the public interface. Moved the message pump code from the main function in the applications into the WindowSystem class. This leads to an interface that is independent of operating system and windowing system in preparation for adding GLX and Linux support. Moved the main functions into the corresponding application window classes to reduce the number of files to maintain.

WindowSystem.h
Samples/Geometrics/AllPairsTriangles/AllPairsTrianglesWindow.vcxproj.filters
Samples/Geometrics/AllPairsTriangles/AllPairsTrianglesWindow.cpp
Samples/Geometrics/AllPairsTriangles/AllPairsTriangles.cpp
Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2DWindow.vcxproj.filters
Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2DWindow.cpp
Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2D.cpp
Samples/Geometrics/ConvexHull2D/ConvexHull2DWindow.vcxproj.filters
Samples/Geometrics/ConvexHull2D/ConvexHull2DWindow.cpp
Samples/Geometrics/ConvexHull2D/ConvexHull2D.cpp
Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow.vcxproj.filters
Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow.cpp
Samples/Geometrics/ConvexHull3D/ConvexHull3D.cpp

Samples/Geometrics/Delaunay2D/Delaunay2DWindow.vcxproj.filters
Samples/Geometrics/Delaunay2D/Delaunay2DWindow.cpp
Samples/Geometrics/Delaunay2D/Delaunay2D.cpp
Samples/Geometrics/Delaunay3D/Delaunay3DWindow.vcxproj.filters
Samples/Geometrics/Delaunay3D/Delaunay3DWindow.cpp
Samples/Geometrics/Delaunay3D/Delaunay3D.cpp
Samples/Geometrics/IntersectBoxCone/IntersectBoxConeWindow.vcxproj.filters
Samples/Geometrics/IntersectBoxCone/IntersectBoxConeWindow.cpp
Samples/Geometrics/IntersectBoxCone/IntersectBoxCone.cpp
Samples/Geometrics/MinimalCycleBasis/MinimalCycleBasisWindow.vcxproj.filters
Samples/Geometrics/MinimalCycleBasis/MinimalCycleBasisWindow.cpp
Samples/Geometrics/MinimalCycleBasis/MinimalCycleBasis.cpp
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.vcxproj.filters
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.cpp
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2D.cpp
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2DWindow.vcxproj.filters
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2DWindow.cpp
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2D.cpp
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.vcxproj.filters
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.cpp
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3D.cpp
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3DWindow.vcxproj.filters
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3DWindow.cpp
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3D.cpp
Samples/Geometrics/ShortestPath/ShortestPathWindow.vcxproj.filters
Samples/Geometrics/ShortestPath/ShortestPathWindow.cpp
Samples/Geometrics/ShortestPath/ShortestPath.cpp
Samples/Geometrics/TriangulationCDT/TriangulationCDTWindow.vcxproj.filters
Samples/Geometrics/TriangulationCDT/TriangulationCDTWindow.cpp
Samples/Geometrics/TriangulationCDT/TriangulationCDT.cpp
Samples/Geometrics/TriangulationEC/TriangulationECWindow.vcxproj.filters
Samples/Geometrics/TriangulationEC/TriangulationECWindow.cpp
Samples/Geometrics/TriangulationEC/TriangulationEC.cpp
Samples/Geometrics/VertexCollapseMesh/VertexCollapseMeshWindow.vcxproj.filters
Samples/Geometrics/VertexCollapseMesh/VertexCollapseMeshWindow.cpp
Samples/Geometrics/VertexCollapseMesh/VertexCollapseMesh.cpp
Samples/Graphics/AreaLights/AreaLightsWindow.vcxproj.filters
Samples/Graphics/AreaLights/AreaLightsWindow.cpp
Samples/Graphics/AreaLights/AreaLights.cpp
Samples/Graphics/BillboardNodes/BillboardNodesWindow.vcxproj.filters
Samples/Graphics/BillboardNodes/BillboardNodesWindow.cpp
Samples/Graphics/BillboardNodes/BillboardNodes.cpp
Samples/Graphics/BlendedAnimations/BlendedAnimationsWindow.vcxproj.filters
Samples/Graphics/BlendedAnimations/BlendedAnimationsWindow.cpp
Samples/Graphics/BlendedAnimations/BlendedAnimations.cpp
Samples/Graphics/BlendedTerrain/BlendedTerrainWindow.vcxproj.filters
Samples/Graphics/BlendedTerrain/BlendedTerrainWindow.cpp
Samples/Graphics/BlendedTerrain/BlendedTerrain.cpp

Samples/Graphics/BlownGlass/BlownGlassWindow.vcxproj.filters
Samples/Graphics/BlownGlass/BlownGlassWindow.cpp
Samples/Graphics/BlownGlass/BlownGlass.cpp
Samples/Graphics/BspNodes/BspNodesWindow.vcxproj.filters
Samples/Graphics/BspNodes/BspNodesWindow.cpp
Samples/Graphics/BspNodes/BspNodes.cpp
Samples/Graphics/BumpMaps/BumpMapsWindow.vcxproj.filters
Samples/Graphics/BumpMaps/BumpMapsWindow.cpp
Samples/Graphics/BumpMaps/BumpMaps.cpp
Samples/Graphics/CameraAndLightNodes/CameraAndLightNodesWindow.vcxproj.filters
Samples/Graphics/CameraAndLightNodes/CameraAndLightNodesWindow.cpp
Samples/Graphics/CameraAndLightNodes/CameraAndLightNodes.cpp
Samples/Graphics/Castle/CastleWindow.vcxproj.filters
Samples/Graphics/Castle/CastleWindow.cpp
Samples/Graphics/Castle/Castle.cpp
Samples/Graphics/CubeMaps/CubeMapsWindow.vcxproj.filters
Samples/Graphics/CubeMaps/CubeMapsWindow.cpp
Samples/Graphics/CubeMaps/CubeMaps.cpp
Samples/Graphics/GeometryShaders/GeometryShadersWindow.vcxproj.filters
Samples/Graphics/GeometryShaders/GeometryShadersWindow.cpp
Samples/Graphics/GeometryShaders/GeometryShaders.cpp
Samples/Graphics/GlossMaps/GlossMapsWindow.vcxproj.filters
Samples/Graphics/GlossMaps/GlossMapsWindow.cpp
Samples/Graphics/GlossMaps/GlossMaps.cpp
Samples/Graphics/Lights/LightsWindow.vcxproj.filters
Samples/Graphics/Lights/LightsWindow.cpp
Samples/Graphics/Lights/Lights.cpp
Samples/Graphics/LightTexture/LightTextureWindow.vcxproj.filters
Samples/Graphics/LightTexture/LightTextureWindow.cpp
Samples/Graphics/LightTexture/LightTexture.cpp
Samples/Graphics/MorphFaces/MorphFacesWindow.vcxproj.filters
Samples/Graphics/MorphFaces/MorphFacesWindow.cpp
Samples/Graphics/MorphFaces/MorphFaces.cpp
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargetsWindow.vcxproj.filters
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargetsWindow.cpp
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargets.cpp
Samples/Graphics/Picking/PickingWindow.vcxproj.filters
Samples/Graphics/Picking/PickingWindow.cpp
Samples/Graphics/Picking/Picking.cpp
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersectionWindow.vcxproj.filters
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersectionWindow.cpp
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersection.cpp
Samples/Graphics/ProjectedTextures/ProjectedTexturesWindow.vcxproj.filters
Samples/Graphics/ProjectedTextures/ProjectedTexturesWindow.cpp
Samples/Graphics/ProjectedTextures/ProjectedTextures.cpp
Samples/Graphics/SphereMaps/SphereMapsWindow.vcxproj.filters
Samples/Graphics/SphereMaps/SphereMapsWindow.cpp
Samples/Graphics/SphereMaps/SphereMaps.cpp

Samples/Graphics/StructuredBuffers/StructuredBuffersWindow.vcxproj.filters
Samples/Graphics/StructuredBuffers/StructuredBuffersWindow.cpp
Samples/Graphics/StructuredBuffers/StructuredBuffers.cpp
Samples/Graphics/TextureArrays/TextureArraysWindow.vcxproj.filters
Samples/Graphics/TextureArrays/TextureArraysWindow.cpp
Samples/Graphics/TextureArrays/TextureArrays.cpp
Samples/Graphics/TextureUpdating/TextureUpdatingWindow.vcxproj.filters
Samples/Graphics/TextureUpdating/TextureUpdatingWindow.cpp
Samples/Graphics/TextureUpdating/TextureUpdating.cpp
Samples/Graphics/Texturing/TexturingWindow.vcxproj.filters
Samples/Graphics/Texturing/TexturingWindow.cpp
Samples/Graphics/Texturing/Texturing.cpp
Samples/Graphics/VertexColoring/VertexColoringWindow.vcxproj.filters
Samples/Graphics/VertexColoring/VertexColoringWindow.cpp
Samples/Graphics/VertexColoring/VertexColoring.cpp
Samples/Graphics/VertexTextures/VertexTexturesWindow.vcxproj.filters
Samples/Graphics/VertexTextures/VertexTexturesWindow.cpp
Samples/Graphics/VertexTextures/VertexTextures.cpp
Samples/Graphics/WireMesh/WireMeshWindow.vcxproj.filters
Samples/Graphics/WireMesh/WireMeshWindow.cpp
Samples/Graphics/WireMesh/WireMesh.cpp
Samples/Imagics/Convolution/ConvolutionWindow.vcxproj.filters
Samples/Imagics/Convolution/ConvolutionWindow.cpp
Samples/Imagics/Convolution/Convolution.cpp
Samples/Imagics/GaussianBlurring/GaussianBlurringWindow.vcxproj.filters
Samples/Imagics/GaussianBlurring/GaussianBlurringWindow.cpp
Samples/Imagics/GaussianBlurring/GaussianBlurring.cpp
Samples/Imagics/MedianFiltering/MedianFilteringWindow.vcxproj.filters
Samples/Imagics/MedianFiltering/MedianFilteringWindow.cpp
Samples/Imagics/MedianFiltering/MedianFiltering.cpp
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.vcxproj.filters
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp
Samples/Imagics/SurfaceExtraction/SurfaceExtraction.cpp
Samples/Imagics/VideoStreams/VideoStreamsWindow.vcxproj.filters
Samples/Imagics/VideoStreams/VideoStreamsWindow.cpp
Samples/Imagics/VideoStreams/VideoStreams.cpp
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitterWindow.vcxproj.filters
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitterWindow.cpp
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitter.cpp
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitterWindow.vcxproj.filters
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitterWindow.cpp
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitter.cpp
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.vcxproj.filters
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.cpp
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVs.cpp
Samples/Mathematics/Interpolation2D/Interpolation2DWindow.vcxproj.filters
Samples/Mathematics/Interpolation2D/Interpolation2DWindow.cpp
Samples/Mathematics/Interpolation2D/Interpolation2D.cpp

Samples/Mathematics/PlaneEstimation/PlaneEstimationWindow.vcxproj.filters
Samples/Mathematics/PlaneEstimation/PlaneEstimationWindow.cpp
Samples/Mathematics/PlaneEstimation/PlaneEstimation.cpp
Samples/Physics/BallHill/BallHillWindow.vcxproj.filters
Samples/Physics/BallHill/BallHillWindow.cpp
Samples/Physics/BallHill/BallHill.cpp
Samples/Physics/BallRubberBand/BallRubberBandWindow.vcxproj.filters
Samples/Physics/BallRubberBand/BallRubberBandWindow.cpp
Samples/Physics/BallRubberBand/BallRubberBand.cpp
Samples/Physics/BouncingBall/BouncingBallWindow.vcxproj.filters
Samples/Physics/BouncingBall/BouncingBallWindow.cpp
Samples/Physics/BouncingBall/BouncingBall.cpp
Samples/Physics/Cloth/ClothWindow.vcxproj.filters
Samples/Physics/Cloth/ClothWindow.cpp
Samples/Physics/Cloth/Cloth.cpp
Samples/Physics/DoublePendulum/DoublePendulumWindow.vcxproj.filters
Samples/Physics/DoublePendulum/DoublePendulumWindow.cpp
Samples/Physics/DoublePendulum/DoublePendulum.cpp
Samples/Physics/ExtremalQuery/ExtremalQueryWindow.vcxproj.filters
Samples/Physics/ExtremalQuery/ExtremalQueryWindow.cpp
Samples/Physics/ExtremalQuery/ExtremalQuery.cpp
Samples/Physics/Fluids2D/Fluids2DWindow.vcxproj.filters
Samples/Physics/Fluids2D/Fluids2DWindow.cpp
Samples/Physics/Fluids2D/Fluids2D.cpp
Samples/Physics/Fluids3D/Fluids3DWindow.vcxproj.filters
Samples/Physics/Fluids3D/Fluids3DWindow.cpp
Samples/Physics/Fluids3D/Fluids3D.cpp
Samples/Physics/IntersectingBoxes/IntersectingBoxesWindow.vcxproj.filters
Samples/Physics/IntersectingBoxes/IntersectingBoxesWindow.cpp
Samples/Physics/IntersectingBoxes/IntersectingBoxes.cpp
Samples/Physics/IntersectingRectangles/IntersectingRectanglesWindow.vcxproj.filters
Samples/Physics/IntersectingRectangles/IntersectingRectanglesWindow.cpp
Samples/Physics/IntersectingRectangles/IntersectingRectangles.cpp
Samples/Physics/KeplerPolarForm/KeplerPolarFormWindow.vcxproj.filters
Samples/Physics/KeplerPolarForm/KeplerPolarFormWindow.cpp
Samples/Physics/KeplerPolarForm/KeplerPolarForm.cpp
Samples/Physics/MassSprings3D/MassSprings3DWindow.vcxproj.filters
Samples/Physics/MassSprings3D/MassSprings3DWindow.cpp
Samples/Physics/MassSprings3D/MassSprings3D.cpp
Samples/Physics/Rope/RopeWindow.vcxproj.filters
Samples/Physics/Rope/RopeWindow.cpp
Samples/Physics/Rope/Rope.cpp
Tools/GenerateProject/ProjectTemplate.{v12,v14}.{h,cpp}

Ported the Wild Magic 5 sample application FlowingSkirt.

GTBuildAll.{v12,v14}.sln
Samples/Physics/FlowingSkirt/FlowingSkirt.{v12,v14}.{sln, vcxproj, vcxproj.filters}

Samples/Physics/FlowingSkirt/FlowingSkirtWindow.{h,cpp}

March 14, 2016. Fixed the PDF link in the comments at the beginning of the header file.

ConvertCoordinates.h

March 17, 2016. Fixed several bugs in the test-intersection query for a line segment and a bounding sphere. The replacement code now matches the discussion in *3D Game Engine Design (2nd edition), Section 15.4.3*. The first bug was the computation of segExtent, which works when tmin is zero. For positive tmin, the segment extent is instead $(t_{\max} - t_{\min})/2$. The second bug was that there must be a test $a_0 \leq 0$ immediately after the computation of a_0 (the book mentions this). The third bug was that the final block of code does not match the discussion in the book (and is incorrect); the book description is correct.

BoundingSphere.cpp

Removed the debugCounter test code. Moved the inline body of Vertex::operator< to the end of the file where the other inline functions are implemented (for consistent coding practice).

MinimalCycleBasis.h

March 23, 2016. Of all the OpenGL buffer classes, only GL4VertexBuffer and GL4IndexBuffer were calling glDeleteBuffers to free up the resource handles. Added a destructor to GL4Buffer to call glDeleteBuffers and removed the destructors from the derived classes.

GL4Buffer.{h,cpp}
GL4VertexBuffer.{h,cpp}
GL4IndexBuffer.{h,cpp}

March 28, 2016. The Update, CopyCpuToGpu, and CopyGpuToCpu for DX11Buffer and GL4Buffer had bugs when the buffer had a positive offset. The various functions and data types involved require values set in terms of number of bytes when the resource is a buffer. Added a new sample application to demonstrate how to update only a portion of a buffer.

DX11Buffer.cpp
GL4Buffer.cpp
GTBuildAll.{v12,v14}.sln
Samples/Graphics/BufferUpdating/BufferUpdating.{v12,v14}.{sln, vcxproj, vcxproj.filters}
Samples/Graphics/BufferUpdating/BufferUpdating.{h,cpp}

April 1, 2016. Dynamic constant buffers in Direct3D 11.0 cannot be mapped using D3D11_MAP_WRITE_NO_OVERWRITE, but they can in Direct 3D 11.1. The MSDN web page for D3D11_MAP has a note about this and suggests how to test whether no-overwrite may be used by calling CheckFeatureSupport on the device with feature D3D11_FEATURE_D3D11_OPTIONS. Unfortunately, the documentation for D3D11_FEATURE_D3D11_OPTIONS states that this option may only be used for Direct3D 11.1 and later. This mechanism fails on an NVIDIA Quadro K2200 (driver 362.13 and previous). A call to the device GetFeatureLevel returns D3D_FEATURE_LEVEL_11_0 but a call to CheckFeatureSupport shows that MapNoOverwriteOnDynamicConstantBuffer is 1 (so no-overwrite is supposed to be allowed). Unfortunately, this appears to cause problems in rendering. Worse is that our text rendering (TextEffect) uses dynamic vertex buffers and has strange behavior when using no-overwrite. All text renders correctly with the discard mode. For now, we have added a member DX11Buffer::mUpdateMapMode whose default value is D3D11_MAP_WRITE_DISCARD but is set to D3D11_MAP_WRITE_NO_OVERWRITE when the feature level is found to be D3D_FEATURE_LEVEL_11_1 or later.

DX11Buffer.{h,cpp}
DX11ConstantBuffer.cpp
DX11IndexBuffer.cpp
DX11IndirectArgumentsBuffer.cpp
DX11RawBuffer.cpp
DX11StructuredBuffer.cpp
DX11TextureBuffer.cpp
DX11VertexBuffer.cpp

April 2, 2016. Updated the cycle-basis algorithm based on the revised discussion in the PDF.

MinimalCycleBasis.h
Samples/Geometrics/MinimalCycleBasis/MinimalCycleBasisWindow.{vcxproj,vcxproj.filters}
Samples/Geometrics/MinimalCycleBasis/MinimalCycleBasisWindow.{h,cpp}
Samples/Geometrics/MinimalCycleBasis/Data/SimpleGraph{0,1,2,3,4,5}.txt

71 Updates to Version 2.2

January 30, 2016. Delete the sample application Samples/Basics/PerformanceAMD and the associated tools folder Tools/GPUPerfAPI-2.11.739.0.

January 31, 2016. Replaced type int by int32_t for consistent notation in the class.

GteUIntegerALU32.h

Added a tool to illustrate the use of BSPrecision. Currently, it computes the N-values for the primal queries which determine the N-values for construction of convex hulls and Delaunay triangulations. Updated the N-values for computing with BSNumber or BSRational.

GTBuildAll.{v12,v14}.sln
Tools/Imagics/PrecisionCalculator/PrecisionCalculator.{v12,v14}.{sln,vcxproj,vcxproj.filters}

Tools/PrecisionCalculator/PrecisionCalculator.cpp
GtePrimalQuery{2,3}.h
GteConvexHull{2,3}.h
GteDelaunayHull{2,3}.h

Moved an assertion inside the GTE_DEV_OPENGL block.

Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp

Removed the Convertfunctions. C++ 11 already supports iterated-based copying between objects of type std::string and std::wstring.

Environment.{h,cpp}
LogToMessageBox.cpp
LogToOutputWindow.cpp

Replaced raw new and delete calls by vector, shared pointer, or unique pointer wrappers. Replaced the application layer engine and factory objects with shared pointers of the base-class types. This supports refactoring the DX11 and OpenGL graphics engine into separate projects.

Environment.{h,cpp}
WICFileIO.cpp
DX11Engine.cpp
HLSLProgramFactory.cpp
GL4Engine.cpp
IndexBuffer.h
Window.{h,cpp}
Window3.cpp
WindowSystem.cpp
LogReporter.{h,cpp} AmbientLightEffect.{h,cpp}
ConstantColorEffect.{h,cpp}
DirectionalLightEffect.{h,cpp}
DirectionalLightTextureEffect.{h,cpp}
Font.{h,cpp}
FontArialW400H18.{h,cpp}
LightingEffect.{h,cpp}
OverlayEffect.{h,cpp}
PlanarReflectionEffect.{h,cpp}
PointLightEffect.{h,cpp}
PointLightTextureEffect.{h,cpp}
SpotLightEffect.{h,cpp}
TextEffect.{h,cpp}
Texture2Effect.{h,cpp}
Texture3Effect.{h,cpp}

VertexColorEffect.{h,cpp}
MeshFactory.cpp
ComputeModel.h
GenerateMeshUV.h
Fluid{2,3}.{h,cpp}
Fluid{2,3}AdjustVelocity.{h,cpp}
Fluid{2,3}ComputeDivergence.{h,cpp}
Fluid{2,3}EnforceStateBoundary.{h,cpp}
Fluid{2,3}InitializeSource.{h,cpp}
Fluid{2,3}InitializeState.{h,cpp}
Fluid{2,3}SolvePoisson.{h,cpp}
Fluid{2,3}UpdateState.{h,cpp}
Samples/Geometrics/AllPairsTriangles/AllPairsTrianglesWindow.cpp
Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow.cpp
Samples/Geometrics/Delaunay3D/Delaunay3DWindow.cpp
Samples/Geometrics/DistanceSegments3/DistanceSegments3.cpp
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.cpp
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3D.cpp
Samples/Geometrics/ShortestPath/ShortestPathWindow.cpp
Samples/Geometrics/ShortestPath/GpuShortestPath.{h,cpp}
Samples/Geometrics/VertexCollapseMesh/VertexCollapseMeshWindow.cpp
Samples/Graphics/BlendedAnimations/BipedManager.{h,cpp}
Samples/Graphics/BlendedTerrain/BlendedTerrainEffect.{h,cpp}
Samples/Graphics/BlownGlass/BlownGlassWindow.cpp
Samples/Graphics/BumpMaps/SimpleBumpMapEffect.{h,cpp}
Samples/Graphics/CubeMaps/CubeMapEffect.{h,cpp}
Samples/Graphics/GeometryShaders/GeometryShadersWindow.cpp
Samples/Graphics/GlossMaps/GlossMapEffect.{h,cpp}
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargetsWindow.cpp
Samples/Graphics/Picking/PickingWindow.cpp
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersectionWindow.cpp
Samples/Graphics/ProjectedTextures/ProjectedTextureEffect.{h,cpp}
Samples/Graphics/SphereMaps/SphereMapEffect.{h,cpp}
Samples/Graphics/StructuredBuffers/StructuredBuffersWindow.cpp
Samples/Graphics/TextureArrays/TextureArraysWindow.cpp
Samples/Graphics/VertexTextures/DisplacementEffect.{h,cpp}
Samples/Graphics/WireMesh/WireMeshWindow.cpp
Samples/Imagics/Convolution/ConvolutionWindow.cpp
Samples/Imagics/GaussianBlurring/GaussianBlurringWindow.cpp
Samples/Imagics/MedianFiltering/MedianFilteringWindow.cpp
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp
Samples/Imagics/VideoStreams/FileVideoStream.{h,cpp}
Samples/Imagics/VideoStreams/VideoStream.{h,cpp}
Samples/Imagics/VideoStreams/VideoStreamsWindow.cpp
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitterWindow.cpp
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.cpp
Samples/Mathematics/PlaneEstimation/PlaneEstimationWindow.cpp
Samples/Physics/Cloth/ClothWindow.cpp

Samples/Physics/Fluids2D/Fluids2DWindow.cpp
Samples/Physics/Fluids3D/Fluids3DWindow.cpp
Samples/Physics/MassSprings3D/MassSprings3DWindow.cpp
Samples/Physics/MassSprings3D/GpuMassSpringVolume.{h,cpp}
Samples/Physics/Rope/RopeWindow.cpp
Tools/GenerateOpenGLWrapper/Initialize.txt
Tools/GenerateOpenGLWrapper/Version.txt
OpenGL.cpp

Window creation and destruction now involve shared pointers rather than raw pointers.

