

Geometric Tools Engine Update History

Last modified: August 29, 2019

Contents

1	Updates to Version 3.27	5
2	Updates to Version 3.26	5
3	Updates to Version 3.25	5
4	Updates to Version 3.24	24
5	Updates to Version 3.23	24
6	Updates to Version 3.22	29
7	Updates to Version 3.21	34
8	Updates to Version 3.20	37
9	Updates to Version 3.19	38
10	Updates to Version 3.18	39
11	Updates to Version 3.17	40
12	Updates to Version 3.16	42
13	Updates to Version 3.15	48
14	Updates to Version 3.14	50
15	Updates to Version 3.13	51
16	Updates to Version 3.12	53
17	Updates to Version 3.11	55
18	Updates to Version 3.10	57
19	Updates to Version 3.9	58
20	Updates to Version 3.8	61
21	Updates to Version 3.7	62
22	Updates to Version 3.6	63
23	Updates to Version 3.5	63
24	Updates to Version 3.4	64
25	Updates to Version 3.3	66
26	Updates to Version 3.2	68
27	Updates to Version 3.1	74
28	Updates to Version 3.0	76

29 Updates to Version 2.5	78
30 Updates to Version 2.4	83
31 Updates to Version 2.3	88
32 Updates to Version 2.2	94
33 Updates to Version 2.1	105
34 Updates to Version 2.0	106
35 Updates to Version 1.14	129
36 Updates to Version 1.13	132
37 Updates to Version 1.12	133
38 Updates to Version 1.11	137
39 Updates to Version 1.10	137
40 Updates to Version 1.9	139
41 Updates to Version 1.8	142
42 Updates to Version 1.7	144
43 Updates to Version 1.6	147
44 Updates to Version 1.5	161
45 Updates to Version 1.4	163
46 Updates to Version 1.3	169
47 Updates to Version 1.2	173
48 Updates to Version 1.1	177
49 Updates to Version 1.0	179

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

- 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 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`

2 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`

3 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 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::lisp` 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 cone 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.

[IntrBSplineUniform.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
SingularValueDecomposition.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
SingularValueDecomposition.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
SingularValueDecomposition.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}.gsl
 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}.gsl
 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}.gsl
 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}.gsl
 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}.gsl
 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}.gsl
 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.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/BoundaryNeumann.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/GaussianBlur.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/BoundaryDirichlet.cs.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/BoundaryNeumann.cs.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur2/Shaders/GaussianBlur.cs.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur3/GpuGaussianBlur3.*.vcxproj*
Samples/Imagics/GpuGaussianBlur3/GpuGaussianBlur3Window.{h,cpp}
Samples/Imagics/GpuGaussianBlur3/Shaders/BoundaryDirichlet.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/DrawImage.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/BoundaryNeumann.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/GaussianBlur.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/BoundaryDirichlet.cs.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/BoundaryNeumann.cs.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/DrawImage.ps.{hsl,gsl}
Samples/Imagics/GpuGaussianBlur3/Shaders/GaussianBlur.cs.{hsl,gsl}
Samples/Imagics/MedianFiltering/MedianFiltering.*.vcxproj*
Samples/Imagics/MedianFiltering/MedianFilteringWindow.{h,cpp}
Samples/Imagics/MedianFiltering/Shaders/Median3x3.{hsl,gsl}
Samples/Imagics/MedianFiltering/Shaders/Median5x5.{hsl,gsl}
Samples/Imagics/MedianFiltering/Shaders/MedianBySort.{hsl,gsl}
Samples/Imagics/MedianFiltering/Shaders/MedianShared.hsl
Samples/Imagics/MedianFiltering/Shaders/Median3x3.cs.{hsl,gsl}
Samples/Imagics/MedianFiltering/Shaders/Median5x5.cs.{hsl,gsl}
Samples/Imagics/MedianFiltering/Shaders/MedianBySort.cs.{hsl,gsl}
Samples/Imagics/MedianFiltering/Shaders/MedianShared.cs.hsl
Samples/Imagics/SurfaceExtraction/SurfaceExtraction.*.vcxproj*
Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.{h,cpp}
Samples/Imagics/SurfaceExtraction/Shaders/DrawSurfaceIndirect.hsl
Samples/Imagics/SurfaceExtraction/Shaders/DrawSurfaceIndirect{VS,GS,PS}.gsl
Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurface.{hsl,gsl}
Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurfaceIndirect.{hsl,gsl}
Samples/Imagics/SurfaceExtraction/Shaders/DrawSurfaceIndirect.vs,gs,ps.hsl
Samples/Imagics/SurfaceExtraction/Shaders/DrawSurfaceIndirect.{vs,gs,ps}.gsl
Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurface.cs.{hsl,gsl}
Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurfaceIndirect.cs.{hsl,gsl}
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.hsl
Samples/Mathematics/AllPairsTriangles/Shaders/DrawUsingVertexID{VS,PS}.gsl
Samples/Mathematics/AllPairsTriangles/Shaders/InitializeColors.{hsl,gsl}
Samples/Mathematics/AllPairsTriangles/Shaders/TriangleIntersection.{hsl,gsl}
Samples/Mathematics/AllPairsTriangles/Shaders/VertexColorIndexed.hsl
Samples/Mathematics/AllPairsTriangles/Shaders/VertexColorIndexed{VS,PS}.gsl
Samples/Mathematics/AllPairsTriangles/Shaders/DrawUsingVertexID.{vs,ps}.{hsl,gsl}
Samples/Mathematics/AllPairsTriangles/Shaders/InitializeColors.cs.{hsl,gsl}