Window.h
Samples/Geometrics/AllPairsTriangles/AllPairsTriangles.cpp
Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2D.cpp
Samples/Geometrics/ConvexHull2D/ConvexHull2D.cpp
Samples/Geometrics/ConvexHull3D/ConvexHull3D.cpp
Samples/Geometrics/Delaunay2D/Delaunay2D.cpp
Samples/Geometrics/Delaunay3D/Delaunay3D.cpp
Samples/Geometrics/IntersectBoxCone/IntersectBoxCone.cpp
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2D.cpp
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2D.cpp
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3D.cpp
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3D.cpp
Samples/Geometrics/ShortestPath/ShortestPath.cpp
Samples/Geometrics/TriangulationCDT/TriangulationCDT.cpp
Samples/Geometrics/TriangulationEC/TriangulationEC.cpp
Samples/Geometrics/VertexCollapseMesh/VertexCollapseMesh.cpp
Samples/Graphics/BillboardNodes/BillboardNodes.cpp
Samples/Graphics/BlendedAnimations/BlendedAnimations.cpp
Samples/Graphics/BlendedTerrain/BlendedTerrain.cpp
Samples/Graphics/BrownGlass/BrownGlass.cpp
Samples/Graphics/BspNodes/BspNodes.cpp
Samples/Graphics/BumpMaps/BumpMaps.cpp
Samples/Graphics/CameraAndLightNodes/CameraAndLightNodes.cpp
Samples/Graphics/Castle/Castle.cpp
Samples/Graphics/CubeMaps/CubeMaps.cpp
Samples/Graphics/GeometryShaders/GeometryShaders.cpp
Samples/Graphics/GlossMaps/GlossMaps.cpp
Samples/Graphics/Lights/Lights.cpp
Samples/Graphics/LightTexture/LightTexture.cpp
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargets.cpp
Samples/Graphics/Picking/Picking.cpp
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersection.cpp
Samples/Graphics/ProjectedTextures/ProjectedTextures.cpp
Samples/Graphics/SharedTextures/SharedTextures.cpp
Samples/Graphics/SharedTextures/SharedTexturesWindow.cpp
Samples/Graphics/SphereMaps/SphereMaps.cpp

Samples/Graphics/StructuredBuffers/StructuredBuffers.cpp
Samples/Graphics/TextureArrays/TextureArrays.cpp
Samples/Graphics/TextureUpdating/TextureUpdating.cpp
Samples/Graphics/Texturing/Texturing.cpp
Samples/Graphics/VertexColoring/VertexColoring.cpp
Samples/Graphics/VertexTextures/VertexTextures.cpp
Samples/Graphics/WireMesh/WireMesh.cpp
Samples/Imagics/Convolution/Convolution.cpp
Samples/Imagics/GaussianBlurring/GaussianBlurring.cpp
Samples/Imagics/MedianFiltering/MedianFiltering.cpp
Samples/Imagics/SurfaceExtraction/SurfaceExtraction.cpp
Samples/Imagics/VideoStreams/VideoStreams.cpp
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitter.cpp
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitter.cpp
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVs.cpp
Samples/Mathematics/Interpolation2D/Interpolation2D.cpp
Samples/Mathematics/PlaneEstimation/PlaneEstimation.cpp
Samples/Physics/BallHill/BallHill.cpp
Samples/Physics/BallRubberBand/BallRubberBand.cpp
Samples/Physics/BouncingBall/BouncingBall.cpp
Samples/Physics/Cloth/Cloth.cpp
Samples/Physics/DoublePendulum/DoublePendulum.cpp
Samples/Physics/ExtremalQuery/ExtremalQuery.cpp
Samples/Physics/Fluids2D/Fluids2D.cpp
Samples/Physics/Fluids3D/Fluids3D.cpp
Samples/Physics/IntersectingBoxes/IntersectingBoxes.cpp
Samples/Physics/IntersectingRectangles/IntersectingRectangles.cpp
Samples/Physics/KeplerPolarForm/KeplerPolarForm.cpp
Samples/Physics/MassSprings3D/MassSprings3D.cpp
Samples/Physics/Rope/Rope.cpp
Tools/GenerateProject/ProjectTemplate.{v12,v14}.cpp

February 1, 2016. Renamed Array2 to LexicoArray2. To avoid the raw new and delete calls in the memory management functions of GteMemory.h, those functions have been converted to classes (to have state). The new classes are Array2, Array3, and Array4.

LexicoArray2.h
Array2.h (the old file)
BandedMatrix.h
GaussianElimination.h
LinearSystem.h
Array2.h (the new file)
Array3.h
Array4.h

Replaced raw new and delete calls by vector, shared pointer, or unique pointer wrappers, or by Array2, Array3, or Array4. Removed the old memory management.

GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
Memory.h
GTLowLevel.h
Histogram.{h,cpp}
ApprPolynomial{2,3,4}.h
BasisFunction.h
BezierCurve.h
BSplineCurveFit.h
BSplineSurfaceFit.h
GaussianElimination.h
IntpVectorField2.h
IntpSphere2.h
IntpAkimaUniform{2,3}.h
KeyframeController.{h,cpp}
SkinController.{h,cpp}
Samples/Graphics/BlendedAnimations/BipedManager.cpp
Samples/Graphics/SharedTexturesD3D11/SharedTexturesWindow.{h,cpp}
Samples/Imagics/VideoStreams/VideoStreamManager.{h,cpp}
Samples/Imagics/VideoStreams/VideoStreamsWindow.{h,cpp}
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitterWindow.{h,cpp}
Samples/Mathematics/Interpolation2D/Interpolation2DWindow.cpp
Samples/Physics/Cloth/ClothWindow.cpp
Samples/Physics/Rope/RopeWindow.cpp
Tools/BitmapFontCreator/BitmapFontCreator.cpp

February 2, 2016. The class Image2 was used to provide a 2-dimensional wrapper around the Weights texture using the data-pointer-sharing mechanism of Image. Replaced this by Array2 so that we can remove the sharing mechanism in Image to simplify the semantics of the class. (The sample application is the only consumer of Image sharing mechanism.)

Samples/Geometrics/ShortestPath/ShortestPathWindow.h
Samples/Geometrics/ShortestPath/CpuShortestPath.{h,cpp}

The Image classes had a lot of infrastructure for metadata and for accessing image values using the syntax myImage2[][] and myImage3[][][]. The infrastructure was added in support of some specific applications not part of the source code distribution. The infrastructure has been removed and the classes greatly simplified. File I/O was removed so that PixelType is no longer required to be plain-old-data (POD). If you need to load and save data, you can roll your own depending on the nature of PixelType.

Histogram.{h,cpp}
GTImagics.h
Image.h
Image2.h

Image3.h
Image.cpp
Image1.h

Removed unused header file (for GMatrix).

MinimumAreaCircle2.h

February 3, 2016. Replaced raw new and delete calls by vector, shared pointer, or unique pointer wrappers.

ImageUtility{2,3}.{h,cpp}
Samples/Graphics/BlendedAnimations/BipedManager.{h,cpp}

Fixed memory/object leak in the controller system. The ControlledObject class needed to own the Controller objects in the list. Also, clarified comments about why std::weak_ptr cannot be used to avoid reference-count cycles when the scene graph is created internally rather than by an external manager (internally, we do not know the owning shared pointer objects from which we can generate weak pointers).

ControlledObject.{h,cpp}
Controller.{h,cpp}
Spatial.{h,cpp}

The bone array now uses std::weak_ptr<Node> rather than Node* to avoid the reference-count cycles in the scene graph.

SkinController.{h,cpp}

Removed the dynamic deletion code in TriangulateEC and TriangulateCDT, replaced the Polygon by a typedef to std::vector, and added a constructor that takes a std::vector of points. This is part of the work to remove raw new and delete calls from the engine.

TriangulateEC.h
TriangulateCDT.h
Samples/Geometrics/TriangulationEC/TriangulationECWindow.{h,cpp}
Samples/Geometrics/TriangulationCDT/TriangulationCDTWindow.cpp

February 4, 2016. Removed the raw new and delete calls in the manifold mesh classes, replacing them with std::shared_ptr and std::weak_ptr wrappers.

Delaunay{2,3}.h
ETManifoldMesh.{h,cpp}
ExtremalQuery3BSP.h
GenerateMeshUV.h
MinimumVolumeBox3.h
PlanarMesh.h
TSManifoldMesh.{h,cpp}
VEManifoldMesh.{h,cpp}
VETManifoldMesh.{h,cpp}
VertexCollapseMesh.h

Modified WICFileIO to use std::shared_ptr rather than raw new and delete calls.

WICFileIO.{h,cpp}
Samples/Graphics/BillboardNodes/BillboardNodesWindow.cpp
Samples/Graphics/BlendedAnimations/BipedManager.cpp
Samples/Graphics/BlendedAnimations/BlendedAnimationsWindow.cpp
Samples/Graphics/BlendedTerrain/BlendedTerrainWindow.cpp
Samples/Graphics/BlendedTerrain/BlendedTerrainEffect.cpp
Samples/Graphics/BspNodes/BspNodesWindow.cpp
Samples/Graphics/BumpMaps/BumpMapsWindow.cpp
Samples/Graphics/BumpMaps/SimpleBumpMapEffect.cpp
Samples/Graphics/CameraAndLightNodes/CameraAndLightNodesWindow.cpp
Samples/Graphics/Castle/CastleWindow.cpp
Samples/Graphics/CubeMaps/CubeMapsWindow.cpp
Samples/Graphics/GeometryShaders/GeometryShadersWindow.cpp
Samples/Graphics/GlossMaps/GlossMapsWindow.cpp
Samples/Graphics/LightTexture/LightTextureWindow.cpp
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargetsWindow.cpp
Samples/Graphics/Picking/PickingWindow.cpp
Samples/Graphics/ProjectedTextures/ProjectedTexturesWindow.cpp
Samples/Graphics/SphereMaps/SphereMapsWindow.cpp
Samples/Graphics/StructuredBuffers/StructuredBuffersWindow.cpp
Samples/Graphics/TextureArrays/TextureArraysWindow.cpp
Samples/Graphics/Texturing/TexturingWindow.cpp
Samples/Graphics/VertexColoring/VertexColoringWindow.cpp
Samples/Graphics/VertexTextures/VertexTexturesWindow.cpp
Samples/Imagics/Convolution/ConvolutionWindow.cpp
Samples/Imagics/GaussianBlurring/GaussianBlurringWindow.cpp
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitterWindow.cpp
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.cpp
Samples/Mathematics/Interpolation2D/Interpolation2DWindow.cpp
Samples/Physics/BallHill/BallHillWindow.cpp
Samples/Physics/BouncingBall/BouncingBallWindow.cpp
Samples/Physics/Cloth/ClothWindow.cpp
Samples/Physics/Fluids2D/Fluids2DWindow.cpp
Samples/Physics/Rope/RopeWindow.cpp

Modified the graphics engine code to eliminate raw new and delete calls.

GraphicsEngine.{h,cpp}
DX11BlendState.{h,cpp}
DX11ComputeShader.{h,cpp}
DX11ConstantBuffer.{h,cpp}
DX11DepthStencilState.{h,cpp}
DX11DrawTarget.{h,cpp}
DX11Engine.cpp
DX11GeometryShader.{h,cpp}
DX11GraphicsObject.h
DX11IndexBuffer.{h,cpp}
DX11IndirectArgumentsBuffer.{h,cpp}
DX11InputLayoutManager.{h,cpp}
DX11PixelShader.{h,cpp}
DX11RasterizerState.{h,cpp}
DX11RawBuffer.{h,cpp}
DX11SamplerState.{h,cpp}
DX11StructuredBuffer.{h,cpp}
DX11Texture1.{h,cpp}
DX11Texture1Array.{h,cpp}
DX11Texture2.{h,cpp}
DX11Texture2Array.{h,cpp}
DX11Texture3.{h,cpp}
DX11TextureBuffer.{h,cpp}
DX11TextureCube.{h,cpp}
DX11TextureCubeArray.{h,cpp}
DX11TextureDS.{h,cpp}
DX11TextureRT.{h,cpp}
DX11VertexBuffer.{h,cpp}
DX11VertexShader.{h,cpp}
GL4AtomicCounterBuffer.{h,cpp}
GL4BlendState.{h,cpp}
GL4ConstantBuffer.{h,cpp}
GL4DepthStencilState.{h,cpp}
GL4DrawTarget.{h,cpp}
GL4Engine.cpp
GL4IndexBuffer.{h,cpp}
GL4InputLayoutManager.{h,cpp}
GL4RasterizerState.{h,cpp}
GL4SamplerState.{h,cpp}
GL4StructuredBuffer.{h,cpp}
GL4Texture1.{h,cpp}
GL4Texture1Array.{h,cpp}
GL4Texture2.{h,cpp}
GL4Texture2Array.{h,cpp}

GL4Texture3.{h,cpp}
GL4TextureCube.{h,cpp}
GL4TextureCubeArray.{h,cpp}
GL4TextureDS.{h,cpp}
GL4TextureRT.{h,cpp}
GL4VertexBuffer.{h,cpp}

Modified projects so that DX11 and GL4 are mutually exclusive, even though it is possible to create both types of engines in an application.

GTEngine.{v12,v14}.vcxproj
ComputeShader.{h,cpp}
GeometryShader.{h,cpp}
PixelShader.{h,cpp}
VertexShader.{h,cpp}
Shader.{h,cpp}

February 6, 2016. Fixed a typographical error in a preprocessor macro used in the creation of a perspective projection matrix.

Matrix4x4.h

February 8, 2016. Removed an unused header file.

MinHeap.h

February 16, 2016. The interval intersection query results needed to be propagated to the query results for ray-circle and segment-circle.

IntrRay2Circle2.h
IntrSegment2Circle2.h

February 25, 2016. Removed unnecessary include of `array`.

UniqueVerticesTriangle.h

February 26, 2016. `Window3::OnResize` need to return `true` in its conditional statement.

Window3.cpp

February 27, 2016. Added missing cases when the intersection of two co-circular arcs contains two disjoint components (two arcs, one arc and one point, or two points). Added internal unit tests for 100 % code coverage and verification of correctness.

IntrArc2Arc2.h

March 1, 2016. The specular term was incorrectly coded in GetShaderSourceLitFunctionGLSL.

LightingEffect.cpp

New sample application to illustrate area lights.

GTBuildAll.{v12,v14}.sln
Samples/Graphics/AreaLights/AreaLights.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/AreaLights/AreaLights.cpp
Samples/Graphics/AreaLights/AreaLightsWindow.{h,cpp}
Samples/Graphics/AreaLights/AreaLightEffect.{h,cpp}
Samples/Graphics/AreaLights/Shaders/AreaLight.hlsl
Samples/Graphics/AreaLights/Shaders/AreaLight{VS,PS}.glsl

March 2, 2016. The port of the find-intersection query from Wild Magic was missing the assignment of the circle radius and circle plane-normal when there is a circle of intersection.

IntrPlane3Sphere3.h

March 4, 2016. The first-order derivative in the w-variable was computed incorrectly because inputs vOrder and wOrder were swapped.

NURBSVolume.h

March 6, 2016. Removed the obsolete include of the GteEnvironment header.

GteLogToOutputWindow.cpp
GteLogToMessageBox.cpp

Fixed compile errors when precompiled headers are turned off.

GTEnginePCH.h
WindowSystem.cpp

DX11Engine.cpp
GL4DrawTarget.cpp
GL4Engine.cpp
GL4SamplerState.cpp
GL4StructuredBuffer.cpp
GL4TextureArray.cpp
GL4TextureDS.cpp
GL4TextureRT.cpp
GL4TextureSingle.cpp
GLSLReflection.cpp
Fluid{2,3}AdjustVelocity.cpp
Fluid{2,3}ComputeDivergence.cpp
Fluid{2,3}EnforceStateBoundary.cpp
Fluid{2,3}InitializeSource.cpp
Fluid{2,3}InitializeState.cpp
Fluid{2,3}SolvePoisson.cpp
Fluid{2,3}UpdateState.cpp

Ported the Wild Magic 5 sample for computing cycle bases. The original code was flawed, so this sample includes a major rewrite of the algorithm and documentation. The code has had significant testing.

GteIsPlanarGraph.h
GteMinimalCycleBasis.h
GTBuildAll.{v12,v14}.sln
GTMathematics.h
Samples/Geometrics/MinimalCycleBasis/MinimalCycleBasis.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Geometrics/MinimalCycleBasis/MinimalCycleBasis.cpp
Samples/Geometrics/MinimalCycleBasis/MinimalCycleBasis.{h,cpp}
Documentation/MinimalCycleBasis.pdf
MinimalCycleBasis.pdf (*Constructing a Cycle Basis for a Planar Graph*)

72 Updates to Version 2.1

January 26, 2016. Added a workaround for an apparent OpenGL bug in the Intel HD 4600 graphics drivers. Frequently, shader storage blocks that are used in shaders are reported as unreferenced by the shaders. The workaround is effectively to parse the shader source code and determine whether in fact the storage block is referenced. (A bug report has been filed with Intel.)

GLSLReflection.{h,cpp}

Modified the shaders to link correctly when using Intel HD 4600 OpenGL 4.3. Moved the shaders to a subfolder, the pattern used in other sample applications.

Samples/Basics/ShaderReflection/Shaders/
Samples/Basics/ShaderReflection/Billboards{VS,GS,PS}.glsl

Removed an orphan reference to a file (GteDistLine3Cylinder3.h) in the projects. The reference was causing the projects to think they were out-of-date even though they were not.

GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}

The noperspective tag for edgeDistance in the geometry shader was not also used in the pixel shader. NVIDIA and AMD drivers are fine with this, because the pixel shader is called after the interpolation has occurred. The Intel HD OpenGL 4.3 complained during linking that it could not match edgeDistance between the geometry and pixel shaders without both matching exactly in all attributes.

Samples/Graphics/WireMesh/Shaders/WireMeshPS.glsl

January 30, 2016. Added a preprocessor wrapper around the include of GTEngine.h so that you can disable the precompile header system. Without precompiled headers, the compilation of the engine and samples is extremely slow. However, until the GTEngine projects are decomposed into smaller libraries, enabling precompiled headers leads to all source code compiled (on Win32 machines) including DX11 graphics code, OpenGL graphics code (if enabled), and the application layer.

GTEnginePCH.h

73 Updates to Version 2.0

September 27, 2015. Ported BallHill physics sample from WM5 to GTE2.

GTBuildAll.v12.sln
GTBuildAll.v14.sln
Samples/Physics/BallHill/BallHill.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/BallHill/BallHill.cpp
Samples/Physics/BallHill/BallHillWindow.{h,cpp}
Samples/Physics/BallHill/PhysicsModule.{h,cpp}

September 28, 2015. Added Window2 to share code among 2D windowed applications. Various projects were modified accordingly. Added WaitMessage calls to the message pump to allow the thread to be suspended until messages are actually received.

Window2.{h,cpp}
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}

GTApplications.h
Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2D.cpp
Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2DWindow.{h,cpp}
Samples/Geometrics/ConvexHull2D/ConvexHull2D.cpp
Samples/Geometrics/ConvexHull2D/ConvexHull2DWindow.{h,cpp}
Samples/Geometrics/Delaunay2D/Delaunay2D.cpp
Samples/Geometrics/Delaunay2D/Delaunay2DWindow.{h,cpp}
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2D.cpp
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.{h,cpp}
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2D.cpp
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2DWindow.{h,cpp}
Samples/Geometrics/TriangulationCDT/TriangulationCDT.cpp
Samples/Geometrics/TriangulationCDT/TriangulationCDTWindow.{h,cpp}
Samples/Geometrics/TriangulationEC/TriangulationEC.cpp
Samples/Geometrics/TriangulationEC/TriangulationECWindow.{h,cpp}

Removed some white space and added C++ keyword `override` to virtual functions.

Window3.{h,cpp}

Moved HLSL files to the shader folders of the project (Visual Studio bug that allows you add HLSL files to a folder, but when you save the project those files show up outside the folder).

Samples/Basics/ShaderReflection/ShaderReflection.v14.vcxproj.filters
Samples/Geometrics/AllPairsTriangles/AllPairsTriangles.v14.vcxproj.filters
Samples/Geometrics/ShortestPath/ShortestPath.v14.vcxproj.filters
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersection.v14.vcxproj.filters
Samples/Imagics/Convolution/Convolution.v14.vcxproj.filters
Samples/Imagics/MedianFiltering/MedianFiltering.v14.vcxproj.filters
Samples/Mathematics/PlaneEstimation/PlaneEstimation.v14.vcxproj.filters
Samples/Physics/MassSprings3D/MassSprings3D.v14.vcxproj.filters

September 29, 2015. Ported BallRubberBand physics sample from WM5 to GTE2. Added test for `Window2` to GTVerify and fixed header issues exposed when precompiled headers are turned off.

GTBuildAll.{v12,v14}.sln
Samples/Physics/BallRubberBand/BallHill.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/BallRubberBand/BallRubberBand.cpp
Samples/Physics/BallRubberBand/BallRubberBandWindow.{h,cpp}
Samples/Physics/BallRubberBand/PhysicsModule.{h,cpp}

September 30, 2015. Ported BeadSlide physics sample from WM5 to GTE2.

GTBuildAll.{v12,v14}.sln
Samples/Physics/BallSlide/BallSlide.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/BallSlide/BallSlide.cpp
Samples/Physics/BallSlide/PhysicsModule.{h,cpp}

October 8, 2015. Replaced VisibleSet by a std::vector container. Added more variations of DX11Engine::Draw to support native Visual pointers. Modified the DX11Engine::CreateDevice to loop over each allowable feature level, searching for a feature level that the adapter supports. This avoids the special-case behavior when DX11.1 is requested and the device does not support it.

GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
DX11Engine.{h,cpp}
Culler.{h,cpp}
Samples/Graphics/BillboardNodes/BillboardNodesWindow.cpp
Samples/Graphics/BspNodes/BspNodesWindow.cpp
Samples/Graphics/Castle/CastleWindow.cpp
VisibleSet.{h,cpp}

Updated Marching Cubes algorithm to be more efficient. Ported BouncingBall physics sample from WM5 to GTE2.

GTBuildAll.{v12,v14}.sln
MarchingCubes.{h,cpp}
SurfaceExtractor.h
UniqueVerticesTriangles.h
Samples/Data/Floor.png
Samples/Data/Wall1.png
Samples/Physics/BouncingBall/BouncingBall.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/BouncingBall/BouncingBall.cpp
Samples/Physics/BouncingBall/BouncingBallWindow.{h,cpp}
Samples/Physics/BouncingBall/DeformableBall.{h,cpp}

Encountered a custom OpenGL implementation for which glGetIntegerv does not set the major and minor versions to zero. The version numbers need to be initialized before the calls.

Tools/GenerateOpenGLWrapper/Version.txt

October 11, 2015. Added function GetGTEPath to encapsulate the common code used in sample applications to query for the GTE_PATH environment variable.

Windows.{h,cpp}

Factored out the pvw-matrix updating system from CameraRig to a separate class PVWUpdater. The camera rig is a convenience for sample applications but the pvw-updater is a more general concept that can be used in applications written by others.

GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GTGraphics.h
CameraRig,{h,cpp}
Windows3.{h,cpp}
PVWUpdater.{h,cpp}

A large number of sample applications were modified to use the last two changes. Also, added override modifiers to the derived-class virtual functions.

October 17, 2015. Modified the signature of the Draw functions to remove const modifiers on the Visual objects.

DX11Engine.{h,cpp}
GL4Engine.{h,cpp}

Added functions to construct projection and reflection matrices.

Matrix4x4.h

Ported a global effect from Wild Magic 5, PlanarReflectionEffect, and added it to the BouncingBall sample physics application just like WM5 had.

PlanarReflectionEffect.{h,cpp}
GTGraphics.h
Samples/Physics/BouncingBall/BouncingBallWindow.{h,cpp}

October 18, 2015. Ported LCPSolver, LCPPolyDist, and BouncingSpheres from Wild Magic 5 to GTEngine2.

GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GTPhysics.h
LCPSolver.{h,cpp}
LCPPolyDist.h
Samples/Physics/BouncingSpheres/BouncingSpheres.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/BouncingSpheres/BouncingSpheres.cpp
Samples/Physics/BouncingSpheres/BouncingSpheresWindow.{h,cpp}
Samples/Physics/BouncingSpheres/RigidBall.{h,cpp}

October 20, 2015. Ported IntersectingRectangles from Wild Magic 5 to GTEngine2.

GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GTPhysics.h
RectangleManager.h
Samples/Physics/IntersectingRectangles/IntersectingRectangles.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/IntersectingRectangles/IntersectingRectangles.cpp
Samples/Physics/IntersectingRectangles/IntersectingRectanglesWindow.{h,cpp}

Added get/set support for 3×3 rotation matrices.

Transform.{h,cpp}

October 21, 2015. Specifying std::ios::in — std::ios::binary for std::ifstream objects of std::ios::out — std::ios::binary for std::ofstream objects is not necessary. The modifier std::ios::binary is all that is necessary.

Image.cpp
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.cpp
Samples/Graphics/BlendedAnimations/BipedManager.cpp
Samples/Graphics/BlownGlass/BlownGlassWindow.cpp
Samples/Imagics/VideoStreams/VideoStreamsWindow.cpp

Ported IntersectingBoxes from Wild Magic 5 to GTEngine2.

GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GTPhysics.h
BoxManager.h
Samples/Physics/IntersectingBoxes/IntersectingBoxes.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/IntersectingBoxes/IntersectingBoxes.cpp
Samples/Physics/IntersectingBoxes/IntersectingBoxesWindow.{h,cpp}

October 22, 2015. Ported DoublePendulum from Wild Magic 5 to GTEngine2.

GTBuildAll.{v12,v14}.sln
Samples/Physics/DoublePendulum/DoublePendulum.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/DoublePendulum/DoublePendulum.cpp
Samples/Physics/DoublePendulum/DoublePendulumWindow.{h,cpp}
Samples/Physics/DoublePendulum/PhysicsModule.{h,cpp}

Ported KeplerPolarForm from Wild Magic 5 to GTEngine2.

GTBuildAll.{v12,v14}.sln
Samples/Physics/KeplerPolarForm/KeplerPolarForm.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/KeplerPolarForm/KeplerPolarForm.cpp
Samples/Physics/KeplerPolarForm/KeplerPolarFormWindow.{h,cpp}
Samples/Physics/KeplerPolarForm/PhysicsModule.{h,cpp}

October 25, 2015. Fixed comments, replaced std::abs by fabs, and fixed validation code in constructor for number of indices required for polyhedron.

Polygon2.h
Polyhedron3.h

November 1, 2015. Added find-intersection query for oriented boxes in 2D.

IntrOrientedBox2OrientedBox2.h

November 15, 2015. Implemented a vertex-edge-triangle manifold mesh class VETManifoldMesh that derives from the edge-triangle ETManifoldMesh class. The new class provides adjacency information at each vertex of the mesh. This is a replacement for the hacked BasicMesh of Wild Magic 5.

VETManifoldMesh.{h,cpp}
ETManifoldMesh.{h,cpp}
GTMathematics.h
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}

Ported the extremal queries for convex polyhedra and the extremal query sample from Wild Magic 5.

ExtremalQuery3.h
ExtremalQuery3BSP.h
ExtremalQuery3PRJ.h
GTPhysics.h
GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
Samples/Physics/ExtremalQuery/ExtremalQuery.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/ExtremalQuery/ExtremalQuery.cpp
Samples/Physics/ExtremalQuery/ExtremalQuery.{h,cpp}

Fixed a bug in Environment::GetPath. The read/write constants passed to _access_s were reversed.

Environment.cpp

Fixed a compile error when developing using OpenGL. The signature of the Draw function had been modified.

PlanarReflectionEffect.cpp

November 18, 2015. The classes were missing implementations of member accessors.

DX11TextureSingle.cpp
DX11TextureArray.cpp

Added subresource index support for texture arrays, texture cubes, and texture cube array.

TextureArray.h
TextureCubeArray.h

Added memory copy functions to DX11Engine for texture cubes.

DX11Engine.{h,cpp}

Fixed a bug in the shader reflection code. The texture cube objects were being classified as single textures but needed to be classified as texture arrays.

HLSLShaderFactory.cpp
HLSLTexture.cpp
HLSLTextureArray.cpp

November 23, 2015. New OpenGL code and corresponding modifications and/or fixes in the DX11 code.

GTBuildAll.{v12,v14}.{sln,vcxproj,vcxproj.filters}
DataFormat.h
OverlayEffect.{h,cpp}
Shader.{h,cpp}
TextureArray.h
TextEffect.cpp
Texture2Effect.cpp
TextureCube.cpp
TextureCubeArray.cpp
DX11Engine.{h,cpp}
GTGraphicsGL4.h
GL4Buffer.cpp
GL4Engine.{h,cpp}
GL4Resource.cpp

GL4Texture.{h,cpp}
GLSLProgramFactory.cpp
GLSLReflection.cpp
GL4SamplerState.{h,cpp}
GL4Texture1.{h,cpp}
GL4Texture1Array.{h,cpp}
GL4Texture2.{h,cpp}
GL4Texture2Array.{h,cpp}
GL4Texture3.{h,cpp}
GL4TextureArray.{h,cpp}
GL4TextureCube.{h,cpp}
GL4TextureCubeArray.{h,cpp}
GL4TextureSingle.{h,cpp}

Ported BillboardNodes from DX11 to OpenGL.

Samples/Graphics/BillboardNodes/BillboardNodes.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/BillboardNodes/BillboardNodesWindow.cpp

Ported BlendedTerrain from DX11 to OpenGL.

Samples/Graphics/BlendedTerrain/BlendedTerrain.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/BlendedTerrain/BlendedTerrainEffect.cpp
Samples/Graphics/BlendedTerrain/Shaders/BlendedTerrain{Pixel,Vertex}.glsl

Ported TextureArrays from DX11 to OpenGL.

Samples/Graphics/TextureArrays/TextureArrays.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/TextureArrays/TextureArraysWindow.cpp
Samples/Graphics/TextureArrays/Shaders/TextureArrays{Pixel,Vertex}.glsl

Ported Texturing from DX11 to OpenGL.

Samples/Graphics/Texturing/Texturing.{v12,v14}.{sln,vcxproj,vcxproj.filters}

Ported PlaneEstimation from DX11 to OpenGL. The sample apparently works on an AMD Radeon 7970 but it is failing an OpenGL call on an NVIDIA Quadro K2200; it needs further work.

Samples/Mathematics/PlaneEstimation/PlaneEstimation.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/PlaneEstimation/PlaneEstimationWindow.{h,cpp}
Samples/Mathematics/PlaneEstimation/Shaders/EvaluateBezier.glsl

Samples/Mathematics/PlaneEstimation/Shaders/PlaneEstimation.glsl
Samples/Mathematics/PlaneEstimation/Shaders/PlaneVisualize.glsl
Samples/Mathematics/PlaneEstimation/Shaders/PositionVisualize.glsl

Added a new sample TextureUpdating.

Samples/Graphics/TextureUpdating/TextureUpdating.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/TextureUpdating/TextureUpdating.cpp
Samples/Graphics/TextureUpdating/TextureUpdating.{h,cpp}

Added a new sample CubeMap that compiles and runs but is not working correctly; it needs further work.

Data/{Xm,Xp,Ym,Yp,Zm,Zp}Face.png
Samples/Graphics/CubeMaps/CubeMaps.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/CubeMaps/CubeMaps.cpp
Samples/Graphics/CubeMaps/CubeMapsWindow.{h,cpp}
Samples/Graphics/CubeMaps/CubeMapEffect.{h,cpp}
Samples/Graphics/CubeMaps/ReflectTexture.{h,cpp}

The BouncingTetrahedra sample compile and runs, but the LCP solver is not working correctly; it needs further work.

Samples/Physics/BouncingTetrahedra/BouncingTetrahedra.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/BouncingTetrahedra/BouncingTetrahedra.cpp
Samples/Physics/BouncingTetrahedra/BouncingTetrahedra.{h,cpp}
Samples/Physics/BouncingTetrahedra/RigidTetra.{h,cpp}

December 3, 2015. Fixed a comment in the region-4 code. The closest point is V1, not V0.

DistPointTriangleExact.h

Added a new Boolean member mDepthRangeZeroOne to ViewVolume. The value is true when the depth range for the view volume is $[0, 1]$, which is the DirectX convention. The value is false when the depth range is $[-1, 1]$, which is the OpenGL convention. Modified Camera and Light constructors accordingly. Modified engine and sample applications that construct such objects.

ViewVolume.{h,cpp}
Camera.{h,cpp}
Light.{h,cpp}
Window3.cpp
Sample/Graphics/CameraAndLightNodes/CameraAndLightNodesWindow.cpp

Sample/Graphics/Castle/CastleWindow.cpp
Sample/Physics/ExtremalQueryExtremalQueryWindow.cpp

Microsoft Visual Studio 2015 Update 1 introduced a bug where a warning is generated for static class members of type std::vector that are initialized in the source file. The bug has been fixed for Update 2. Added Microsoft-specific code to disable the warning via the pragma system.

Tools/GenerateProject/ProjectTemplate.{v12,v14}.cpp
Sample/Graphics/Castle/CastleWindow.cpp

December 5, 2015. DX11 does not allow texture cubes to be dynamically updated. Modified the usage flag to D3D11_USAGE_DEFAULT.

GteDX11TextureCube.cpp

Fixed a bug in the constructor for Texture. The level-zero offsets were all set to zero for texture arrays, which is incorrect for items with positive index.

GteTexture.cpp

Finished the port of the Wild Magic sample CubeMaps.

GTBuildAll.{v12,v14}.sln
Samples/Graphics/CubeMaps/CubeMaps.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/CubeMaps/CubeMapsWindow.{h,cpp}
Samples/Graphics/CubeMaps/CubeMapEffect.{h,cpp}
Samples/Graphics/CubeMaps/Shaders/CubeMap.hlsl
Samples/Graphics/CubeMaps/ReflectTexture.{h,cpp}

December 6, 2015. Ported the Wild Magic 5 sample GlossMaps to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Graphics/GlossMaps/GlossMaps.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/GlossMaps/GlossMapsWindow.{h,cpp}
Samples/Graphics/GlossMaps/GlossMapEffect.{h,cpp}
Samples/Graphics/GlossMaps/Shaders/GlossMap.hlsl
Samples/Data/Magic.png

December 8, 2015. Ported the Wild Magic 5 sample ProjectedTextures to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Graphics/ProjectedTextures/ProjectedTextures.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/ProjectedTextures/ProjectedTexturesWindow.{h,cpp}
Samples/Graphics/ProjectedTextures/ProjectedTextureEffect.{h,cpp}
Samples/Graphics/ProjectedTextures/Shaders/ProjectedTexture.hlsl
Samples/Data/Magician.png

December 9, 2015. Ported the Wild Magic 5 sample SphereMaps to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Graphics/SphereMaps/SphereMaps.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/SphereMaps/SphereMapsWindow.{h,cpp}
Samples/Graphics/SphereMaps/SphereMapEffect.{h,cpp}
Samples/Graphics/SphereMaps/Shaders/SphereMap.hlsl
Samples/Data/SphereMap.png

Ported the Wild Magic 5 sample VertexTextures to GTEngine.

GTBuildAll.{v12,v14}.sln
Samples/Graphics/VertexTextures/VertexTextures.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/VertexTextures/VertexTexturesWindow.{h,cpp}
Samples/Graphics/VertexTextures/DisplacementEffect.{h,cpp}
Samples/Graphics/VertexTextures/Shaders/Displacement.hlsl
Samples/Data/HeightField.png

December 10, 2015. Added support for cube maps and draw targets in OpenGL. Modified the GLSL reflection. Other code changes are based on adding support for OpenGL features.

GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GTGraphicsGL4.h
GL4Engine.{h,cpp}
GL4DrawTarget.{h,cpp}
GL4TextureRT.{h,cpp}
GL4TextureDS.{h,cpp}
GL4Texture.{h,cpp}
GL4Texture1.{h,cpp}
GL4Texture1Array.{h,cpp}
GL4Texture2.{h,cpp}
GL4Texture2Array.{h,cpp}
GL4Texture3.{h,cpp}
GL4TextureArray.{h,cpp}
GL4TextureCube.{h,cpp}
GL4TextureCubeArray.{h,cpp}

GL4TextureSingle.{h,cpp}
GLSLReflection.{h,cpp}
GLSLProgramFactory.cpp
DataFormat.cpp
OverlayEffect.cpp
Shader.cpp
TextEffect.cpp
Texture2Effect.cpp
Texture3Effect.cpp
VertexColorEffect.cpp
Fluid3AdjustVelocity.cpp
Fluid3ComputeDivergence.cpp
Fluid3EnforceStateBoundary.cpp
Fluid3InitializeSource.cpp
Fluid3InitializeState.cpp
Fluid3SolvePoisson.cpp
Fluid3UpdateState.cpp
Samples/Basics/ShaderReflection/ShaderReflection.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Basics/ShaderReflection/ShaderReflection.cpp
Samples/Basics/ShaderReflection/Billboards{GS,PS,VS}.glsl
Samples/Basics/ShaderReflection/SimpleBuffers.glsl
Samples/Basics/ShaderReflection/TextureArrays{PS,VS}.glsl
Samples/Basics/ShaderReflection/Texturing{PS,VS}.glsl
Samples/Basics/ShaderReflection/VertexColoring{PS,VS}.glsl
Samples/Graphics/BlendedTerrain/Shaders/BlendedTerrain{Pixel,Vertex}.glsl
Samples/Graphics/BlownGlass/BlownGlass.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/BlownGlass/BlownGlassWindows.{h,cpp}
Samples/Graphics/BlownGlass/Shaders/VolumeRender{PS,VS}.glsl
Samples/Graphics/CubeMaps/CubeMaps.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/CubeMaps/CubeMapsWindows.{h,cpp}
Samples/Graphics/CubeMaps/CubeMapsEffect.cpp
Samples/Graphics/CubeMaps/Shaders/CubeMap{PS,VS}.glsl
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargets.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargetsWindows.{h,cpp}
Samples/Graphics/MultipleRenderTargets/Shaders/MultipleRenderTargets{Pixel,Vertex}.glsl
Samples/Graphics/MultipleRenderTargets/Shaders/MultipleRenderTargets.hlsl
Samples/Graphics/TextureArrays/Shaders/TextureArraysVertex.glsl
Samples/Mathematics/PlaneEstimation/Shaders/EvaluateBezier.glsl
Samples/Mathematics/PlaneEstimation/Shaders/PlaneEstimation.glsl
Samples/Physics/Rope/Rope.{v12,v14}.{sln,vcxproj}

Disabled the message-box logger to avoid the unexpected warnings reported by the OpenGL error system.

Samples/Graphics/CubeMaps/CubeMaps.cpp

Added GL4 configurations to the cloth sample.

Samples/Physics/Cloth/Cloth.{v12,v14}.{sln,vcxproj}

December 13, 2015. Added test for simple polygon and test for convex polygon.

Polygon2.h

Fixed a bug in VETManifoldMesh where the base-class vertex creator was called rather than the member-function override.

VETManifoldMesh.cpp

December 14, 2015. Got BlendedAnimations and BumpMaps to work using OpenGL.

GTBuildAll.{v12,v14}.sln
Samples/Graphics/BlendedAnimations/BlendedAnimations.{v12,v14}.{sln,vcxproj}
Samples/Graphics/BlendedAnimations/BlendedAnimationsWindow.{h,cpp}
Samples/Graphics/BlendedAnimations/BipedManager.{h,cpp}
Samples/Graphics/BumpMaps/BumpMaps.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/BumpMaps/SimpleBumpMapEffect.cpp
Samples/Graphics/BumpMaps/Shaders/SimpleBumpMap{PS,VS}.glsl

December 15, 2015. Latest code changes and sample ports. In the engine, replaced an OpenGL 4.4 function call with functions that are supported in OpenGL 4.3. The blend state includes color and sample masks. These must be set regardless of whether blending is enabled. In the draw target, the binding needed to be for depth and stencil together.

GTBuildAll.{v12,v14}.sln
LightingEffect.{h,cpp}
Texture.h
AmbientLightEffect.cpp
ConstantColorEffect.cpp
DirectionalLightEffect.cpp
DirectionalLightTextureEffect.cpp
PointLightEffect.cpp
PointLightTextureEffect.cpp
SpotLightEffect.cpp
GL4BlendState.cpp
GL4DrawTarget.cpp
GL4Engine.cpp
GLSLReflection.cpp
Samples/Basics/ShaderReflection/ShaderReflection.v12.{sln,vcxproj,vcxproj.filters}
Samples/Basics/ShaderReflection/ShaderReflection.v14.vcxproj.filters

Samples/Basics/ShaderReflection/NestedStruct.glsl
Samples/Graphics/BumpMaps/Shaders/SimpleBumpMapPS.glsl
Samples/Graphics/CameraAndLightNodes/CameraAndLightNodes.{v12,v14}.{sln,vcxproj}
Samples/Graphics/CameraAndLightNodes/CameraAndLightNodesWindow.cpp
Samples/Graphics/GeometryShaders/GeometryShaders.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/GeometryShaders/GeometryShadersWindow.cpp
Samples/Graphics/GeometryShaders/Shaders/RandomSquareDirect{GS,VS}.glsl
Samples/Graphics/GeometryShaders/Shaders/RandomSquareIndirect{GS,VS}.glsl
Samples/Graphics/GeometryShaders/Shaders/RandomSquaresPS.glsl
Samples/Graphics/GlossMaps/GlossMaps.{v12,v14}.{sln,vcxproj}
Samples/Graphics/GlossMaps/GlossMapsEffect.cpp
Samples/Graphics/Lights/Lights.{v12,v14}.{sln,vcxproj}
Samples/Graphics/LightTexture/LightTexture.{v12,v14}.{sln,vcxproj}
Samples/Graphics/MultipleRenderTargets/MultipleRenderTargets.v12.vcxproj
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersection.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersectionWindow.cpp
Samples/Graphics/PlaneMeshIntersection/Shaders/DrawIntersections.glsl
Samples/Graphics/PlaneMeshIntersection/Shaders/PlaneMeshIntersection{PS,VS}.glsl
Samples/Graphics/ProjectedTextures/ProjectedTextures.{v12,v14}.{sln,vcxproj}
Samples/Graphics/ProjectedTextures/ProjectedTextureEffect.cpp
Samples/Graphics/SphereMaps/SphereMaps.{v12,v14}.{sln,vcxproj}
Samples/Graphics/SphereMaps/SphereMapEffect.cpp
Samples/Graphics/VertexTextures/VertexTextures.{v12,v14}.{sln,vcxproj}
Samples/Graphics/VertexTextures/DisplacementEffect.cpp
Samples/Physics/BouncingSpheres/BouncingSpheres.{v12,v14}.{sln,vcxproj}

December 18, 2015. Added a filter for shader files and added the Gaussian blurring HLSL file to it.

Samples/Imagics/GaussianBlurring/GaussianBlurring.v14.{vcxproj,vcxproj.filters}

Latest OpenGL changes to support indirect vertex access and structured buffers.

Samples/Graphics/GeometryShaders/GeometryShadersWindows.cpp
Samples/Graphics/GeometryShaders/Shaders/RandomSquaresDirect{GS,VS}.glsl
Samples/Graphics/GeometryShaders/Shaders/RandomSquaresIndirect{GS,VS}.glsl
Samples/Graphics/GeometryShaders/Shaders/RandomSquaresPS.glsl

December 19, 2015. The path to the *.cso files depends on the compiler version.

Samples/Basics/LowLevelD3D11/Applications.cpp

December 20, 2015. Implemented a vertex-collapse algorithm for manifold meshes that preserves the manifold condition for each collapse. Added a sample application to illustrate its use. Fixed the memory leaks (of the vertex data structures) in VETManifoldMesh.

GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GTMathematics.h
VertexCollapseMesh.h
VETManifoldMesh.cpp
Samples/Mathematics/VertexCollapseMesh/VertexCollapseMesh.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/VertexCollapseMesh/VertexCollapseMeshWindow.{h,cpp}
Samples/Mathematics/VertexCollapseMesh/VertexCollapseMesh.cpp

December 22, 2015. Added OpenGL support for structured buffers with atomic counters. Modified some samples to run using OpenGL.

GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
GTGraphicsGL4.h
GL4AtomicCounterBuffer.{h,cpp}
GL4Buffer.{h,cpp}
GL4ConstantBuffer.{h,cpp}
GL4Engine.{h,cpp}
GL4IndexBuffer.cpp
GL4StructuredBuffer.{h,cpp}
GL4VertexBuffer.cpp
GLSLReflection.{h,cpp}
PlanarReflectionEffet.cpp
MemberLayout.h
Shader.{h,cpp}
Samples/Basics/AppendConsumeBuffers/AppendConsumeBuffers.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Basics/AppendConsumeBuffers/AppendConsumeBuffers.cpp
Samples/Basics/AppendConsumeBuffers/Shaders/AppendConsume.{glsl,hlsl}
Samples/Basics/ShaderReflection/ShaderReflection.{v12,v14}.{vcxproj,vcxproj.filters}
Samples/Basics/ShaderReflection/ShaderReflection.cpp
Samples/Basics/ShaderReflection/AppendConsume.{glsl,hlsl}
Samples/Graphics/GeometryShaders/GeometryShadersWindow.{h,cpp}
Samples/Graphics/GeometryShaders/Shaders/RandomSquaresIndirect{GS,VS}.glsl
Samples/Graphics/StructuredBuffers/StructuredBuffers.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/StructuredBuffers/StructuredBuffersWindow.cpp
Samples/Graphics/StructuredBuffers/Shaders/StructuredBuffers{PS,VS}.glsl
Samples/Imagics/SurfaceExtraction/SurfaceExtraction.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.{h,cpp}
Samples/Imagics/SurfaceExtraction/Shaders/DrawSurfaceIndirect{GS,PS,VS}.glsl
Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurface.glsl
Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurfaceIndirect.glsl
Samples/Physics/BouncingBall/BouncingBall.{v12,v14}.{sln,vcxproj}

December 23, 2015. Modified the vertex collapse function `DoCollapse` not to return when the collapse is deferred (the user does not care), instead just updating the min-heap and selecting another vertex to collapse. Removed the high-resolution mesh, keeping only the mesh built after collapses; the caller can store a copy of the high-resolution mesh if needed. Encapsulated the collapse results into a single structure.

VertexCollapseMesh.h
Samples/Mathematics/VertexCollapseMesh/VertexCollapseMeshWindow.{h,cpp}

December 26, 2015. Added an include of GTEngineDEF.h to access the define for GTE_IMPEXP.

GLSLReflection.h

Restructured the header includes and class forward declarations to make the graphics engine headers similar.

DX11Engine.{h,cpp}
GL4Engine.{h,cpp}

January 13, 2016. Added abstract base classes to support polymorphic passing of the graphics engine and other objects to functions (engine is DX11 or GL4). This also supports refactoring the DX11Engine and GL4Engine classes.

$$GraphicsEngine.\{h, cpp\}GEDrawTarget.\{h, cpp\}GEInputLayoutManager.hGEObject.\{h, cpp\}GTGraphics.h \tag{1}$$

Replaced a D3D11 enumeration by an integer constant so that the Window::Parameters initialization is not tied to D3D11 header files. Removed the reference to GTEngine1 in the window class name. Fixed some other parameters.

Window.h
WindowSystem.cpp

Added GetWindowClassName member function.

WindowSystem.h

Changed the listeners for destruction from raw pointers to shared pointers.

DrawTarget.{h,cpp}
GraphicsObjects.{h,cpp}

The z-parameter of the clip position for 2D drawing needs to be -1 for OpenGL, not 0 as for Direct3D.

OverlayEffect.cpp
TextEffect.cpp

Major refactoring to avoid explicit derived-class engine names in resource creation. Moved various members to inline status.

DX11*.{h,cpp}
GL4*.{h,cpp}
GLSL*.{h,cpp}

Added DebugGL4 and ReleaseGL4 configurations to projects.

GTBuildAll.{v12,v14}.sln
GTEngine.{v12,v14}.{vcxproj,vcxproj.filters}
Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2D.{sln,vcxproj}
Samples/Geometrics/ConvexHull2D/ConvexHull2D.{sln,vcxproj}
Samples/Geometrics/ConvexHull3D/ConvexHull3D.{sln,vcxproj}
Samples/Geometrics/Delaunay2D/Delaunay2D.{sln,vcxproj}
Samples/Geometrics/Delaunay3D/Delaunay3D.{sln,vcxproj}
Samples/Geometrics/IntersectBoxCone/IntersectBoxCone.{sln,vcxproj}
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2D.{sln,vcxproj}
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2D.{sln,vcxproj}
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3D.{sln,vcxproj}
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3D.{sln,vcxproj}
Samples/Geometrics/ShortestPath/ShortestPath.{sln,vcxproj}
Samples/Geometrics/TriangulationCDT/TriangulationCDT.{sln,vcxproj}
Samples/Geometrics/TriangulationEC/TriangulationEC.{sln,vcxproj}
Samples/Graphics/BspNodes/BspNodes.{sln,vcxproj}
Samples/Graphics/Castle/Castle.{sln,vcxproj}
Samples/Graphics/Picking/Picking.{sln,vcxproj}
Samples/Graphics/VideoStreams/VideoStreams.{sln,vcxproj}
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitter.{sln,vcxproj}
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitter.{sln,vcxproj}
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVs.{sln,vcxproj}
Samples/Mathematics/Interpolation2D/Interpolation2D.{sln,vcxproj}
Samples/Mathematics/SymmetricEigensolver3x3/SymmetricEigensolver3x3.{sln,vcxproj}
Samples/Physics/BallHill/BallHill.{sln,vcxproj}
Samples/Physics/BallRubberBand/BallRubberBand.{sln,vcxproj}
Samples/Physics/BeadSlide/BeadSlide.{sln,vcxproj}
Samples/Physics/BouncingTetrahedra/BouncingTetrahedra.{sln,vcxproj,vcxproj.filters}
Samples/Physics/DoublePendulum/DoublePendulum.{sln,vcxproj}
Samples/Physics/ExtremalQuery/ExtremalQuery.{sln,vcxproj}
Samples/Physics/IntersectingBoxes/IntersectingBoxes.{sln,vcxproj}
Samples/Physics/IntersectingRectangles/IntersectingRectangles.{sln,vcxproj}
Samples/Physics/KeplerPolarForm/KeplerPolarForm.{sln,vcxproj}

The GL4-based code used a Window-based application to create an OpenGL engine. Rewrote the application to use the new GL4 constructor that hides the backing window for the device, making the code similar to that for DX11.

Samples/Basics/AppendConsumeBuffers/AppendConsumeBuffers.cpp

The engine Execute function that took a ComputeShader input has been removed, so the application code needed to be updated accordingly to use the Execute function that takes a ComputeProgram input.

GenerateMeshUV.h
Fluid2AdjustVelocity.cpp
Fluid2ComputeDivergence.cpp
Fluid2EnforceStateBoundary.cpp
Fluid2InitializeSource.cpp
Fluid2InitializeState.cpp
Fluid2SolvePoisson.cpp
Fluid2UpdateState.cpp
Samples/Basics/IEEEFloatingPoint/IEEEFloatingPoint.cpp
Samples/Basics/RawBuffers/RawBuffers.cpp
Samples/Geometrics/AllPairsTriangles/AllPairsTrianglesWindow.cpp
Samples/Geometrics/DistanceSegments3/DistanceSegments3.cpp
Samples/Geometrics/ShortestPath/ShortestPathWindow.cpp
Samples/Geometrics/ShortestPath/GpuShortestPath.cpp
Samples/Imagics/Convolution/ConvolutionWindow.cpp
Samples/Imagics/GaussianBlurring/GaussianBlurringWindow.cpp
Samples/Imagics/MedianFiltering/MedianFilteringWindow.{h,cpp}
Samples/Mathematics/PartialSums/PartialSums.cpp
Samples/Mathematics/RootFinding/RootFinding.cpp
Samples/Physics/MassSprings3D/GpuMassSpringVolume.cpp

The text Draw function now takes a color that is std::array<float,4> rather than Vector4<float>.

Samples/Geometrics/ConvexHull3D/ConvexHull3D.cpp
Samples/Graphics/BlendedAnimations/BlendedAnimationsWindow.cpp
Samples/Graphics/Castle/CastleWindow.cpp
Samples/Graphics/Lights/LightsWindow.cpp
Samples/Graphics/LightTexture/LightTextureWindow.cpp
Samples/Imagics/Convolution/ConvolutionWindow.cpp
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp
Samples/Imagics/VideoStreams/VideoStreamsWindow.cpp
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitterWindow.cpp
Samples/Physics/BouncingBall/BouncingBallWindow.cpp
Samples/Physics/BouncingSpheres/BouncingSpheresWindow.cpp

Replaced the DX11Engine* parameter by the base class GraphicsEngine.

Samples/Geometrics/ShortestPath/GpuShortestPath.{h,cpp}
Samples/Imagics/VideoStreams/FileVideoStream.{h,cpp}
Samples/Imagics/VideoStreams/VideoStream.{h,cpp}

The application was missing a line of code that indicates the 2D screen needs to be updated in video memory.

Samples/Geometrics/TriangulationCDT/TriangulationCDTWindow.cpp

Removed the conditional compilation on engine type to use the generic base class GraphicsEngine.

PlanarReflectionEffect.{h,cpp}

January 17, 2016. Implemented the GLSL shaders for the sample application. Modified some application code to handle the problem with GLSL wanting to pad vec3 arrays in the shaders as if they had vec4 elements.

GTBuildAll.{v12,v14}.sln
Samples/Physics/MassSprings3D/MassSprings3D.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/MassSprings3D/Shaders/RungeKutta.hlsl
Samples/Physics/MassSprings3D/MassSprings3DWindow.cpp
Samples/Physics/MassSprings3D/GpuMasSpringVolume.{h,cpp}
Samples/Physics/MassSprings3D/Shaders/DrawUsingVertexID{VS,PS}.glsl
Samples/Physics/MassSprings3D/Shaders/RungeKutta*.glsl

Implemented the GLSL shaders for the sample application. Modified the application code as needed.

Fluid2.{h,cpp}
Fluid2AdjustVelocity.{h,cpp}
Fluid2ComputeDivergence.{h,cpp}
Fluid2EnforceStateBoundary.{h,cpp}
Fluid2InitializeSource.{h,cpp}
Fluid2InitializeState.{h,cpp}
Fluid2SolvePoisson.{h,cpp}
Fluid2UpdateState.{h,cpp}
GTBuildAll.{v12,v14}.sln
Samples/Physics/Fluids2D/Fluids2D.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/Fluids2D/Fluids2DWindow.cpp
Samples/Physics/Fluids2D/Shaders/DrawDensity.glsl

Implemented the GLSL shaders for the sample application. Modified the application code as needed.

Fluid3.{h,cpp}
Fluid3AdjustVelocity.{h,cpp}

Fluid3ComputeDivergence.{h,cpp}
Fluid3EnforceStateBoundary.{h,cpp}
Fluid3InitializeSource.{h,cpp}
Fluid3InitializeState.{h,cpp}
Fluid3SolvePoisson.{h,cpp}
Fluid3UpdateState.{h,cpp}
GTBuildAll.{v12,v14}.sln
Samples/Physics/Fluids3D/Fluids3D.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Physics/Fluids3D/Fluids3DWindow.cpp
Samples/Physics/Fluids3D/Shaders/VolumeRender{VS,PS}.glsl

Implemented the GLSL shaders for the sample application. Modified the application code as needed.

GTBuildAll.{v12,v14}.sln
Samples/Graphics/WireMesh/WireMesh.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/WireMesh/WireMeshWindow.cpp
Samples/Graphics/WireMesh/Shaders/WireMesh{VS,PS,GS}.glsl

Replaced the Window-based approach to get a GL4 engine by using the GL4Engine constructor that hides the backing window.

Samples/Basics/ShaderReflection/ShaderReflection.cpp

January 18, 2016. The GL4 texture classes copied CPU data to the GPU on initial creation only when the texture object was not marked as SHADER_OUTPUT; however, this prevents a texture from being read-write in the shaders. Modified the code to copy CPU data to the GPU when the CPU data exists, just as in the DX11 engine.

GL4TextureSingle.cpp
GL4TextureArray.cpp
GL4TextureCubeArray.cpp

Implemented the GLSL shaders for the sample application. Modified the application code as needed. GLSL will not expand an expression in a conditional define that turns out to be a known constant. Commented out the defines both in the HLSL and GLSL code and required the application to set the defines directly.

GTBuildAll.{v12,v14}.sln
Samples/Mathematics/PartialSums/PartialSums.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/PartialSums/PartialSums.cpp
Samples/Mathematics/PartialSums/Shaders/PartialSums.hlsl
Samples/Mathematics/PartialSums/Shaders/PartialSums.glsl

Implemented the GLSL shaders for the sample application. Modified the application code as needed. Fixed an out-of-date comment in the HLSL file.

GTBuildAll.{v12,v14}.sln
Samples/Mathematics/RootFinding/RootFinding.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Mathematics/RootFinding/RootFinding.cpp
Samples/Mathematics/RootFinding/Shaders/RootFinding.hlsl
Samples/Mathematics/RootFinding/Shaders/RootFinding.glsl

Implemented the GLSL shaders for the sample application. Modified the application code as needed. Changed the shader resource names from `input` and `output` to `inImage` and `outImage`, because GLSL has the keyword `output`.

GTBuildAll.{v12,v14}.sln
Samples/Imagics/GaussianBlurring/GaussianBlurring.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/GaussianBlurring/GaussianBlurringWindow.cpp
Samples/Imagics/GaussianBlurring/Shaders/GaussianBlur3x3.hlsl
Samples/Imagics/GaussianBlurring/Shaders/GaussianBlur3x3.glsl

Implemented the GLSL shaders for the sample application. Modified the application code as needed. Changed the shader resource names from `input` and `output` to `inImage` and `outImage`, because GLSL has the keyword `output`.

GTBuildAll.{v12,v14}.sln
Samples/Imagics/MedianFiltering/MedianFiltering.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/MedianFiltering/MedianFilteringWindow.cpp
Samples/Imagics/MedianFiltering/Shaders/Median3x3.hlsl
Samples/Imagics/MedianFiltering/Shaders/Median5x5.hlsl
Samples/Imagics/MedianFiltering/Shaders/MedianBySort.hlsl
Samples/Imagics/MedianFiltering/Shaders/Median3x3.glsl
Samples/Imagics/MedianFiltering/Shaders/Median5x5.glsl
Samples/Imagics/MedianFiltering/Shaders/MedianBySort.glsl

January 19, 2016. Added layout information for GLSL.

VertexColorEffect.cpp

Implemented the GLSL shaders for the sample application. Modified the application code as needed. Modified the shaders so that the structured buffers use 4-tuple vectors rather than 3-tuple vectors.

GTBuildAll.{v12,v14}.sln
Samples/Geometrics/AllPairsTriangles/AllPairsTriangles.{v12,v14}.{sln,vcxproj,vcxproj.filters}

Samples/Geometrics/AllPairsTriangles/AllPairsTrianglesWindow.cpp
Samples/Geometrics/AllPairsTriangles/Shaders/DrawUsingVertexID.hlsl
Samples/Geometrics/AllPairsTriangles/Shaders/TriangleIntersection.hlsl
Samples/Geometrics/AllPairsTriangles/Shaders/DrawUsingVertexID{VS,}.glsl
Samples/Geometrics/AllPairsTriangles/Shaders/InitializeColors.glsl
Samples/Geometrics/AllPairsTriangles/Shaders/TriangleIntersection.glsl
Samples/Geometrics/AllPairsTriangles/Shaders/VertexColorIndexed{VS,}.glsl

January 21, 2016. Replaced calls to glBufferStorage by glBufferData based on the requirement that one need only OpenGL 4.3 to compile and run the code.

GL4Buffer.cpp
GL4StructuredBuffer.cpp

Implemented the GLSL shaders for the sample application. Modified the application code as needed.

GTBuildAll.{v12,v14}.sln
Samples/Geometrics/DistanceSegments3/DistanceSegments3.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Geometrics/DistanceSegments3/DistanceSegments3.cpp
Samples/Geometrics/DistanceSegments3/Shaders/DistanceSeg3Seg3.{hlsl,glsl}
Samples/Geometrics/DistanceSegments3/DistanceSeg3Seg3.hlsl

Implemented the GLSL shaders for the sample application. Modified the application code as needed.

GTBuildAll.{v12,v14}.sln
Samples/Graphics/ShortestPath/ShortestPath.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Graphics/ShortestPath/ShortestPathWindow.cpp
Samples/Graphics/ShortestPath/GpuShortestPathWindow.{h,cpp}
Samples/Graphics/ShortestPath/CpuShortestPathWindow.cpp
Samples/Graphics/ShortestPath/Shaders/PartialSumsDiagToCol.hlsl
Samples/Graphics/ShortestPath/Shaders/PartialSumsDiagToRow.hlsl
Samples/Graphics/ShortestPath/Shaders/WeightsShader.hlsl
Samples/Graphics/ShortestPath/Shaders/InitializeDiagToCol.glsl
Samples/Graphics/ShortestPath/Shaders/InitializeDiagToRow.glsl
Samples/Graphics/ShortestPath/Shaders/PartialSumsDiagToCol.glsl
Samples/Graphics/ShortestPath/Shaders/PartialSumsDiagToRow.glsl
Samples/Graphics/ShortestPath/Shaders/UpdateShader.glsl
Samples/Graphics/ShortestPath/Shaders/WeightsShader.glsl

Implemented the GLSL shaders for the sample application. Modified the application code as needed.

GTBuildAll.{v12,v14}.sln
Samples/Basics/IEEEFloatingPoint/IEEEFloatingPoint.{v12,v14}.{sln,vcxproj,vcxproj.filters}

Samples/Basics/IEEEFloatingPoint/IEEEFloatingPoint.cpp
Samples/Basics/IEEEFloatingPoint/Shaders/TestSubnormals.hlsl
Samples/Basics/IEEEFloatingPoint/Shaders/TestSubnormals.glsl

Moved the bodies of functions of ComputeModel from the cpp file to the header. Instead of activating GPGPU via the Win32 define, a new define has been added called GTE_COMPUTE_MODEL_ALLOW_GPGPU. This is currently not exposed so that the mathematics system does not include the graphics system. You can either expose the define by editing the file or you can add the define to preprocessor options in the projects. The GenerateMeshUVs sample application uses the latter choice.

GTBuildAll.{v12,v14}.sln
ComputeModel.h
ComputeModel.cpp
GenerateMeshUV.{h,cpp}
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVs.{v12,v14}.vcxproj
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.cpp
Samples/Mathematics/GenerateMeshUVs/Shaders/TestSubnormals.hlsl
Samples/Mathematics/GenerateMeshUVs/Shaders/TestSubnormals.glsl

Implemented the GLSL shaders for the sample application. Modified the application code as needed.

GTBuildAll.{v12,v14}.sln
Samples/Imagics/Convolution/Convolution.{v12,v14}.{sln,vcxproj,vcxproj.filters}
Samples/Imagics/Convolution/ConvolutionWindow.cpp
Samples/Imagics/Convolution/Shaders/Convolve.hlsl
Samples/Imagics/Convolution/Shaders/ConvolveGS.hlsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableH.hlsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableHGS.hlsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableHGS2.hlsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableV.hlsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableVGS.hlsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableVGS.hlsl
Samples/Imagics/Convolution/Shaders/Convolve.glsl
Samples/Imagics/Convolution/Shaders/ConvolveGS.glsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableH.glsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableHGS.glsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableHGS2.glsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableV.glsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableVGS.glsl
Samples/Imagics/Convolution/Shaders/ConvolveSeparableVGS.glsl

January 23, 2016. Removed the projects RawBuffers and SharedTextures from the build-all solutions. Renamed them to RawBuffersD3D11 and SharedTexturesD3D11 because they use features specific to D3D11. The build-all solutions now have projects, each having configurations for D3D11 (Debug, Release) and OpenGL4.3 (DebugGL4, ReleaseGL4).

74 Updates to Version 1.14

June 7, 2015. Added MSVS 2015 solutions and projects for GTEngine library.

GTEngine.v14.sln
GTEngine.v14.vcxproj
GTEngine.v14.vcxproj.filters

June 27, 2015. Added MSVS 2015 solutions and projects for all samples.

June 28, 2015. Fixed a typographical error in the comments before SqrDistanceSpecial.

DistPointHyperellipsoid.h

Fixed a bug that was introduced into the GTEngine code after it was ported from Wild Magic 5. The order of parameters in the DoQuery calls was incorrect.

IntrRay3AlignedBox3.h

Fixed compiler warnings in MSVS 2015 about name in one scope visible to the same name in another scope.

DistCircle3Circle3.h
Integration.h
IntrEllipse2Ellipse2.h
IntrEllipsoid3Ellipsoid3.h
SingularValueDecomposition.h

Minor change. Modified variable name to be consistent with other DX11 resource files.

DX11Buffer.{h,cpp}

July 12, 2015. The level offsets for a TextureArray without mipmaps were all zero. They need to be set to multiples of the number of bytes of each texture in the array.

Texture.cpp

July 13, 2015. The Matrix constructor with initializer list input was missing an increment of the row counter when the number of initializers fills a row only partially.

Matrix.inl

July 26, 2015. Removed the thread ID check in DX11Engine. This was a Geometric Tools Engine constraint, forcing the programmer to use an engine object in the same thread because the device context is not thread safe. However, you may use the engine object in multiple threads as long as you provide the critical section to avoid concurrent access. Rewrote the creation and destruction for the swap chain and back buffer to ensure that the application shuts down gracefully. In the previous code, if the device creation failed, the immediate context was null and then dereferenced during the destruction phase. The new code was tested using fault injection to produce a failure at each attempt to query for an interface or to create a resource.

DX11Engine.{h,cpp}

July 29, 2015. The template alias for 2D distance from point to box was defined incorrectly (currently no engine code is using the alias).

DistPointOrientedBox.h

Added support for saving Texture2 objects to JPEG.

WICFileIO.{h,cpp}

August 1, 2015. The Boolean value of interval-interval find-intersection was not assigned to the Boolean value of the result.

IntrSegment2AlignedBox2.h
IntrRay2AlignedBox2.h

August 2, 2015. Enclosed the class in namespace gte.

HLSLParameter.h

August 5, 2015. DX11Texture1 needed to derive from DX11TextureSingle rather than DX11Texture.

DX11Texture1.{h,cpp}

August 9, 2015. The y-coordinate of the Platonic texture coordinates was not computed (a WM5 porting error).

MeshFactory.cpp

August 11, 2015. The ESC key was handled in the window procedure outside of OnCharPress and is mapped to exiting the application with a PostQuitMessage(0) call. This made it impossible to override. The handling has been moved to Window::OnCharPress to allow applications to override the behavior.

Window.{h,cpp}
WindowSystem.cpp

The generation of project files was not creating exactly the character pattern to allow the Microsoft Visual Studio Selector to recognize which version of Visual Studio to launch when a solution is selected. The SLN files are UT8 and require 0xEF, 0xBB, 0xBF, 0x0D, 0x0A as the starting characters.

ProjectTemplate.{v12,v14}.{h,cpp}

August 12, 2015. Eliminated unnecessary c_str() call.

HLSLShaderFactory.cpp

The function DX11Engine::Share needed to create a regular 2D texture, a render-target texture, or a depth-stencil texture based on the type information in the GraphicsObject class.

DX11Engine.cpp

August 15, 2015. The subscribe and unsubscribe procedures for listeners to participate in updates of pvw-matrices and w-matrices was more complicated than it should be. Removed hiding the complexity. The w-matrix processing is handled now by explicitly manipulating the mTrackballNode object. The pvw-matrix subscribe and unsubscribe functions have appropriate names and require you to pass in explicitly the pair of world transform and constant buffer. Sample applications were modified accordingly.

Window.{h,cpp}

August 16, 2015. Fixed a porting error for creating a torus mesh. The texture coordinates were incorrect at the wrap-around.

MeshFactory.cpp

August 21, 2015. Made several functions protected to prevent applications from calling them. The Spatial and Visual classes must call these from public scope, so they were made friends of Culler.

Culler.h

August 23, 2015. Factored the trackball code out of Window and into a class Trackball. Applications that require a virtual trackball must create an object of the new class and implement MouseClick and MouseMotion callbacks to manage the trackball. Many of our sample applications have the new system, so see those for examples of how to do this. The modified sample files are not listed here.

Trackball.{h,cpp}
Window.{h,cpp}

The LogReporter objects in the main functions of the sample applications are now exposed only in debug configurations. The Window::Parameters structure was given a new constructor for the common case of specifying the title-bar string, the window origin, and the window dimensions. Such windows are not resizable. If you need additional parameters set, you can do so after the constructor call—or you can set all members explicitly after a default-constructor call. The modified sample files are not listed here.

Window.h

Moved Window::GetPickLine to Camera and removed the explicit dependency on the DX11Engine by requiring that the viewport be passed to GetPickLine.

Window.{h,cpp}
Camera.{h,cpp}
Samples/Graphics/Picking/PickingWindow.cpp

August 24 to September 23, 2015. Major revision of the lighting effects, including adding classes Light and Material. Added a class CameraRig that encapsulates camera motion and updating the dependent pvw-matrices. Added a class Window3, derived from Window, that encapsulates the common code for 3D sample applications. Ported Wild Magic 5 graphics samples: BillboardNodes, BlendedAnimations, BspNodes, BumpMaps, CameraAndLightNodes, and Castle. Added scene-graph classes as needed to support these. Tested the distribution when the convention GTE_USE_VEC_MAT is selected (default convention is GTE_USE_MAT_VEC and fixed any problems that arose.

75 Updates to Version 1.13

June 3, 2015. Changes to get code to compile using MSVS 2015 RC. All errors were actually warnings about reusing variable names in a nested scope.

HLSLShaderFactory.cpp
ConvexHull2.h
Delaunay2.h
Delaunay3.h
ETManifoldMesh.cpp
TriangulateEDT.h

TriangulateEC.h
TubeSurface.h
Samples/Basics/LowLevelD3D11/Application.cpp

June 7, 2015. The intersection queries were missing the public modifier. Enabled the internal unit tests for these classes (all tests succeed).

IntrPlane3Capsule3.h
IntrPlane3Cylinder3.h
IntrPlane3Ellipsoid3.h
IntrPlane3OrientedBox3.h
IntrPlane3Sphere3.h

Added -fPIC to the CC flags for dynamic library builds. This avoids a compiler-linker error in Fedora 21.

makefile.gte

76 Updates to Version 1.12

April 23, 2015. Major rewrite of the document on converting between coordinate systems. The original version of this document was entitled *Conversion of Left-Handed Coordinates to Right-Handed Coordinates* and was written to handle the conversion of LightWave coordinate systems (left-handed) to Wild Magic coordinate systems (right-handed). The process was specific to LightWave's choice of representing rotations using Euler angles, and the discussion included how to deal with cameras, lights, and transformation hierarchies. The new version is the general process of converting between any two coordinate systems. An implementation is provided that automates the process.

ConvertingBetweenCoordinateSystems.pdf (*Converting Between Coordinate Systems*)
ConvertCoordinates.h
LeftHandedToRightHanded.pdf (*Conversion of Left-Handed Coordinates to Right-Handed Coordinates*)

April 29, 2015. Converted some accessor functions to inline.

DX11GraphicsObject.{h,cpp}

Changed LogWarning to LogError when the preparation for copy fails.

DX11Resource.cpp

April 30, 2015. Removed the Set* members from the class. These are never called in any GTEngine applications, nor should they. Once you create a visual effect, it should remain immutable (so to speak).

VisualEffect.h

May 3, 2015. Removed the unused template classes derived from StructuredBuffer.

GTEngine.{h,vcxproj,vcxproj.filters}
StructuredBuffer{1,2,3}.h

May 10, 2015. Removed errant inline modifiers on functions that are not likely to be inlined.

MinimumAreaBox2.h

May 17, 2015. Modified the algorithm based on the analysis and description in the modified PDF for the minimum-area rectangle containing points. For 256-point convex polygons with random vertices, the new code executes nearly twice as fast as the old code.

MinimumAreaBox2.h
MinimumAreaRectangle.pdf (*Minimum-Area Rectangle Containing a Set of Points*)

May 18, 2015. Modified the implementation of MinimumVolumeBox3 to be similar to the new changes in MinimumAreaBox2. There is some performance improvement (about 10% faster). Added the ability to specify that the ProcessEdges function should be run in a thread concurrently when ProcessFaces is executing. The speed-up is noticeable (about 20% for a test data set of about 6000 points).

MinimumVolumeBox3.h

Removed mVisited as a class member and moved it to local scope in a function. This had to be done in MinimumVolumeBox3 to avoid concurrent access to mVisited when multithreading is used. Made the same change to MinimumAreaBox2 in case we decide to add threading support.

MinimumAreaBox2.h

May 19, 2015. As mentioned, if ComputeType is not an exact arithmetic type, the Delaunay triangulation can fail because of incorrect sign classification due to numerical rounding errors. The class still had several places where failures of this type were not trapped and handled gracefully. Checks were added in the appropriate locations so that Update now reports a Boolean value that is true when successful. The failure is plumbed all the way back to the application call of operator(...).

Delaunay2.h

May 22, 2015. Ported Polynomial1 from Wild Magic 5 to GTEngine.

GTEngine.{h,vcxproj,vcxproj.filters}
Polynomial1.h

May 28, 2015. Modified the point-circle algorithm to return a Boolean flag equidistant rather than number of closest circle points. Modified the comments. Added unit tests.

DistPoint3Circle3.h

May 29, 2015. Ported the line-circle distance algorithm from Wild Magic 5, but modified the bisection approach to be easier to read. Added the polynomial-based approach that finds roots of at most a degree-4 polynomial. Added unit tests internally.

GTEngine.{h,vcxproj,vcxproj.filters}
DistLine3Circle3.h

May 31, 2015. Added GetMaxBisections to the Functions wrapper that hides the details of determining the maximum number of iterations for using bisection with a floating-point number type. This is useful in RootsBisection::Find.

Functions.h
DistPointHyperellipsoid.h

Added a function GetOrthogonal to generate one vector perpendicular to the input vectors. The returned vector can be selected to be normalized (or not). The functions ComputeOrthogonalComplement compute a set of basis vectors for the orthogonal complement of an input, but in some algorithms not all are needed.

Vector.h

Added a document that combines three previous documents regarding distance queries involving circles in 3D. The new document provides much greater detail about circle-circle queries than the brief note in the original version. Source code was updated based on the new document and unit tests were added internally.

DistanceToCircle3.pdf (*Distance to Circles in 3D*)
DistPoint3Circle3.h
DistLine3Circle3.h
DistCircle3Circle3.h
DistancePoint3Circle3.pdf (*Distance Between Point and Circle or Disk in 3D*)
DistanceLine3Circle3.pdf (*Distance Between Line and Circle or Disk in 3D*)
DistanceCircle3Circle3.pdf (*Distance Between Two Circles in 3D*)
DistanceCircle3Disk3.pdf (*Distance Between a Circle and a Disk in 3D*)

Apple LLVM 6.1.0 complained yet once again that "std::abs is ambiguous", this time in the minimum-area and minimum-volume box code when running our tool to trap compiler problems with template instantiation. The relevant verification code is

```
#include <GteMinimumAreaBox2.h>
// Has line of code
//   ComputeType cmax = std::max(std::abs(axis[0]), std::abs(axis[1]));

#include <GteBSRational.h>
// Has lines of code
//   namespace std
//   {
//       template <typename UIntegerType> inline
//       gte::BSRational<UIntegerType> abs(gte::BSRational<UIntegerType> const& number)
//       {
//           return (number.GetSign() >= 0 ? number : -number);
//       }
//   }

#include <GteUIntegerAP32.h>

namespace gte
{
    // ComputeType for this instantiation is BSRational<UIntegerAP32>.
    template class MinimumAreaBox2<float, BSRational<UIntegerAP32>>;
}
```

The compiler complains that std::abs is ambiguous in the line of code that computes cmax. However, it has no complaints about the code

```
#include <GteBSRational.h>
#include <GteUIntegerAP32.h>
#include <GteMinimumAreaBox2.h>
namespace gte
{
    template class MinimumAreaBox2<float, BSRational<UIntegerAP32>>;
}
```

In the listing with GteMinimumAreaBox2.h included first, the implied behavior is the following. LLVM attempts to instantiate class MinimumAreaBox2 with ComputeType equal to BSRational<UIntegerAP32>. For LLVM to complain about the std::abs in the line of code for cmax, it has not yet processed the definition of std::abs in the file GteBSRational.h even though it should have read the contents of this file before encountering the explicit instantiation code. Using the template code for the Standard C++ Library, it tries to find a match for std::abs with type BSRational<UIntegerAP32> but fails. In the listing with GteMinimumAreaBox2.h included last, the implied behavior is that LLVM processes the definition of std::abs in the file GteBSRational.h before attempting to instantiate class MinimumAreaBox2. During the instantiation, it finds a match for std::abs for the type BSRational<UIntegerAP32>.

Is this an Apple LLVM bug? Or is the header file order dependency required because of some C++ 11 specification? Removing the const reference in the std::abs input for BSRational<UIntegerAP32> did not eliminate the problem. The Xcode symbol navigator actually shows the various definitions of abs in <cmath>, <cstdlib>, stdlib.h, GteBSNumber.h, and GteBSRational.h. For the latter two files, the symbol navigator shows abs<>(). For the other occurrences, the symbol navigator shows abs(). Perhaps the Standard C++ Library just has too many occurrences of abs wrapped with layers of "Is this C code? C++ 0X code or C++ 11 code? Inside or outside namespace std?"

77 Updates to Version 1.11

April 11, 2015. The Apple LLVM 6.1.0 compiler generated an error, stating that declarations of the specializations for oppositeFace must occur before the definitions (in the cpp file). Added the declarations in the h file, but in a conditional compilation block to prevent exposing them to Microsoft Visual Studio 2013. The Microsoft compiler incorrectly generates an error about a redefinition of oppositeFace. A comment in the code states what the C++ specification says about explicit instantiation of static members of a template class. This is the main change that led to shipping a new version of GTEngine so quickly after the previous version.

TetrahedronKey.h

April 11, 2015. Added functions GetRootInfo* to count the number of real-valued roots and their multiplicities without actually computing the roots.

RootsPolynomial.h

April 20, 2015. Two blocks of code were returning true rather than an integer representing number of bisections used in the code, a carry-over from when the functions used to return Boolean values. Modified the functions to return meaningful integers in all cases.

RootsBisection.h

Added significant improvements to (1) the document for the test-intersection and find-intersection queries for ellipses and (2) the document for computing the area of intersection of ellipses. The source code implementation for the test-intersection and find-intersection queries were updated to use the details described in the document. Implemented the query for area of intersection of ellipses. Unit tests for the queries were added internally. The area queries were also computed numerically using Mathematica 10.1 to verify that the GTEngine code is producing correct results.

IntrEllipse2Ellipse2.h
IntersectionOfEllipses.pdf (*Intersection of Ellipses*)
AreaIntersectingEllipses.pdf (*The Area of Intersecting Ellipses*)

78 Updates to Version 1.10

March 11, 2015. Added new distance queries for points and cylinders.

GTEngine.{h,vcxproj,vcxproj.filters}
DistPoint3Cylinder3.{h,cpp}

March 13, 2015. Added tag-dispatch-based type traits class Arithmetic. Added a class Function that provides math wrapper functions with implementations based on the tag types. The wrappers are for float, double, and long double (floating-point types); for IEEEBinary16 (16-bit floating-point type); and for BSNumber and BSRational (exact types, but the functions return approximations because the results cannot be represented exactly). Removed the previous support for square roots from the distance query header. Removed the min and max implementations from the exact types, because std::min and std::max work as-is due to the existence of comparison operators in the binary scientific classes. Modified the segment-segment distance query to use the new system.

GTEngine.{h,vcxproj,vcxproj.filters}
Constants.h
Functions.h
IEEEBinary16.{h,cpp}
DCPQuery.h
DistSegmentSegment.h
BSNumber.h
BSRational.h
Arithmetic.h

March 29, 2015. Added some definitions for variables used in the equations in the section *Separation Tests Involving Other Directions*. Converted the verbatim pseudocode to use lstlisting format.

IntersectionOfCylinders.pdf (*Intersection of Cylinders*)

April 2, 2015. Moved the debug members mValue to be the first members of the classes. This allows one to see the converted values in the MSVS debugger variable summary; thus, you do not have to open tree controls to drill down to the converted value. Added adjustments to BSRational to enforce a constraint that when the numerator is 0, the denominator should be 1. Fixed a bug in the conversion from BSRational to a floating-point type. When the round-up step causes a carry-out, so to speak, from the trailing significand, a block of code was executed to set w to 1 and adjust $p - q$. This was incorrect and, in fact, not necessary because w is not used as the trailing significand in the conversion. The documentation was updated accordingly.

BSNumber.h
BSRational.h
ArbitraryPrecision.pdf (*GTEngine: Arbitrary Precision Arithmetic*)

April 4, 2015. Rewrote the document for computing the roots of low-degree polynomials. The new document goes into great detail about the classification of roots (real or non-real, multiplicities) and how to use exact rational arithmetic to correctly classify the roots in a program. This leads to more robust root finding using the closed-form expressions for polynomials of degrees 2, 3, and 4. Implementations of this replace the old code in RootsPolynomial. Unit tests were added (in-house) that provide 100 % code coverage for the robust root finders. The motivation for the revisions was based on trying to compute intersections of ellipses, and the nonrobust root finder for quartic polynomials created many problems numerically.

LowDegreePolynomialRoots.pdf (*Low-Degree Polynomial Roots*)
RootsPolynomial.h

79 Updates to Version 1.9

February 10, 2015. Added Distance functions that are controlled by the *tag-dispatch pattern* in order to provide a single interface to the square root function for numerical types (float, double) and for exact types (BSNumber, BSRational).

DCPQuery.h
DistSegmentSegment.h

February 11, 2015. In the else clause at the end of operator(), the 2×2 matrix needs to be a rotation, not a reflection.

SymmetricEigensolver2x2.h

February 19, 2015. A new tool to compile an HLSL shader and store it as bytecode embedded in a C++ array. This is useful if you do not want to expose an algorithm by an embedded human-readable string in the source code and executable.

GTBuildAll.sln
HLSLShaderFactory.{h,cpp}
ShaderFactory.{h,cpp}
Tools/HLSLToByteCode/HLSLToByteCode.sln
Tools/HLSLToByteCode/HLSLToByteCode.vcxproj
Tools/HLSLToByteCode/HLSLToByteCode.vcxproj.filters
Tools/HLSLToByteCode/HLSLToByteCode.cpp

February 23, 2015. The general symmetric eigensolver had an incorrect algorithm for tracking the number of reflections in order to determine whether the product of Householder reflections, Givens rotations, and sorting of eigenvalues leads to a rotation or a reflection. The other changes involving replacing the general eigensolver with solvers specific to 2D and 3D.

SymmetricEigensolver.h
ApprGaussian2.h
ApprGaussian3.h
ApprGreatCircle.h
ApprOrthogonalLine2.h
ApprOrthogonalLine3.h
ApprOrthogonalPlane3.h
Hyperellipsoid.h

IntrEllipse2Ellipse2.h
IntrEllipsoid3Ellipsoid3.h

February 24, 2015. MinimumVolumeBox3 no longer uses the code in MinimumAreaBox2, so the latter code is now private to the class. MinimumVolumeBox3 now uses exact rational arithmetic for all phases, avoiding the floating-point rounding problems of the old code that projected the extreme polyline to the supporting plane of the hull face.

MinimumAreaBox2.h
MinimumVolumeBox3.h

The box vertex order changed for OrientedBox2 and OrientedBox3, so the drawing code needed modification.

Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.cpp
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.cpp

Made TransformBy a const function.

BoundingSphere.{h,cpp}

February 26, 2015. Replaced the $O(n^2)$ algorithm for computing the minimum-area rectangle derived from a hull-face normal and its corresponding extreme polyline. The replacement uses the $O(n)$ Rotating Calipers algorithm extended to 3D boxes that are extrusions of a 2D rectangle.

MinimumVolumeBox3.h

February 28, 2015. Added CPU multithreading support to ConvexHull3 and MinimumVolumeBox3. Improved the performance of the comparison operators for BSRational by early-out testing of the signs of numerators. Extended BSPrecision to measure precision for BSRational expressions. Added comments to header files about the choice of N for UIntegerFP<N>.

ConvexHull3.h
MinimumVolumeBox3.h
BSRational.h
BSPrecision.{h,cpp}
ConvexHull2.h
PrimalQuery2.h
PrimalQuery3.h

March 3, 2015. The conversion from the rational minimum-volume box to floating-point box had a loss of precision too early in the conversion. This led to divisions by zero in the extents and to zero-valued axes. Some preconditioning was added to defer the precision loss until the very last step.

MinimumVolumeBox3.h

Added support for disk input/output.

BSNumber.h
BSRational.h
UIntegerAP32.{h,cpp}
UIntegerFP32.h

March 4, 2015. Fixed a subtle error in the minimum-volume bounding box code when rotating calipers is used. The starting rectangle needed axes that satisfied a particular scaling constraint in order to properly compare scaled areas. The 2D code was rewritten to be similar in structure to the 3D code.

MinimumVolumeBox3.h
MinimumAreaBox2.h
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.{h,cpp}

Fixed the coefficients in `b[]` in the ChebyshevRatio file. Modified the slerp code in the SlerpEstimate file to compile on Playstation 4 clang compiler.

ChebyshevRatio.h
SlerpEstimate.h

March 10, 2015. Based on data sets provided in bug reports, we decided to do a *deep dive* into the rotating calipers algorithm and code. The convex polygon must be ordered counterclockwise. The loop of the algorithm has two invariants:

1. the edges emanating from the current supporting points of the box must not be coincident with a box edge, and

2. of all consecutive polygon points that support a box edge, the chosen support point must be right-most on the edge (as the box is traversed counterclockwise).

To satisfy the first invariant, collinear vertices must be removed from the convex polygon; otherwise, an box edge might contain three or more support points. To satisfy the second invariant, the new bottom support point for a minimum-angle polygon edge is trivial to select and the new support points are a subset of the old support points. However, two (or more) emanating edges that tie for the minimum angle with box edges can cause a new support point *not* to be right-most on an edge. We revised the minimum-area and minimum-volume box code to ensure the invariants are satisfied. The 3D problem is somewhat more complicated because projecting a polyline that supports a box with face coincident with a convex hull face is not possible when using exact rational arithmetic.

MinimumVolumeBox3.h
MinimumAreaBox2.h
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.{h,cpp}
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.{h,cpp}

Added a conditional define so that one may view double-precision values of BSNumber and BSRational when debugging code that uses exact rational arithmetic. Be aware that this can be very slow because of the conversion to double-precision whenever new objects are created by arithmetic operations.

BSNumber.h
BSRational.h

Increased the integer template parameter of UIntegerFP32 in the sample algorithm so that it supports any float input.

Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow.cpp

Some traps are placed in the edge-insertion code to avoid crashing when the constrained triangulation failed due to use of floating-point ComputeType. Two additional tests were needed to verify that the item iterator was valid before dereferencing it.

ConstrainedDelaunay2.h

80 Updates to Version 1.8

January 14, 2015. Added DX11.1 formats to the DF_* enumeration.

DataFormat.{h,cpp}

January 22, 2015. Fixed bugs in the line-rectangle distance query. The last block of code used result.lsParameter[0] when it should have used result.lsParameter[1]. The indexing for result.rectangleParameter[1] was incorrect.

DistLine3Rectangle3.h

January 22, 2015. Added a template wrapper for the square root function in order to overload float, double, BSNumber, and BSRational implementations. The wrapper is named Sqrt and will replace any calls to sqrt where the programmer has the option to use an exact arithmetic type instead of a standard floating-point type.

Functions.h
BSNumber.h
BSRational.h

January 22, 2015. Converted the point-triangle distance query to a template that has the dimension as a parameter.

DistPoint3Tetrahedron3.h
DistRay3Triangle3.h
DistSegment3Tetrahedron3.h
DistPointTriangle.h
DistPoint3Triangle3.h

January 25, 2015. Added a function to count the number of active graphics objects and the system memory they are using.

DX11Engine.{h,cpp}

January 26, 2015. Fixed an error in the description for region 2 and added some clarification about the sign tests for the partial derivatives of $R(s,t)$.

DistanceLine3Line3.pdf (*Robust Computation of Distance Between Line Segments*)

Implemented a robust version of point-triangle distance that avoids the problems when a nearly degenerate triangle causes the numerical determinant to be zero. The previous version is now written for exact arithmetic, because it uses fewer arithmetic operations than the robust version.

DistPointTriangleExact.h
DistPointTriangle.h

January 31, 2015. Fixed an error in the logic for handling the directional derivative of the quadratic function $R(s,t)$ along the segment of intersection where $\partial R(s,t)/\partial s = 0$.

DistSegmentSegment.h
Samples/Mathematics/Geometrics/DistanceSegments3/DistanceSeg3Segs.hlsl

Fixed an error in the computation of the t-parameter.

DistPointSegment.h

Added an include of <cstdlib> to resolve an Xcode compiler complaint about ambiguous use of std::abs.

IntrAlignedBox3Cone3.h

81 Updates to Version 1.7

December 14, 2014. Moved the INL file content into the H files and deleted all the INL files. The number of INL files was on the order of over 400, so they are not listed here.

December 16, 2014. Replaced 3D-specific Rectangle3 with n-dimensional Rectangle.

GTEngine.{h,vcxproj,vcxproj.filters}
DistPoint3Rectangle3.h
DistLine3Rectangle3.h
Rectangle.h
Rectangle3.h

Converted distance and intersection code to use new Line, Ray, and Segment objects.

DistLineLine.h
DistLineRay.h
DistLineSegment.h
DistRayRay.h
DistRaySegment.h
DistLine3Triangle3.h
DistIntrLine3Capsule3.h
DistIntrRay3Capsule3.h
Picker.cpp
DistLine{2,3}Line{2,3}.h
DistLine{2,3}Ray{2,3}.h
DistLine{2,3}Segment{2,3}.h
DistRay{2,3}Ray{2,3}.h
DistRay{2,3}Segment{2,3}.h

Added convenient template aliases.

DistPointAlignedBox.h
DistPointHyperellipsoid.h
DistPointLine.h
DistPointOrientedBox.h
DistPointRay.h
DistPointSegment.h
DistSegmentSegment.h

Renamed the source file to match the header file name.

> GenerateMeshUV.cpp
> GenerateMeshUVs.cpp

December 17, 2014. Replaced 3D-specific Cone3 with n-dimensional Cone.

> GTEngine.{h,vcxproj,vcxproj.filters}
> IntrLine3Cone3.h
> IntrSphere3Cone3.h
> Cone.h
> Cone3.h

Replaced 3D-specific Capsule3 with n-dimensional Capsule.

> GTEngine.{h,vcxproj,vcxproj.filters}
> ContCapsule3.h
> IntrCapsule3Capsule3.h
> IntrHalfspace3Capsule3.h
> IntrLine3Capsule3.h
> IntrPlane3Capsule3.h
> Capsule.h
> Capsule3.h

Added intersection testing for 2D oriented box and cone.

> GTEngine.{h,vcxproj,vcxproj.filters}
> IntrOrientedBox2Cone2.h

December 17, 2014. The constructor initialization for mTextColor in the project generation needed to have braces based on the recent change to support std::initialize.

> Tools/GenerateProject/ProjectTemplate.cpp

December 19, 2014. The GetVertices function needed to initialize the mask to 1, not to 0.

> OrientedBox.h

December 26, 2014. Added intersection testing for 3D oriented box and cone, including a sample to illustrate visually that the tests work correctly.

GTEngine.{h,vcxproj,vcxproj.filters}
GTBuildAll.sln
IntrOrientedBox3Cone3.h
Samples/Geometrics/IntersectBoxCone/IntersectBoxConeWindow.{h,cpp}
Samples/Geometrics/IntersectBoxCone/IntersectBoxCone.{sln,vcxproj,vcxproj.filters,cpp}

Moved the inline function bodies to the header files and removed the inline files from the distribution (to reduce file count).

Tools/GenerateApproximations/GteAMDPerformance.{vcxproj,vcxproj.filters}
Tools/GPUPerfAPI-2.1.739.0/GteAMDPerformance.h
Tools/GenerateApproximations/FitASin.h
Tools/GenerateApproximations/FitATan.h
Tools/GenerateApproximations/FitCos.h
Tools/GenerateApproximations/FitExp2.h
Tools/GenerateApproximations/FitASin.h
Tools/GenerateApproximations/FitInvSqrt.h
Tools/GenerateApproximations/FitLog2.h
Tools/GenerateApproximations/FitReciprocal.h
Tools/GenerateApproximations/FitSin.h
Tools/GenerateApproximations/FitTan.h
Tools/GPUPerfAPI-2.1.739.0/GteAMDPerformance.inl
Tools/GenerateApproximations/FitASin.inl
Tools/GenerateApproximations/FitATan.inl
Tools/GenerateApproximations/FitCos.inl
Tools/GenerateApproximations/FitExp2.inl
Tools/GenerateApproximations/FitASin.inl
Tools/GenerateApproximations/FitInvSqrt.inl
Tools/GenerateApproximations/FitLog2.inl
Tools/GenerateApproximations/FitReciprocal.inl
Tools/GenerateApproximations/FitSin.inl
Tools/GenerateApproximations/FitTan.inl

Fixed a build break in the AMD performance sample. The sample needed updating because of several recent changes in GTEngine.

Samples/Basics/PerformanceAMD/PerformanceAMDWindow.{h,cpp}
Samples/Basics/PerformanceAMD/PerformanceAMD.{vcxproj,vcxproj.filters}

December 28, 2014. Added template aliases TIIntervalInterval and FIIntervalInterval.

IntrIntervals.h

Modified the document on line-cone intersection and updated the code comments to match it.

IntersectionLineCone.pdf (*Intersection of a Line and a Cone*)
IntrLine3Cone3.h
IntrRay3Cone3.h
IntrSegment3Cone3.h

January 4, 2015. Added test-intersection query for 3D aligned boxes and cones. Modified the test-intersection query for 3D oriented boxes to derive from the new query. Added template aliases TIAligned-Box3Cone3 and TIOrientedBox3Cone3 for convenience. Modified the sample application for testing the code.

IntrOrientedBox3Cone3.h
Samples/Geometrics/IntersectBoxCone/IntersectBoxConeWindow.{h,cpp}
IntrAlignedBox3Cone3.h
IntersectionBoxCone.pdf (*Intersection of a Box and a Cone*)

82 Updates to Version 1.6

November 27, 2014. The pseudocode for computing the fitted cylinder subtracted the input point average for numerically stable computations. The returned center needed the average added to it.

CylinderFitting.pdf (*Fitting 3D Data with a Cylinder*)

November 28, 2014. The B-spline curve and surface fitting had problems when the matrix storage convention is GTE_USE_COL_MAJOR. The banded matrix system solver and inverter selected storage convention based on the macro setting. Instead, the convention must be selected by the user based on his knowledge of the storage used by the input matrices. The relevant functions in BandedMatrix are now templates with a Boolean parameter that specifies the storage convention. The fitter classes have always used row-major order, regardless of the GTEngine macro that is active.

BandedMatrix.{h,inl}
BSplineCurveFit.inl
BSplineSurfaceFit.inl

December 1, 2014. Added estimates for the SLERP of quaternions. Factored out the estimations into the class ChebyshevRatio so that they can be used for vectors of other dimensions. The error bounds for the Chebyshev ratios are now reported based on numerical experiments; the paper on which they were based had a flawed error analysis. Revised the Quaternion class to use the general formula for SLERP rather than the estimate it had been using. Added more SLERP functions that have restricted domains and allow for preprocessing of quaternions (as animation data). In particular the EstimateRPH is useful for preprocessed quaternions that allow the estimates to be applied for angles no larger than $\pi/4$, leading to very small errors.

GTEngine.{h,vcxproj,vcxproj.filters}
Quaternion.{h,inl}

ChebyshevRatio.{h,inl}
SlerpEstimate.{h,inl}

December 2, 2014. The Vector2, Vector3, and Vector4 classes derived from Vector existed solely to provide constructors for 2, 3, or 4 inputs, respectively. We added two constructors to Vector, one having an input std::array<Real> and one that has the input std::initialize, allowing an arbitrary number of inputs. We have removed the derived classes, but you can still use the class names because we have used template aliasing to define them properly. For example, the alias for 2D vectors is

```
template <typename Real>
using Vector2 = Vector<2,Real>;
```

We eliminated special static members and used instead Vector specials. The comparison operators now directly use the lexicographically based ones for std::array. Quite a large number of files have changed, both engine and application code.

Vector.{h,inl}
Vector2.{h,inl}
Vector3.{h,inl}
Vector4.{h,inl}
ApprEllipse2.inl
ApprEllipsoid3.inl
ApprGaussian{2,3}.inl
ApprOrthogonalLine{2,3}.inl
Arc2.inl
BoundingSphere.{inl,cpp}
Circle3.inl
Cone3.inl
ContEllipse2.inl
ContEllipsoid3.inl
ContOrientedBox{2,3}.inl
ContPointInPolyhedron3.inl
ContScribeCircle2.inl
ContScribeCircle3Sphere3.inl
Delanuay{2,3}.inl
Delaunay{2,3}Mesh.inl
DistPoint3Frustum3.inl
Ellipse2.inl
Ellipsoid3.inl
Frustum3.inl
GenerateMeshUV.inl
Halfspace3.inl
IntpSphere2.inl
IntrCircle2Circle2.inl
IntrLine{2,3}OrientedBox{2,3}.inl
IntrLine3Capsule3.inl
IntrLine3Cylinder3.inl
IntrRay{2,3}OrientedBox{2,3}.inl

IntrSegment{2,3}OrientedBox{2,3}.inl
Line{2,3}.inl
MinimumAreaBox2.inl
MinimumAreaCircle2.inl
MinimumVolumeBox3.inl
MinimumVolumeSphere3.inl
OrientedBox{2,3}.inl
PlanarMesh.{h,inl}
Plane3.inl
Polygon2.inl
PolyhedralMassProperties.inl
Polyhedron3.inl
Ray{2,3}.inl
Segment{2,3}.inl
Tetrahedron3.inl
Torus3.inl
Transform.inl
Triangle{2,3}.inl
TriangulateEC.inl
BlendState.cpp
Camera.cpp
CullingPlane.cpp
Fluid{2,3}.cpp
Fluid{2,3}InitializeSource.cpp
LightingConstants.cpp
MeshFactory.cpp
Node.cpp
OverlayEffect.cpp
Picker.cpp
SamplerState.cpp
TextEffect.cpp
Visual.cpp
Window.cpp
Samples/Basics/DirectionalLightTexture/DirectionalLightTextureWindow.cpp
Samples/Basics/GaussianBlurring/GaussianBlurringWindow.cpp
Samples/Basics/GeometryShaders/GeometryShadersWindow.cpp
Samples/Basics/MultipleRenderTargets/MultipleRenderTargetsWindow.cpp
Samples/Basics/SharedTextures/SharedTexturesWindow.cpp
Samples/Basics/StructuredBuffers/StructuredBuffersWindow.cpp
Samples/Basics/TextureArrays/TextureArraysWindow.cpp
Samples/Basics/Texturing/TexturingWindow.cpp
Samples/Basics/VertexColoring/VertexColoringWindow.cpp
Samples/Geometrics/AllPairsTriangles/AllPairsTrianglesWindow.cpp
Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2DWindow.cpp
Samples/Geometrics/ConvexHull2D/ConvexHull2DWindow.cpp
Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow.cpp
Samples/Geometrics/Delaunay2D/Delaunay2DWindow.cpp
Samples/Geometrics/Delaunay3D/Delaunay3DWindow.cpp

Samples/Geometrics/DistanceSegments3/DistanceSegments3.cpp
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.cpp
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2DWindow.cpp
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.cpp
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3DWindow.cpp
Samples/Geometrics/ShortestPath/ShortestPathWindow.cpp
Samples/Geometrics/TriangulationCDT/TriangulationCDTWindow.cpp
Samples/Geometrics/TriangulationEC/TriangulationECWindow.cpp
Samples/Graphics/BlendedTerrain/BlendedTerrainEffect.cpp
Samples/Graphics/BlendedTerrain/BlendedTerrainWindow.cpp
Samples/Graphics/BlownGlass/BlownGlassWindow.cpp
Samples/Graphics/Lights/LightsWindow.cpp
Samples/Graphics/Picking/PickingWindow.cpp
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersectionWindow.cpp
Samples/Graphics/VideoStreams/VideoStreamsWindow.cpp
Samples/Graphics/WireMesh/WireMeshWindow.cpp
Samples/Imagics/Convolution/ConvolutionWindow.cpp
Samples/Imagics/MedianFiltering/MedianFilteringWindow.cpp
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitterWindow.cpp
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitterWindow.cpp
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.cpp
Samples/Mathematics/Interpolation2D/Interpolation2DWindow.cpp
Samples/Mathematics/PlaneEstimation/PlaneEstimationWindow.cpp
Samples/Physics/Cloth/ClothWindow.cpp
Samples/Physics/Fluids2D/Fluids2DWindow.cpp
Samples/Physics/Fluids3D/Fluids3DWindow.cpp
Samples/Physics/MassSprings3D/MassSprings3DWindow.cpp
Samples/Physics/Rope/RopeWindow.cpp

Modified files to have uniformity in using `#if defined(symbol)` rather than `#ifdef symbol`.

GMatrix.inl
GVector.inl
GaussianElimination.inl
IntelSSE.inl
LinearSystem.inl
Logger.h
MinimizeN.inl
SingularValueDecomposition.inl
SymmetricEigensolver.inl
DX11DrawTarget.cpp
DX11Engine.cpp
DX11InputLayoutManager.cpp
HLSLDefiner.cpp
HLSLShaderFactory.cpp
Samples/Basics/DirectionalLightTexture/DirectionalLightTextureEffect.cpp

Samples/Basics/GaussianBlurring/GaussianBlurringWindow.{h,cpp}
Samples/Basics/GeometryShaders/GeometryShadersWindow.{h,cpp}
Samples/Basics/LowLevelD3D11/Application.cpp
Samples/Basics/Texturing/TexturingWindow.{h,cpp}
Samples/Basics/VertexColoring/VertexColoringWindow.{h,cpp}
Samples/Geometrics/AllPairsTriangles/AllPairsTrianglesWindow.{h,cpp}
Samples/Geometrics/ShortestPath/ShortestPathWindow.{h,cpp}
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.{h,cpp}
Samples/Physics/MassSprings3D/MassSprings3DWindow.{h,cpp}

The file had a template parameter hard-coded as float when it should have been Real.

IntrSphere3Frustum3.inl

December 3, 2014. The Matrix2x2, Matrix3x3, and Matrix4x4 classes derived from Matrix existed solely to provide constructors for 2×2, 3×3, and 4×4 matrices. We added two constructors to Matrix, one having an input std::array<Real, NumRows*NumCols> and one that has the input std::initialize, allowing an arbitrary number of inputs. We have removed the derived classes, but you can still use the class names because we have added template aliasing to define them properly. For example, the alias for 2×2 matrices is

```
template <typename Real>
using Matrix2x2 = Matrix<2,2,Real>;
```

We also eliminated special static members and used instead Matrix specials. The comparison operators now use the lexicographically based ones for std::array.

Matrix.{h,inl}
Matrix2x2.{h,inl}
Matrix3x3.{h,inl}
Matrix4x4.{h,inl}
GMatrix.{h,inl}
ApprEllipse2.inl
ApprEllipsoid3.inl
ContOrientedBox3.inl
ContScribeCircle2.inl
ContScribeCircle3Sphere3.inl
IntrEllipse2Ellipse2.inl
IntrEllipsoid3Ellipsoid3.inl
OdeImplicitEuler.inl
Rotation.inl
Camera.cpp
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp

Fixed a comment about the static member required by the class when you expose a conditional define.

UIntegerFP32.inl

The UpdateConstants function needed conditional compilation to handle either matrix-vector multiplication convention.

DirectionalLightTextureWindow.cpp

The general vector and matrix classes were modified to have the same interfaces as their template-based counterparts.

GMatrix.{h,inl}
GVector.{h,inl}
Matrix.{h,inl}
Vector.{h,inl}

December 4, 2014. Removed the macro GTE_MAKE_HLSL_STRING. This macro was handy, allowing you to embed HLSL code in a CPP file, view it, and modify it as if it were regular C++ code. The macro builds a string that is sent to the HLSL compiler, but before doing so we used <regex> to substitute matches in regular expressions involving preprocess macros such as #if and #ifdef. Unfortunately, the macro does not work on Linux or Macintosh machines as a general string building mechanism using gcc compilers. Our built-in effects such as Texture2Effect all had preprocessor defines involving #ifdef GTE_USE_MAT_VEC to control whether the shader should use the matrix-vector or vector-matrix multiplication convention. The C++ syntax highlighting made it appear as if the correct code blocks were exposed. The Microsoft Visual Studio 2013 compiler builds the string first, so in fact the code blocks that appeared to be active were actually not. The HLSLDefiner class has a mechanism for setting macro names in HLSL files before compilation; for example, it sets the string "GTE_USE_MAT_VEC" to the string "1" or "0" according to the convention active in the $C++$ code. It turns out this is incorrect, because you would instead need #if GTE_USE_MAT_VEC in the HLSL files. As soon as you switch to using the vector-matrix convention in C++ code, the HLSL files are incorrectly compiled and some sample applications no longer work. We also sometimes had to rewrite some HLSL lines of code because the macro was confused by the content of that code that interfered with the preprocessor's interpretation of the macro. In the end, this turned out to be a bad idea. After removal of the macro, the embedded shader code was replaced with quoted strings. Other shaders in HLSL files had to be modified to produce the correct code for vector-matrix convention. The samples have now all been tested using either matrix-vector or vector-matrix multiplication convention and using either row-major or column-major storage of the matrices (4 possible configurations). We discovered one bug in the Transform class, where the translation of the underlying 4×4 matrix needed to be built differently depending on the multiplication convention. It showed up only when you use the GTEngine scene graph support and some node in the graph has a local transform with a nonzero translation.

ShaderFactory.{h,cpp}
ContanstColorEffect.cpp
Fluid{2,3}AdjustVelocity.cpp
Fluid{2,3}ComputeDivergence.cpp
Fluid{2,3}EnforceStateBoundary.cpp

Fluid{2,3}InitializeSource.cpp
Fluid{2,3}InitializeState.cpp
Fluid{2,3}SolvePoisson.cpp
Fluid{2,3}UpdateState.cpp
GenerateMeshUVs.cpp
LightAmbientEffect.cpp
LightDirectionPerPixelEffect.cpp
LightDirectionPerVertexEffect.cpp
LightPointPerPixelEffect.cpp
LightPointPerVertexEffect.cpp
LightSpotPerPixelEffect.cpp
LightSpotPerVertexEffect.cpp
OverlayEffect.cpp
TextEffect.cpp
Texture{2,3}Effect.cpp
VertexColorEffect.cpp
Samples/Basics/DirectionalLightTexture/DirectionalLightTextureEffect.cpp
Samples/Basics/GeometryShaders/Shaders/RandomSquares.hlsl
Samples/Basics/GeometryShaders/Shaders/RandomSquaresIndirect.hlsl
Samples/Basics/MultipleRenderTargets/MultipleRenderTargetsWindow.cpp
Samples/Basics/MultipleRenderTargets/Shaders/MultipleRenderTargets.hlsl
Samples/Basics/ShaderReflection/TextureArrays.hlsl
Samples/Basics/ShaderReflection/Texturing.hlsl
Samples/Basics/ShaderReflection/VertexColoring.hlsl
Samples/Basics/StructuredBuffers/Shaders/StructuredBuffers.hlsl
Samples/Basics/TextureArrays/Shaders/TextureArrays.hlsl
Samples/Geometrics/AllPairsTriangles/Shaders/DrawUsingVertexID.hlsl
Samples/Geometrics/AllPairsTriangles/Shaders/TriangleIntersection.hlsl
Samples/Geometrics/AllPairsTriangles/Shaders/VertexColorIndexed.hlsl
Samples/Graphics/BlownGlass/Shaders/VolumeRender.hlsl
Samples/Graphics/PlaneMeshIntersection/Shaders/PlaneMeshIntersection.hlsl
Samples/Graphics/WireMesh/Shaders/WireMesh.hlsl
Samples/Imagics/SurfaceExtraction/Shaders/DrawSurfaceIndirect.hlsl
Samples/Physics/Fluids3D/Shaders/VolumeRender.hlsl
Samples/Physics/MassSprings3D/Shaders/DrawUsingVertexID.hlsl

As mentioned, we replaced the macroized embedded HLSL code in built-in effects with quoted strings. A new folder was added, `GeometricTools/GTEngine/Shaders` that contain HLSL files with this code (no quoted strings). These are just for reference and are not loaded from disk during engine/application run time. Other HLSL files used in specialized applications are loaded from disk, but the HLSL files are in the application project directory trees.

Shaders/ConstantColorEffect.hlsl
Shaders/DirectionalLightTextureEffect.hlsl
Shaders/Fluid{2,3}AdjustVelocity.hlsl
Shaders/Fluid{2,3}ComputeDivergence.hlsl
Shaders/Fluid{2,3}EnforcePoissonBoundary.hlsl

Shaders/Fluid{2,3}EnforceStateBoundary.hlsl
Shaders/Fluid{2,3}GenerateVortex.hlsl
Shaders/Fluid{2,3}InitializeSource.hlsl
Shaders/Fluid{2,3}InitializeState.hlsl
Shaders/Fluid{2,3}SolvePoisson.hlsl
Shaders/Fluid{2,3}UpdateState.hlsl
Shaders/Fluid{2,3}ZeroPoisson.hlsl
Shaders/GenerateMeshUVs.hlsl
Shaders/LightAmbientEffect.hlsl
Shaders/LightDirectionPerPixelEffect.hlsl
Shaders/LightDirectionPerVertexEffect.hlsl
Shaders/LightPointPerPixelEffect.hlsl
Shaders/LightPointPerVertexEffect.hlsl
Shaders/LightSpotPerPixelEffect.hlsl
Shaders/LightSpotPerVertexEffect.hlsl
Shaders/MultipleRenderTargetsPShader{0,1,2,3,4}.hlsl
Shaders/OverlayEffectColorPShader.hlsl
Shaders/OverlayEffectGrayPShader.hlsl
Shaders/OverlayEffectVShader.hlsl
Shaders/TextEffect.hlsl
Shaders/Texture{2,3}Effect.hlsl
Shaders/VertexColorEffect.hlsl

We discovered a bug in the Transform class, where the translation of the underlying 4×4 matrix needed to be built differently depending on the multiplication convention. It showed up only when you use the GTEngine scene graph support and some node in the graph has a local transform with a nonzero translation.

Transform.{h,inl}

We had added the capability to select (GTE_USE_MAT_VEC or GTE_USE_VEC_MAT) and (GTE_USE_ROW_MAJOR or GTE_USE_COL_MAJOR). This capability has been removed because it has the potential to cause mismatches in CPP code. The comments in the file had indicated this. The conclusion is that if you want conventions different from our defaults, you will have to modify `GTEngineDEF.h` to select yours, and you will have to modify this file every time you grab a newer distribution.

GTEngineDEF.h

The points were not rotating with the trackball (the segments and sphere were). Subscribed the points up to the automatic update mechanism of class Window.

Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3D.h
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3D.cpp

Converted an #ifdef to #if defined().

Samples/Graphics/VideoStreams/VideoStreamsWindow.cpp

December 7, 2014. Factored out the calls to memcpy and memcpy_s to a wrapper function Memcpy to encapsulate the platform differences in a single location.

GTEngine.{h,vcxproj,vcxproj.filters}
GaussianElimination.{h,inl}
GenerateMeshUV.{h,inl}
LinearSystem.{h,inl}
MinimizeN.{h,inl}
SingularValueDecomposition.{h,inl}
SymmetricEigensolver.{h,inl}
DX11Buffer.cpp
DX11Texture.cpp
HLSLShader.cpp
HLSLShaderVariable.cpp
Font.cpp
MarchingCubes.cpp
Image.cpp
Samples/Basics/GeometryShaders", "GeometryShadersWindow.cpp
Samples/Basics/MultipleRenderTargets", "MultipleRenderTargetsWindow.cpp
Samples/Basics/TextureArrays", "TextureArraysWindow.cpp
Samples/Geometrics/ConvexHull3D", "ConvexHull3DWindow.cpp
Samples/Geometrics/MinimumVolumeBox3D", "MinimumVolumeBox3DWindow.cpp
Samples/Imagics/MedianFiltering", "MedianFilteringWindow.cpp
Wrapper.{h,cpp}

The function body for Contain was in the header file. Moved it to the inline file to be consistent with style.

PlanarMesh.h

A new document describing a variation of an iterative eigensolver for symmetric 3×3 matrices. A source code implementation and a sample application are provided.

GTEngine.{h,vcxproj,vcxproj.filters}
GTBuildAll.sln
SymmetricEigensolver3x3.{h,inl}
Samples/Mathematics/SymmetricEigensolver3x3/SymmetricEigensolver3x3.{sln,vcxproj}
Samples/Mathematics/SymmetricEigensolver3x3/SymmetricEigensolver3x3.{vcxproj.filters,cpp}
RobustEigenSymmetric3x3.pdf (*A Robust Eigensolver for 3x3 Symmetric Matrices*)

December 8, 2014. Started the process of consolidating and sharing code for various geometric primitives by using templates for which the dimension is one of the parameters. This will occur in stages. The list of new files for the first stage is shown next.

GTEngine.{h,vcxproj,vcxproj.filters}
AlignedBox.{h,inl}
DistPointAlignedBox.{h,inl}
DistPointLine.{h,inl}
DistPointOrientedBox.{h,inl}
DistPointRay.{h,inl}
DistPointSegment.{h,inl}
DistSegmentSegment.{h,inl}
Hyperellipsoid.{h,inl}
Hypersphere.{h,inl}
Line.{h,inl}
OrientedBox.{h,inl}
Ray.{h,inl}
Segment.{h,inl}
ApprEllipse2.{h,inl}
ApprEllipsoid3.{h,inl}
ApprGaussian{2,3}.h
ApprQuadratic{2,3}.h
ApprOrthogonalLine{2,3}.h
Capsule3.h
ContCapsule3.h
ContCircle2.h
ContCylinder3.h
ContEllipse2.h
ContEllipsoid3.h
ContOrientedBox{2,3}.inl
ContOrientedBox3.inl
ContScribeCircle3Sphere3.h
ContScribeSphere3.h
ConvexHull{2,3}.h
Cylinder3.h
Delaunay{2,3}.h
DistLine{2,3}Line{2,3}.h
DistLine{2,3}Ray{2,3}.h
DistLine{2,3}Segment{2,3}.h
DistLine3AlignedBox3.h
DistPoint2Ellipse2.h
DistPoint3Ellipsoid3.h
DistRay{2,3}Ray{2,3}.h
DistRay{2,3}Segment{2,3}.h
DistRay3AlignedBox3.{h,inl}
DistRay3OrientedBox3.{h,inl}
DistRay3Rectangle3.h
DistRay3Triangle3.h
DistSegment3AlignedBox3.{h,inl}
DistSegment3OrientedBox3.{h,inl}
DistSegment3Rectangle3.h
DistSegment3Triangle3.h

IntpQuadraticNonuniform2.{h,inl}
IntrCircle2Circle2.h
IntrAlignedBox{2,3}AlignedBox{2,3}.h
IntrAlignedBox{2,3}OrientedBox{2,3}.{h,inl}
IntrCapsule3Capsule3.h
IntrEllipse2Ellipse2.{h,inl}
IntrHalfspace3Ellipsoid3.h
IntrHalfspace3OrientedBox3.h
IntrHalfspace3Segment3.h
IntrHalfspace3Sphere3.h
IntrLine2AlignedBox2.h
IntrLine2Circle2.h
IntrLine2Line2.h
IntrLine2OrientedBox2.h
IntrLine2Ray2.h
IntrLine2Segment2.h
IntrLine2Triangle2.h
IntrLine3AlignedBox3.h
IntrLine3Cone3.h
IntrLine3Cylinder3.h
IntrLine3Ellipsoid3.h
IntrLine3OrientedBox3.h
IntrLine3Plane3.h
IntrLine3Sphere3.h
IntrLine3Triangle3.h
IntrOrientedBox2Circle2.h
IntrOrientedBox{2,3}OrientedBox{2,3}.{h,inl}
IntrOrientedBox3Frustum3.{h,inl}
IntrOrientedBox3Sphere3.h
IntrPlane3Cylinder3.h
IntrPlane3Ellipsoid3.h
IntrPlane3OrientedBox3.h
IntrPlane3Plane3.h
IntrRay2AlignedBox2.h
IntrRay2Circle2.h
IntrRay2OrientedBox2.h
IntrRay2Ray2.h
IntrRay2Segment2.h
IntrRay2Triangle2.h
IntrRay3AlignedBox3.h
IntrRay3Cone3.h
IntrRay3Cylinder3.h
IntrRay3Ellipsoid3.h
IntrRay3OrientedBox3.h
IntrRay3Plane3.h
IntrRay3Sphere3.h
IntrRay3Triangle3.h
IntrSegment{2,3}AlignedBox{2,3}.h

IntrSegment2Circle2.h
IntrSegment{2,3}OrientedBox{2,3}.h
IntrSegment2Segment2.h
IntrSegment2Triangle2.h
IntrSegment3Capsule3.h
IntrSegment3Cone3.h
IntrSegment3Cylinder3.h
IntrSegment3Ellipsoid3.h
IntrSegment3Plane3.h
IntrSegment3Sphere3.h
IntrSegment3Triangle3.h
IntrSphere3Cone3.h
MinimumAreaBox2.h
MinimumAreaCircle2.h
MinimumVolumeBox3.{h,inl}
MinimumVolumeSphere3.h
Picker.{h,cpp}
Projection.h
Samples/Geometrics/DistanceSegments3/DistanceSegments3.cpp
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.cpp
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.cpp
AlignedBox{2,3}.{h,inl}
Circle2.{h,inl}
Ellipse2.{h,inl}
Ellipsoid3.{h,inl}
Point{2,3}AlignedBox{2,3}.{h,inl}
Point{2,3}Line{2,3}.{h,inl}
Point{2,3}OrientedBox{2,3}.{h,inl}
Point{2,3}Ray{2,3}.{h,inl}
Point{2,3}Segment{2,3}.{h,inl}
DistSegment{2,3}Segment{2,3}.{h,inl}
Line{2,3}.{h,inl}
OrientedBox{2,3}.{h,inl}
Ray{2,3}.{h,inl}
Segment{2,3}.{h,inl}
Sphere3.{h,inl}

Implemented separate code for segment-segment intersection using exact rational arithmetic.

DistSegmentSegmentExact.{h,inl}
Samples/Geometrics/DistanceSegments3/DistanceSegments3.cpp
DistanceSegmentsRobust.{h,inl}

Implemented the special case symmetric eigensolver for 2×2 matrices.

SymmetricEigensolver2x2.{h,inl}

December 9, 2014. Another stage of consolidating code to support objects of multiple dimension, classes Hyperplane, Halfspace, and Triangle.

GTEngine.{h,vcxproj,vcxproj.filters}
Halfspace.{h,inl}
Hyperplane.{h,inl}
Triangle.{h,inl}
ConvexHull3.h
Delaunay3.h
DistLine3Triangle3.h
DistPoint3Triangle3.h
DistPoint3Plane3.h
IntrHalfspace3Capsule3.h
IntrHalfspace3Cylinder3.h
IntrHalfspace3Ellipsoid3.h
IntrHalfspace3OrientedBox3.h
IntrHalfspace3Segment3.h
IntrHalfspace3Sphere3.h
IntrHalfspace3Triangle3.h
IntrLine{2,3}Triangle{2,3}.h
IntrPlane3Plane3.h
IntrPlane3Triangle3.h
IntrRay3Triangle3.h
IntrSegment3Triangle3.h
SeparatePoints3.inl
Tetrahedron.h
Halfspace3.{h,inl}
Plane3.{h,inl}
Triangle2.{h,inl}

Removed the class GeometricPrimitive. We thought it would store more information per object type, but given it is a fancy base class for comparison operators, it is not necessary.

GTEngine.{h,vcxproj,vcxproj.filters}
Arc2.{h,inl}
Capsule3.{h,inl}
Circle3.{h,inl}
Cone3.{h,inl}
Cylinder3.{h,inl}
Ellipse3.{h,inl}
Frustum3.{h,inl}
Polygon2.h
Polyhedron3.h
Rectangle3.{h,inl}

Tetrahedron3.{h,inl}
Torus3.{h,inl}
GeometricPrimitive.{h,inl}

In SymmetricEigensolver, the test for a sort request in the function to get a single eigenvector was incorrect. Added functions to SingularValueDecomposition to get one singular value or one column of an orthogonal matrix. This is useful when you do not need an entire matrix.

SymmetricEigensolver.inl
SingularValueDecomposition.{h,inl}

December 12, 2014. Replaced the code for distance queries point-ellipse and point-ellipsoid with point-hyperellipsoid, where the dimension of the hyperellipsoid is now a template parameter. Added unit tests for the code to compute the point-ellipse and point-ellipsoid distance. The tests provide 100 percent code coverage. Updated the PDF that describes the algorithm to reflect some modifications made in the code.

DistancePointEllipseEllipsoid.pdf (*Distance from a Point to an Ellipse, an Ellipsoid, or a Hyperellipsoid*)
GTEngine.{h,vcxproj,vcxproj.filters}
ApprEllipse2.h
ApprEllipsoid3.h
DistPointHyperellipsoid.{h,inl}
DistPoint2Ellipse2.{h,inl}
DistPoint3Ellipsoid3.{h,inl}

Fixed a porting bug in IntrLine2Line2 that had the cases swapped between parallel-and-equal and parallel-and-not-equal lines.

IntrLine2Line2.inl

Removed unused variables (LLVM on Macintosh complained).

ContCapsule3.inl

Return type of SetMaxSizeToZero needed to be void (LLVM on Macintosh complained).

UIntegerAP32.h
UIntegerFP32.h

LLVM on Macintosh complained about the SolveSystem<Real> function call, requiring the template keyword to modify the function call. The fitter classes have a non-templated SolveSystem and a templated SolveSystem, but they have different numbers of inputs. Perhaps C++ 11 requires the disambiguation anyway.

BSplineCurveFit.inl
BSplineSurfaceFit.inl

83 Updates to Version 1.5

November 6, 2014. Implemented a robust algorithm for computing the distance between line segments in any dimension. Revised the PDF for computing distance between segments in 3D to describe the new algorithm. A GPU implementation is available in the sample application.

GTEngine.{h,vcxproj,vcxproj.filters}
GTBuildAll.sln
DistanceSegmentsRobust.{h,inl}
Samples/Geometrics/DistanceSegments3/DistanceSegments3.{sln,vcxproj,vcxproj.filters,cpp}
Samples/Geometrics/DistanceSegments3/DistanceSeg3Seg3.hlsl
DistanceLine3Line3.pdf (*Distance Between Two Line Segments in 3D*)

Added a class to simplify counting how many bits of precision are needed by a sequence of expressions. This is useful to determine how large N must be in the template class BSUInteger<N> for BSRational to work properly in a geometric algorithm.

GTEngine.{h,vcxproj,vcxproj.filters}
BSPrecision.{h,cpp}

Added a copy constructor and assignment operator.

MinHeap.{h,inl}

Added an equality comparison operator.

FeatureKey.{h,inl}

November 7, 2014. Implemented the TODO items in BSRational, adding conversions to float or double that produce correctly rounded results using round-to-nearest-ties-to-even. Added std::min and std::max functions for BSNumber and BSRational.

BSNumber.h
BSRational.{h,inl}

November 8, 2014. Added static_assert statements to ensure that the templates are instantiated only for dimensions 3 or 4.

Vector3.inl
AxisAngle.{h,inl}
Rotation.inl

Fixed a build break in the explicit instantiation tests for the IsValid debug support function.

MinHeap.inl

November 9, 2014. Modified an accessor to return the actual UIntegerType object rather than a pointer to the array it manages. This allows you to query how many unsigned integers are stored by the object.

BSNumber.{h,inl}

November 18, 2014. Removed the division of delta by WHEEL_DELTA in the message handler for the message WM_MOUSEWHEEL. The documentation indicates the delta is supposed to be in multiples of WHEEL_DELTA, but finer-resolution mouse wheels can send values smaller than WHEEL_DELTA. The delta passed to Window::OnMouseWheel(delta,x,y,modifiers) is now whatever the message has stored in the WPARAM value.

Window.h
WindowSystem.cpp

November 25, 2014. Overhauled the arbitrary precision library to improve the performance and to improve readability. The unsigned integer arithmetic was factored out of BSNumber into two classes, one for arbitrary precision with storage of type std::vector and one for user-selected fixed precision with storage of type std::array. Both classes share code for the arithmetic logic unit. Many computations are now performed in-place to avoid expensive allocation, deallocation, and memory copies. A new PDF is posted that greatly expands on the library compared to the discussion in the *GPGPU Programming for Games and Science* book. The document serves as a discussion about the design of the library and a reference for how to use it. Examples are provided for using BSPrecision to determine the template parameter of UIntegerFP<N> that represents the maximum number of bits required to compute the exact results for a sequence of expressions.

GTEngine.{h,vcxproj,vcxproj.filters}
BSNumber.{h,inl}
BSRational.{h,inl}
BSPrecision.{h,cpp}
Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2DWindow.h
Samples/Geometrics/ConvexHull2D/ConvexHull2DWindow.h
Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow.cpp
Samples/Geometrics/Delaunay2D/Delaunay2DWindow.h

Samples/Geometrics/Delaunay3D/Delaunay3DWindow.h
Samples/Geometrics/DistanceSegments3/DistanceSegments3.cpp
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2DWindow.h
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2DWindow.h
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.h
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3DWindow.h
Samples/Geometrics/TriangulationCDT/TriangulationCDTWindow.h
Samples/Geometrics/TriangulationEC/TriangulationECWindow.h
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.cpp
Samples/Mathematics/Interpolation2D/Interpolation2DWindow.h
UIntegerALU32.{h,inl}
UIntegerAP32.{h,inl,cpp}
UIntegerFP32.{h,inl}
ArbitraryPrecision.pdf (*GTEngine: Arbitrary Precision Arithmetic*)

84 Updates to Version 1.4

September 27, 2014. Removed the include of `atlcomcli.h` and added a simple wrapper for the COM objects. This eliminates the dependency of GTEngine on Microsoft's Active Template Library (ATL) so that the engine compiles using Microsoft Visual Studio 2013 Express Edition (which does not ship with MFC or ATL).

WICFileIO.{h,cpp}

September 28, 2014. The engine was designed to allow you to run versions of DirectX 10 and previous. However, the default target strings to the ShaderFactory::Create* compilation functions were hard-coded to use Shader Model 5 (DirectX 11). A static member was added, std::string ShaderFactory::defaultShaderModel, whose default is "5_0". If you want to use DirectX 10.0 in any of our samples, you must add two lines of code in the main function. For example, in the sample file `GTEngine/Basics/VertexColoring/VertexColoring.cpp` there is a block of code that sets members of parameters. Add two more lines,

```
parameters.featureLevel = D3D_FEATURE_LEVEL_10_0;
ShaderFactory::defaultShaderModel = "4_0";
```

DirectX 10.0 corresponds to Shader Model 4.0. If instead you want to run the sample with DirectX 10.1, then use

```
parameters.featureLevel = D3D_FEATURE_LEVEL_10_1;
ShaderFactory::defaultShaderModel = "4_1";
```

You may select a default shader model that is less powerful than what the feature level supports. For example, you can select feature level 11.0 yet have a default shader model of "4_1" for compiling the shaders. The sample file for blended terrain had hard-coded targets provided explicitly in the calls to the shader factory; these were replaced by strings using defaultShaderModel.

ShaderFactory.{h,cpp}

Samples/Graphics/BlendedTerrain/BlendedTerrainEffect.cpp

During drag-resize of the window boundary, Window::OnResize needed to trigger a redraw. Added a call to Window::OnIdle to allow the application to redraw.

Window.cpp

October 2, 2014. Added a new sample that illustrates shaders that combine texturing and per-pixel directional lighting.

GTBuildAll.sln
Samples/Basics/DirectionalLightTexture/DirectionalLightTexture.{sln,vcxproj,vcxproj.filters,cpp}
Samples/Basics/DirectionalLightTexture/DirectionalLightTextureWindow.{h,cpp}
Samples/Basics/DirectionalLightTexture/DirectionalLightTextureEffect.{h,cpp}

October 5, 2014. Added a query IsOriented() to determine whether the mesh is orientable (all triangles in a connected component have the same topological orientation). Added a query GetComponents(...) to compute the connected components of the edge-triangle graph of the mesh.

ETManifoldMesh.{h,cpp}

Added a smaller helper class to generate a set of unique vertices from a triangle soup or from a collection of indexed triangles.

GTEngine.{h,vcxproj,vcxproj.filters}
UniqueVerticesTriangles.{h,inl}

The file did not compile when GTE_IMAGICS_ASSERT_ON_INVALID_INDEX was exposed. Changed an index i to d. Moved the code contain this preprocessor symbol to the inline file so that our test framework will trap such problems. Made similar changes to test GTE_VERIFY_PRIMITIVE_TYPE in the IndexBuffer class. The inline file is new.

Image.{h,inl,cpp}
IndexBuffer.{h,inl,cpp}

October 6, 2014. Revised the design of the sample application to be more clear about the concepts it illustrates. The original design had a 512×256 application with the left subwindow showing the rendered square and the right subwindow showing the linearized depth of the rendering. The new design has a 512×512 window and allows you to select any of 7 overlays. See the sample application webpage for details and screen captures.

Samples/Basics/MultipleRenderTargets/MultipleRenderTargets.cpp
Samples/Basics/MultipleRenderTargets/MultipleRenderTargetsWindow.{h,cpp}
Samples/Basics/MultipleRenderTargets/Shaders/MultipleRenderTargets.hlsl

October 7, 2014. Removed some of the flag testing in the Initialize call because they do not occur for structured buffers.

HLSLStructuredBuffer.cpp

Added LogWarning statements to the TestIntersection calls when a bounding sphere radius is zero. The picking system requires the bounding spheres to be computed, but the user is responsible for calling UpdateModelBound on a Visual object before passing it to the picking system. The LogWarning, when it occurs, is a hint that perhaps you did not call the update function.

BoundingSphere.cpp

The typecast of mCounterStaging in GetNumActiveElements is not necessary. In a previous version of the class, mCounterStaging was in a base class and declared as ID3D11Resource*, which at that time required the typecast.

DX11StructuredBuffer.cpp

The CPU copy of the power factor is modified in the application code but on each change the engine needed to update the GPU copy.

Samples/Graphics/BlendedTerrain/BlendedTerrainWindow.cpp

October 8, 2014. The SetEnvironment functions needed their search paths modified because of the merging of the old Numerics sample folder into Mathematics.

Samples/Mathematics/PartialSums/PartialSums.cpp
Samples/Mathematics/PlaneEstimation/PlaneEstimationWindow.cpp
Samples/Mathematics/RootFinding/RootFinding.cpp

October 12, 2014. Ported the Wild Magic code for triangulation of polygons, nested polygons, and trees of nested polygons using ear clipping.

GTBuildAll.sln
GTEngine.{h,vcxproj,vcxproj.filters}
TriangulationByEarClipping.pdf (*Triangulation by Ear Clipping*)

TriangulateEC.{h,inl}
Samples/Geometrics/TriangulationEC/TriangulationEC.{sln,vcxproj,vcxproj.filters,cpp}
Samples/Geometrics/TriangulationEC/TriangulationECWindow.{h,cpp}

October 13, 2014. The segment-object and ray-object tests use the line-object test to produce a t-interval of intersection corresponding to the line. The t-interval is then intersected with the segment t-interval or ray t-interval to determine the final intersection. The interval-interval test results needed to be used in the final determination. This was a porting error from Wild Magic to GTEngine.

IntrRay3Capsule3.inl
IntrRay3Cone3.inl
IntrRay3Cylinder3.inl
IntrRay3Ellipsoid3.inl
IntrRay3Sphere3.inl
IntrSegment3Capsule3.inl
IntrSegment3Cone3.inl
IntrSegment3Cylinder3.inl
IntrSegment3Ellipsoid3.inl
IntrSegment3Sphere3.inl

October 14, 2014. Some code contained assignments of ETManifoldMesh objects, but the class does not have a copy constructor or assignment operator. The std::map members have dynamically allocated objects, so the assignments lead to memory leaks. Implemented a Clear() function, because the application code really wants to reset the mesh objects rather than copy them, but added a copy constructor and assignment operator in case other applications do require copies.

ETManifoldMesh.{h,cpp}
GenerateMeshUV.inl
ConvexHull3.inl
Delaunay2.inl
ConstrainedDelaunay2.inl

October 15, 2014. The DoQuery function had a case where result.intersect was uninitialized on return. Restructured the code to be clearer about setting members of result and removed the Logger code. Revised the test code for both line-capsule and line-cylinder and moved it into the formal unit test suite.

IntrLine3Capsule3.inl
IntrLine3Cylinder3.{h,inl}

October 16, 2014. Added a convenient GetComponents function that returns TriangleKey objects rather than pointers from the containers of the calling ETManifoldMesh object. This allows you to clear or destroy the mesh before consuming the components.

ETManifoldMesh.{h,cpp}

Added tests for edges that have length zero and assigned reasonable weights. Such degeneracies can happen unexpectedly due to floating-point rounding errors; for example, you might start with a mesh with distinct vertices and run a smoothing filter that can lead to duplicate vertices.

GenerateMeshUV.inl

October 18, 2014. Added two member functions that can be used to address some practical problems that arise when working with planar meshes. The extensive comments in the header file explain what the problems can be and how to deal with them.

PlanarMesh.{h,inl}

October 19, 2014. Fixed the conversions from matrix or quaternion to axis-angle. The Wild Magic code had a wrapper Math<Real>::ACos(z) that tested for out-of-range z, clamping the input to [-1,1] by using if-then statements. This code was replaced by clamping using std::min and std::max, but was incorrectly implemented. Also, the implicit conversion from Rotation<N,Real> to Euler angles did not work because the caller needs to specify the order of axes. The implicit conversion was replaced by an operator() member that allows the user to specify the order. Our internal unit tests were updated accordingly.

Rotation.{h,inl}
Transform.inl

October 20, 2014. The volume equation for a dodecahedron was in error.

PlatonicSolids.pdf (*Platonic Solids (parameters, vertices, mesh connectivity)*)

October 23, 2014. Ported the min-heap template class from Wild Magic. The class is intended to be a priority queue with the additional behavior that non-root heap nodes can have their weights modified followed by an update step that restores the heap to a min-heap. This is useful in several geometric algorithms. The comments in the header file provide more details and an example use in a geometric algorithm.

GTEngine.{h,vcxproj,vcxproj.filters}
MinHeap.{h,inl}

These 2D sample applications were designed to illustrate computational geometry algorithms and draw the results in a fixed-side window. The lambda expressions used for drawing, however, require that the window not be resized because they access the screen texture whose size is determined by the initial window size. In the main function, we now set parameters.allowResize = false.

Samples/Geometrics/ConstrainedDelaunay2D/ConstrainedDelaunay2D.cpp
Samples/Geometrics/ConvexHull2D/ConvexHull2D.cpp
Samples/Geometrics/Delaunay2D/Delaunay2D.cpp
Samples/Geometrics/Delaunay2D/Delaunay2DWindow.cpp
Samples/Geometrics/MinimumAreaBox2D/MinimumAreaBox2D.cpp
Samples/Geometrics/MinimumAreaCircle2D/MinimumAreaCircle2D.cpp
Samples/Geometrics/TriangulationEC/TriangulationEC.cpp
Samples/Geometrics/TriangulationEC/TriangulationECWindow.h
Samples/Geometrics/TriangulationEC/TriangulationECWindow.cpp

Modified the `Load` functions to have a parameter that allows you to disable the pixel type check. This is useful when one application saves the file, another loads it, the pixel types are compatible, but the run-time type information does not match.

Image1.{h,inl}
Image2.{h,inl}
Image3.{h,inl}

Added an include command to ensure the file compiles if the pre-compiled header system is disabled.

GenerateMeshUVs.cpp

October 24, 2014. Added code for triangulation of a simple polygon, a polygon with holes, or a tree of nested polygons. The code uses constrained Delaunay triangulation. The sample application shows how you can take advantage of the convex hull that the code generates, where the triangles are classified as inside the polygon or outside the polygon.

GTEngine.{h,vcxproj,vcxproj.filters}
GTBuildAll.sln
TriangulateCDT.{h,inl}
Samples/Geometrics/TriangulationCDT/TriangulationCDT.{sln,vcxproj,vcxproj.filters,cpp}
Samples/Geometrics/TriangulationCDT/TriangulationCDTWindow.{h,cpp}

October 25, 2014. Added the ability to control `_ITERATOR_DEBUG_LEVEL` globally by defining values for this in `GTEngineDEF.h`. The global exposure required some header files to have includes of `GTEngineDEF.h`. In debug configurations, the maximum amount of iterator debugging is enabled in Microsoft Visual Studio. This can often lead to very long debug times, which might be painful during development. Turning it off can speed up the times, but of course you will not have the benefit of trapping iterator problems or range checking. Three files needed the `GTE_IMPEXP` flag added (DLL configurations have not yet been added, however).

GTEngineDEF.h
ApprQuery.h

Array2.h
AtomicMinMax.h
Constants.h
DCPQuery.h
FeatureKey.h
FIQuery.h
GeometricPrimitive.h
Histogram.h
IEEEBinary.h
LogToMessageBox.h
LogToOutputWindow.h
Memory.h
RangeIteration.h
RootsBisection.h
RootsBrentsMethod.h
RootsPolynomial.h
SingularValueDecomposition.h
ThreadSafeMap.h
ThreadSafeQueue.h
TIQuery.h
UniqueVerticesTriangles.h
Vector.h

On Linux and Macintosh, the file needed an include of <cstring> for memset to be defined. Microsoft Visual Studio implicitly allows this function without the include.

MassSpringArbitrary.h

85 Updates to Version 1.3

September 14, 2014. The `#Samples` and `#Tools` folder names are not friendly to Linux because of the leading hash mark. The folders were moved to the corresponding folders without the hash marks. The `#Data` subfolder of Samples was also renamed (to Data), which required changing the path handling in several sample applications. Also, the `Samples/Numerics` applications were moved to the `Samples/Mathematics` folder because there is really no important distinction between the two types (just as in Wild Magic).

GTBuildAll.sln
Samples/Basics/GaussianBlurring/GaussianBlurringWindow.cpp
Samples/Basics/MultipleRenderTargets/MultipleRenderTargetsWindow.cpp
Samples/Basics/StructuredBuffers/StructuredBuffersWindow.cpp
Samples/Basics/TextureArrays/TextureArraysWindow.cpp
Samples/Basics/Texturing/TexturingWindow.cpp
Samples/Graphics/BlendedTerrain/BlendedTerrainWindow.cpp
Samples/Graphics/Picking/PickingWindow.cpp

Samples/Imagics/Convolution/ConvolutionWindow.cpp
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitterWindow.cpp
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.cpp
Samples/Mathematics/Interpolation2D/Interpolation2DWindow.cpp

Ported several Wild Magic physics library files to GTEngine.

GTEngine.{h,vcxproj,vcxproj.filters}
MassSpringArbitrary.{h,inl}
MassSpringCurve.{h,inl}
MassSpringSurface.{h,inl}
MassSpringVolume.{h,inl}
ParticleSystem.{h,inl}
PolyhedralMassProperties.{h,inl}
RigidBody.{h,inl}

Changed the second public to private. This is a cosmetic change with no difference in the behavior of the application. In the constructor for CpuMassSpringVolume, reordered the initialization list according to C++ standards (initialization order is member order).

Samples/Physics/MassSprings3D/CpuMassSpringVolume.{h,cpp}
Samples/Physics/MassSprings3D/GpuMassSpringVolume.h

September 15, 2014. Added a base class ParametricCurve for the various curve classes. This is effectively a port and consolidation of the Wild Magic Curve{2,3}, SingleCurve{2,3}, and MultipleCurve{2,3} classes. Added set/get accessors for individual control points and weights for the B-spline and NURBS curve classes. Unit tests have been written for the base class and the derived classes.

GTEngine.{h,vcxproj,vcxproj.filters}
BezierCurve.{h,inl}
BSplineCurve.{h,inl}
NURBSCurve.{h,inl}
NaturalSplineCurve.{h,inl}
TCBSplineCurve.{h,inl}
ParametricCurve.{h,inl}

September 16, 2014. Ported the Wild Magic physics sample named Rope, which illustrates 1-dimensional mass-spring systems.

GTBuildAll.sln
Samples/Physics/Rope/Rope.{sln,vcxproj,vcxproj.filters,cpp}
Samples/Physics/Rope/RopeWindow.{h,cpp}

Samples/Physics/Rope/PhysicsModule.{h,cpp}
Samples/Data/Rope.bmp

Added classes for computing the Frenet frame for a parametric curve. Added a class for representing tube surfaces with specified medial curve and radial function. Wild Magic had such a class, but it was tied to the graphics system. The GTEngine class is independent of the graphics system.

GTEngine.{h,vcxproj,vcxproj.filters}
FrenetFrame.{h,inl}
TubeSurface.{h,inl}

September 17, 2014. Fixed an error in the comments about which knots are equispaced for an open uniform curve.

BasisFunction.h

September 18, 2014. Added a base class ParametricSurface for several of the surface classes. Added a class for computing the Darboux frame for a parametric surface. These are effectively a port of the Wild Magic class ParametricSurface.

GTEngine.{h,vcxproj,vcxproj.filters}
DarbouxFrame.{h,inl}
BSplineSurface.{h,inl}
NURBSSurface.{h,inl}
ParametricSurface.{h,inl}

The uv-coordinate generation is slow when the number of vertices is very large. The bottleneck was the iteration over the std::map objects that store the adjacent vertex information. The sparse linear system solver now creates a data structure that stores the relevant information in an array and performs much better.

GenerateMeshUV.{h,inl}

The implementation of slerp had one more term in its arrays than what the theory specifies in *D. Eberly, A fast and accurate algorithm for computing SLERP, The Journal of Graphics, GPU, and Game Tools, vol. 15, no. 3, pp. 161-176, October 21, 2011.*

Quaternion.inl

September 19, 2014. Ported the Wild Magic physics sample named Cloth, which illustrates 2-dimensional mass-spring systems.

GTBuildAll.sln
Samples/Physics/Cloth/Cloth.{sln,vcxproj,vcxproj.filters,cpp}
Samples/Physics/Cloth/ClothWindow.{h,cpp}
Samples/Physics/Cloth/PhysicsModule.{h,cpp}
Samples/Data/Cloth.bmp

Added a class for representing rectangle surfaces with specified surface but used for sampling to build triangle meshes. Wild Magic had such a class, but it was tied to the graphics system. The GTEngine class is independent of the graphics system.

RectangleSurface.{h,inl}

The linear system solver used successive replacement rather than simultaneous replacement. Switched to simultaneous replacement, using ping-pong buffers, to allow for simple multithreading. The class now accepts a parameter that specifies how many threads are dedicated to the linear system solver. Added an include of <string> so that the file would compile by itself as the only include in a cpp file.

GenerateMeshUV.{h,inl}

September 20, 2014. The member mNumBytes was set twice in the else-clause in the constructor.

Resource.cpp

September 22, 2014. An include of <iostream> was needed to access std::cout.

GenerateMeshUVsWindow.cpp

Modified the functions to return the number of actual iterations used during bisection rather than just a Boolean result.

RootsBisection.{h,inl}
IntrEllipse2Ellipse2.inl
IntrEllipsoid3Ellipsoid3.inl

September 24, 2014. Added an explicit namespace scope on ShaderFactory in the GTE_MAKE_HLSL_STRING macro so that the macro can be used outside a using namespace gte block.

ShaderFactory.h

September 25, 2014. Added a new class ComputeModel that allows you to pass information to algorithm implementations whether to use the GPU, multiple threads on the CPU, or single threading on the CPU. Derived classes can provide additional behavior such as callbacks to report progress of an algorithm.

GTEngine.{h,vcxproj,vcxproj.filters}
ComputeModel.{h,cpp}
GenerateMeshUV.{h,inl}
GenerateMeshUVs.cpp
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.cpp

The number of bytes for an image was stored as a 32-bit integer. This is a limitation for 64-bit processors that can allocate more than 2GB of memory. Modified several members to be size_t, which is a 32-bit unsigned integer on a 32-bit processor or a 64-bit unsigned integer on a 64-bit processor. Any functions that use int for 1-dimensional indices were modified to use size_t.

Image.{h,inl,cpp}
Image2.{h,inl}
Image3.{h,inl}
ImageUtility2.{h,inl,cpp}
ImageUtility3.{h,inl,cpp}

86 Updates to Version 1.2

August 30, 2014. Added the ability to specify the matrix storage convention and matrix-vector multiplication convention externally. The comments in the file explain the rationale and consequences of doing so.

GTEngineDEF.h

Added new template functions to allow conversions between N-tuples and (N+1)-tuples. These include HLift, HProject, Lift, and Project. The comments in the header file explain what each function does. The HLift and HProject functions are particularly useful for conversions between Vector3<Real> and Vector4<Real>; objects.

Vector.{h,inl}

The SetSegment and GetSegment functions needed to distinguish between disjoint line segments and contiguous line segments.

IndexBuffer.cpp

August 31, 2014. Added support for picking point primitives and line segment primitives. The picking for triangles uses intersection tests. The picking for points and segments uses distance tests. The sample application was updated to show how this works. The line-segment distance code had a porting bug that was fixed; the returned distance was incorrectly set to zero when the segment closest point was an interior point.

Picker.{h,cpp}
PickRecord.{h,inl,cpp}
DistLine3Segment3.inl
Samples/Graphics/Picking/PickingWindow.{h,cpp}

Modified our header dependency tool to trap problems when precompiled headers are disabled. The include of `GTEngine.h` in `GTEnginePCH.h` needed to be disabled to trap problems in cpp files. The following files had missing dependencies when we did this.

Camera.cpp
Command.cpp
ConstantBuffer.cpp
DrawTarget.cpp
DX11Engine.{h,cpp}
Environment.cpp
ETManifoldMesh.cpp
Fluid2InitializeSource.cpp
Fluid2InitializeState.cpp
Fluid3InitializeSource.cpp
Fluid3InitializeState.cpp
Histogram.cpp
HLSLBaseBuffer.h
HLSLParameter.h
HLSLResource.h
HLSLShader.h
HLSLShader.cpp
HLSLShaderVariable.h
HLSLShaderType.h
IEEEBinary16.cpp
Image.cpp
IndexBuffer.cpp
LightingEffect.cpp
Logger.cpp
LogToMessageBox.cpp
LogToOutputWindow.cpp
MarchingCubes.cpp
OverlayEffect.cpp
Resource.cpp
Texture.cpp
TextEffect.cpp
Timer.cpp

TriangleKey.cpp
TSManifoldMesh.cpp
VertexFormat.cpp
VEManifoldMesh.cpp
WICFileIO.cpp
Window.cpp

September 3, 2014. Changed the LogError messages to LogInformation when an insertion will violate the manifold mesh requirement. Some applications might find it useful to attempt an insertion, using the returned null pointer as an indication to take an alternative action. You can set your Logger listeners to ignore LogInformation messages (or just ignore ones that are sent to MessageBox or OutputWindow); however, we also added the ability to turn off the LogInformation message specifically for these classes via a class member function.

VEManifoldMesh.{h,cpp}
ETManifoldMesh.{h,cpp}
TSManifoldMesh.{h,cpp}

Added class member function to get 1-dimensional indices or tuples for various neighborhood configurations of a pixel. Modified the image processing utilities to use these member functions. Added a static assertion to the image classes that requires the PixelType to be trivially copyable, a requirement of the design of the Image classes. Removed the PixelRGBA8 and PixelBGRA8 classes because the engine does not use them (nor are they that useful). Removed the `GteStandardImages.cpp` file that explicitly instantiated image classes. The engine design has been not to force explicit instantiation of template classes. The mass-spring system class used Image3<Vector3<float>>, but the template type is not trivially copyable. The code was easily modified to use std::vector<Vector3<float>>.

GTEngine.{vcxproj,vcxproj.filters}
GTEngineDEF.h
Image1.{h,inl}
Image2.{h,inl}
Image3.{h,inl}
ImageUtility2.{h,cpp}
ImageUtility3.{h,cpp}
Samples/Physics/MassSprings3D/CpuMassSpringVolume.{h,cpp}
Samples/Physics/MassSprings3D/MassSprings3DWindow.cpp

September 11, 2014. Ported the Wild Magic B-spline curve and surface code for least-squares fitting of data, including the sample applications.

GTEngine.{h,vcxproj,vcxproj.filters}
GTBuildAll.sln
BSplineCurveFit.{h,inl}
BSplineSurfaceFit.{h,inl}

Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitter.{sln,vcxproj,vcxproj.filters,cpp}
Samples/Mathematics/BSplineCurveFitter/BSplineCurveFitterWindow.{h,cpp}
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitter.{sln,vcxproj,vcxproj.filters,cpp}
Samples/Mathematics/BSplineSurfaceFitter/BSplineSurfaceFitterWindow.{h,cpp}

The mKeys member of the class was a std::vector type and is used heavily in the range-iteration-based for-loop in GetIndex. Because of the checked iterators used in that loop, the performance in debug builds was horrific. Changed the type of mKeys to a native array to improve the performance for debugging.

BasisFunction.{h,inl}

In order to re-use a MeshFactory object with a different vertex format, the array that keeps track of whether texture coordinate channels occur had to be cleared when creating the vertex buffer.

MeshFactory.cpp

Added a new class that generates texture coordinates automatically for a mesh with rectangle or disk topology. The algorithm is based on barycentric mapping, mean-value weights, and Gauss-Seidel iteration for large sparse linear systems. Added a new class that manages a planar mesh and allows fast and exact point-in-triangle queries and computation of barycentric coordinates. This was added to support resampling of meshes with automatically generated texture coordinates.

GTEngine.{h,vcxproj,vcxproj.filters}
GenerateMeshUV.{h,inl}
PlanarMesh.{h,inl}

Fixed a comment that was cut-and-paste from another similar file.

LightDirectionPerPixelEffect.h

September 13, 2014. A new sample application that demonstrates using the class GenerateMeshUV to automatically generate texture coordinates for a mesh that has rectangle or disk topology. The sample also illustrates resampling using the class PlanarMesh.

GTBuildAll.sln
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVs.{sln,vcxproj,vcxproj.filters,cpp}
Samples/Mathematics/GenerateMeshUVs/GenerateMeshUVsWindow.{h,cpp}

87 Updates to Version 1.1

August 26, 2014. Refactored the basic lighting effects to share code, fixing several bugs in the process. Two new classes were added, LightingConstants and LightingEffect. Added a new sample application to illustrate the effects; this is a port of the Wild Magic 5 Lights sample.

GTEngine.{h,vcxproj,vcxproj.filters
GTBuildAll.sln
VisualEffect.{h,cpp}
LightAmbientEffect.{h,cpp}
LightDirectionPerPixelEffect.{h,cpp}
LightDirectionPerVertexEffect.{h,cpp}
LightPointPerPixelEffect.{h,cpp}
LightPointPerVertexEffect.{h,cpp}
LightSpotPerPixelEffect.{h,cpp}
LightSpotPerPixelEffect.{h,cpp}
Samples/Graphics/Lights/Lights.{sln,vcxproj,vcxproj.filters,cpp}
Samples/Graphics/Lights/LightsWindow.{h,cpp}
LightingConstants.{h,cpp}
LightingEffect.{h,inl,cpp}
LightAmbientEffect.inl
LightDirectionPerVertexEffect.inl
LightDirectionPerPixelEffect.inl
LightPointPerVertexEffect.inl
LightPointPerPixelEffect.inl
LightSpotPerVertexEffect.inl
LightSpotPerPixelEffect.inl

The MSDN documentation for ID3D11DeviceContext::IASetVertexBuffers and ID3D11DeviceContext::Draw(numVertices, startVertex)¡ appears not to mention that startVertex is relative to the offsets[] passed to IASetVertexBuffers. Fixed the enabling of vertex buffers to set the offsets to zero. The DX11Engine::DrawPrimitive function passes the vertex buffer offset through the startVertex parameter.

DX11VertexBuffer.cpp

August 29, 2014. Added HasMember functions to test for the existence of member data with the specified name.

ConstantBuffer.{h,cpp}
TextureBuffer.{h,cpp}

A small amount of refactoring of Picker to prepare for supporting point and line primitives (distance based). The triangle picking code was tested using a new sample application.

GTBuildAll.sln
Samples/Graphics/Picking/Picking.{sln,vcxproj,vcxproj,cpp}
Samples/Graphics/Picking/PickingWindow.{h,cpp}

Fixed a bug in the Shader constructor when generating the member layouts for constant buffers and texture buffers. The indices into the layout arrays were not incremented, which led to incorrect behavior for shaders with two or more such buffers.

Shader.cpp

Fixed a bug in the GenerateLayout member function. The shader type's offset was used to determine member offset in the structure, but the shader variable's offset has to be used instead (at the root of the recursion).

HLSLBaseBuffer.cpp

A change in the semantics for Environment::GetPath when no directories exist was not propagated to the append-consume buffer sample application, causing a bogus assertion when trying to find the HLSL file associated with the application.

Samples/Basics/AppendConsumeBuffers/AppendConsumeBuffers.cpp

The hiding of the trackball rotation matrix inside the Window class prevented picking from working properly. The registration system that updated shader constants with projection-view-world transforms from the world transforms of associated objects needed a redesign. The trackball rotation is no longer applied implicitly. You can access it via a public member function. The Register/Unregister calls were replaced by two subscription interfaces, one to subscribe to changes in the camera changes and have the pvw-matrices automatically updated—this is what the Register/Unregister system did. However, there is interaction between updating world transforms and pvw-matrices for effects, so a second subscription interface is used to have world transforms of objects automatically updated when the virtual trackball moves. The redesign led to many changes in sample applications. While we were in there, we cleaned up and refactored some of the samples to make the code more readable (consistent use of bool SetEnvironment and bool CreateScene). The Window¡ class also has a new member function for computing a picking line for the current viewport and camera settings. This is used in conjunction with the Picker class.

Window.{h,inl,cpp}
Samples/Basics/GaussianBlurring/GaussianBlurringWindow.{h,cpp}
Samples/Basics/GeometryShaders/GeometryShadersWindow.{h,cpp}
Samples/Basics/MultipleRenderTargets/MultipleRenderTargetsWindow.{h,cpp}
Samples/Basics/StructuredBuffers/StructuredBuffersWindow.{h,cpp}
Samples/Basics/TextureArrays/TextureArraysWindow.{h,cpp}
Samples/Basics/Texturing/TexturingWindow.{h,cpp}
Samples/Basics/VertexColoring/VertexColoringWindow.{h,cpp}

Samples/Geometrics/AllPairsTriangles/AllPairsTrianglesWindow.cpp
Samples/Geometrics/ConvexHull3D/ConvexHull3DWindow.{h,cpp}
Samples/Geometrics/Delaunay3D/Delaunay3DWindow.{h,cpp}
Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow.{h,cpp}
Samples/Geometrics/MinimumVolumeSphere3D/MinimumVolumeSphere3DWindow.cpp
Samples/Geometrics/ShortestPath/ShortestPathWindow.{h,cpp}
Samples/Graphics/BlendedTerrain/BlendedTerrainWindow.cpp
Samples/Graphics/BlownGlass/BlownGlassWindow.{h,cpp}
Samples/Graphics/Lights/LightsWindow.cpp
Samples/Graphics/PlaneMeshIntersection/PlaneMeshIntersectionWindow.{h,cpp}
Samples/Graphics/WireMesh/WireMeshWindow.{h,cpp}
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.cpp
Samples/Mathematics/Interpolation2D/Interpolation2DWindow.cpp
Samples/Physics/Fluids2D/Fluids2DWindow.{h,cpp}
Samples/Physics/Fluids3D/Fluids3DWindow.{h,cpp}
Samples/Physics/MassSprings3D/MassSprings3DWindow.{h,cpp}

88 Updates to Version 1.0

August 13, 2014. The files were missing the `#pragma once` guards against multiple inclusions.

Constants.h
MarchingCubesTable.h
SingularValueDecomposition.h

August 15, 2014. Fixed a typographical error in Equation (14). Replaced the LaTeX 2$_\varepsilon$verbatim commands with lstlisting commands for more readable pseudocode.

AreaIntersectingEllipses.pdf (*The Area of Intersecting Ellipses*)

August 17, 2014. These modifications are related to the porting of the Wild Magic interpolation code to GTEngine.

Ported most of the Wild Magic interpolation code to GTEngine. A sample application illustrates use of the interpolators for 2D height-field data.

GTEngine.{h,vcxproj,vcxproj.filters}
IntpAkima1.{h,inl}
IntpAkimaUniform1.{h,inl}
IntpAkimaUniform2.{h,inl}
IntpAkimaUniform3.{h,inl}
IntpBicubic2.{h,inl}
IntpBilinear2.{h,inl}

IntpLinearNonuniform2.{h,inl}
IntpQuadraticNonuniform2.{h,inl}
IntpSphere2.{h,inl}
IntpThinPlateSpline2.{h,inl}
IntpThinPlateSpline3.{h,inl}
IntpTricubic3.{h,inl}
IntpTrilinear3.{h,inl}
IntpVectorField2.{h,inl}
Samples/Mathematics/Interpolation2D/Interpolation2D.{sln,vcxproj,vcxproj.filters,cpp}
Samples/Mathematics/Interpolation2D/Interpolation2DWindow.{h,cpp}
Samples/Data/Checkerboard.png

Added wrappers for the meshes produced by Delaunay triangulation and tetrahedralization. Access to the input vertices of the DelaunayN classes was required to support this. The wrappers allow the interpolators to interact with general triangles mesh that are not produced by the Delaunay code. (Wild Magic forced you to use Delaunay meshes.)

Delaunay2.{h,inl}
Delaunay3.{h,inl}
Delaunay2Mesh.{h,inl}
Delaunay3Mesh.{h,inl}

Added a mesh creator for regular triangle meshes ('half' a rectangle mesh).

MeshFactory.{h,cpp}

When computing barycentric coordinates, replaced the separate Dot and Cross functions by the single calls to DotCross.

Vector3.inl

Added functions AllocateMapN and DeallocateMapN for $N = 2, 3, 4$. This allows for wrapping an already existing 1-dimensional array with multidimensional array access.

Memory.{h,inl}

August 19, 2014. These modifications are related to getting the code, not including the Windows-specific graphics and application code, to compile on Linux and Macintosh OS X. The first batch of changes were consequences of trying to compile on Macintosh OS X. The second batch is due to Linux compilation, although changes in the first batch no doubt were needed anyway on Linux.

Hide the Windows-specific information from the other platforms.

GTEngine.h
GTEngineDEF.h

The compiler did not like the #pragma once in the precompiled header file.

GTEnginePCH.h

The LLVM compiler successfully compiled Delaunay2Mesh and Delaunay3Mesh in our header verification tool when you compile in a single file. However, when a build-project is initiated, LLVM attempted to use the template parameter Rational as if it were an actual type and complained when trying to instantiate Vector{2,3}::ComputeBarycentrics. In particular, we had a line if (std::abs(det) > epsilon)for which LLVM said std::abs is ambiguous and that it did not match any of the standard functions for floating-point types. For now, we modified the code for ComputeBarycentrics to use if (det < -epsilon || det > epsilon).

Vector2.inl
Vector3.inl

LLVM does not have versions of the secure memcpy_s that Microsoft Visual Studio does. Added conditional compilation to handle this.

GaussianElimination.inl
LinearSystem.inl
MinimizeN.inl

Eliminated the typedef of Function. LLVM had difficulties with this in the derived classes. Removed the (WM5) using statements and added the explicit scoping by this→.

OdeSolver.{h,inl}
OdeEuler.{h,inl}
OdeImplicitEuler.{h,inl}
OdeMidpoint.{h,inl}
OdeRungeKutta4.{h,inl}

Removed the explicit last parameter to ComputeBarycentrics. The default parameter has the same value.

IntpQuadraticNonuniform2.inl

The file <iterator> needed to be included to access the definitions of std::begin, std::end, and std::reverse_iterator. Microsoft Visual Studio 2013 allowed the compilation without it, but not LLVM.

RangeIteration.h

The class had two member functions that required two implicit conversions to return values. One of them has to be explicit, and LLVM requires this.

Transform.inl

The definition of size_t appears to be built into Microsoft Visual Studio 2013. On LLVM, it is defined in <cstddef>, so this header file needed to be included.

Memory.h

The declaration for triangle barycentric coordinates was the incorrect size.

DistPoint3Triangle3.h
DistTriangle3Triangle3.h

The header needed to include <set> because the class has a member using this container.

ConvexHull3.h

We have an internal tool that include each header file in a cpp file and tests whether it compiles. This is designed to expose a header file that might not include any dependencies it has. If the header is for template classes or functions, we also explicitly instantiate those in order to trap any compiler errors. Naturally, the tool has precompiled headers disabled. Surprisingly, some code passed the tests when compiling with Microsoft Visual Studio 2013 but the same code failed compilation on LLVM. For example, the GetContainer function in `GteContEllipsoid3.inl` incorrectly had a Vector2 input when it should have been Vector3. This code passed the explicit instantiation compiler test using Microsoft Visual Studio 2013.

ContEllipsoid3.inl

Microsoft's compilers still violate the ANSI C++ standard that requires template derived classes to explicitly scope any access to template base class members or functions via this→mBaseMember and this→BaseFunction. In Wild Magic, we chose the using statement in the derived-class headers but decided against this in GTEngine because of the potential change in access rights (public, protected, private).

ConstrainedDelaunay2.inl
DistPoint2OrientedBox2.inl
DistPoint3OrientedBox3.inl

IntpAkimaUniform1.inl
IntrLine2OrientedBox2.inl
IntrLine3OrientedBox3.inl
IntrRay2OrientedBox2.inl
IntrRay3OrientedBox3.inl
IntrSegment2OrientedBox2.inl
IntrSegment3OrientedBox3.inl
Matrix2x2.inl
Matrix3x3.inl
Matrix4x4.inl
Quaternion.inl
Transform.inl
Vector2.inl
Vector3.inl
Vector4.inl

Added typename modifiers to some variable declarations. Microsoft Visual Studio 2013 allowed the declarations without the modifiers, but not LLVM.

BSNumber.inl
Delaunay2Mesh.inl
Delaunay3Mesh.inl
DistRectangle3Rectangle3.inl
DistTriangle3Triangle3.inl

BSNumber had a declaration to make BSRational a friend. Microsoft Visual Studio 2013 allowed the original code, but not LLVM. Forward declared the BSRational template and modified the friend statement. The same mechanism had been used in Wild Magic to satisfy all compilers.

BSNumber.h

Microsoft Visual Studio 2013 allows you to make calls to math library functions such as std::abs and sqrt without including <cmath>. Added the include to satisfy LLVM.

BandedMatrix.h
GaussianElimination.h
GVector.h
IntpAkima1.h
IntpAkimaUniform2.h
IntpAkimaUniform3.h
Minimize1.h
RootsBrentsMethod.h
SingularValueDecomposition.h
SymmetricEigensolver.h

Vector.h

Removed an incorrect static cast of a 32-bit value to a 64-bit value when the return value of the function is in fact 32-bit.

BitHacks.cpp

The explicit instantiation of the oppositeFace array required a `template <>` modifier.

TetrahedronKey.cpp

The explicit instantiation of image classes needed to be inside a namespace block.

StandardImages.cpp

Removed the AlignedMemory class from the engine. It is not used, and the `_aligned_malloc` and `_aligned_free` calls do not exist on Macintosh OS X.

GTEngine.{h,vcxproj,vcxproj.filters}
AlignedMemory.{h,inl}

The changes below were necessary after the previous ones to get the code to compile on Linux.

The explicit instantiations had to occur inside a namespace block. A `using namespace gte` was not sufficient.

EdgeKey.cpp
TriangleKey.cpp
TetrahedronKey.cpp

The class uses the var-args system so we needed to include `<cstdarg>`.

Image.cpp

The files use `std::numeric_limits` and needed to include `<limits>`.

ImageUtility2.cpp
DistLine3Rectangle3.h
DistLine3Triangle3.h

DistPoint3Circle3.h
DistPoint3Tetrahedron3.h
IntrIntervals.h
IntrLine2AlignedBox2.h
IntrLine2Line2.h
IntrLine3AlignedBox3.h
IntrLine3Plane3.h
IntrCircle2Circle2.h
NearestNeighborQuery.h

The files use memcpy or memset and needed to include <cstring>. The conditional compilation is now based on WIN32 (or not).

BasisFunction.h
GaussianElimination.inl
IntpAkimaUniform2.h
IntpAkimaUniform3.h
LinearSystem.inl
MinimizeN.{h,inl}
SurfaceExtractor.h
SingularValueDecomposition.{h,inl}
SymmetricEigensolver.{h,inl}

The files use std::min, std::max, std::sort, or std::nth_element and needed to include <algorithm>.

BSNumber.h
ConvexHull2.h
ContEllipse2MinCR.h
MinimumAreaCircle2.h
MinimumVolumeSphere3.h
NearestNeighborQuery.h