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,gsl}
 Samples/Mathematics/PlaneEstimation/Shaders/PlaneVisualize.ps.{hlsl,gsl}
 Samples/Mathematics/PlaneEstimation/Shaders/PositionVisualize.ps.{hlsl,gsl}
 Samples/Mathematics/RootFinding/RootFinding.*.vcxproj*
 Samples/Mathematics/RootFinding/RootFinding.cpp
 Samples/Mathematics/RootFinding/Shaders/RootFinder.{hlsl,gsl}
 Samples/Mathematics/RootFinding/Shaders/RootFinder.cs.{hlsl,gsl}
 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,gsl}
 Samples/Mathematics/ShortestPath/Shaders/InitializeDiagToRow.{hlsl,gsl}
 Samples/Mathematics/ShortestPath/Shaders/PartialSumsDiagToCol.{hlsl,gsl}
 Samples/Mathematics/ShortestPath/Shaders/PartialSumsDiagToRow.{hlsl,gsl}
 Samples/Mathematics/ShortestPath/Shaders/UpdateShader.{hlsl,gsl}
 Samples/Mathematics/ShortestPath/Shaders/WeightsShader.{hlsl,gsl}
 Samples/Mathematics/ShortestPath/Shaders/InitializeDiagToCol.cs.{hlsl,gsl}
 Samples/Mathematics/ShortestPath/Shaders/InitializeDiagToRow.cs.{hlsl,gsl}
 Samples/Mathematics/ShortestPath/Shaders/PartialSumsDiagToCol.cs.{hlsl,gsl}
 Samples/Mathematics/ShortestPath/Shaders/PartialSumsDiagToRow.cs.{hlsl,gsl}
 Samples/Mathematics/ShortestPath/Shaders/UpdateShader.cs.{hlsl,gsl}
 Samples/Mathematics/ShortestPath/Shaders/WeightsShader.cs.{hlsl,gsl}
 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,gsl}
 Samples/Physics/Fluids2D/Shaders/DrawDensity.ps.{hlsl,gsl}
 Samples/Physics/Fluids3D/Fluids3D.*.vcxproj*
 Samples/Physics/Fluids3D/Fluids3DWindow.{h,cpp}
 Samples/Physics/Fluids3D/Shaders/VolumeRender.hlsl
 Samples/Physics/Fluids3D/Shaders/VolumeRenderVS.gsl
 Samples/Physics/Fluids3D/Shaders/VolumeRenderPS.gsl
 Samples/Physics/Fluids3D/Shaders/VolumeRender.vs.{hlsl,gsl}
 Samples/Physics/Fluids3D/Shaders/VolumeRender.ps.{hlsl,gsl}
 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}.gsl
 Samples/Physics/MassSprings3D/Shaders/RungeKutta.hlsl
 Samples/Physics/MassSprings3D/Shaders/RungeKutta{1a,1b,2a,2b,3a,3b,4a,4b}.gsl
 Samples/Physics/MassSprings3D/Shaders/DrawUsingVertexID.{vs,ps}.{hlsl,gsl}
 Samples/Physics/MassSprings3D/Shaders/RungeKutta.cs.hlsl
 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.gsl
 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.gsl](#) 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 the `WGLEngine` 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`
`GTEApplication.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
```

4 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
```

5 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 [DistancePointHyperellipsoid.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

6 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 [GeodesicHeightField](#).

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 templated 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}
```

7 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}
```

8 Updates to Version 3.20

January 17, 2019. Added new files. Two type traits were created in [GteMath.h](#), one is [is_arbitrary_precision](#) to 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 division 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
```

9 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`

10 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
```

11 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
```

12 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 integral 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`.

13 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}
```

14 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 avoid 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}](#)

15 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 `GreatestCommonDivisor` 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
```

16 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 minimum-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 occurring 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

17 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 initializer 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 compiler 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

18 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 [GteFunctionsBSNumber.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](#)

19 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 perspective 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
```

20 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](#)

21 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
```

22 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}
```

23 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`

24 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

25 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 create 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 were 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}](#)

26 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`.

```
MinimumAreaBox2.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  $2 \times 2$  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}
```

27 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/GteWGLEExtensions.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. Redesigning [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

28 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
```

29 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 instead 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](#)

30 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

31 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
```

32 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 `Convert` functions. 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 [PixelFormat](#) 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 [PixelFormat](#).

```
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*)

33 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}.gsl

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.gsl

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

34 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}.gsl

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}.gsl

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.gsl

```
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.{gsl,hsl}
 Samples/Basics/ShaderReflection/ShaderReflection.{v12,v14}.{vcxproj,vcxproj.filters}
 Samples/Basics/ShaderReflection/ShaderReflection.cpp
 Samples/Basics/ShaderReflection/AppendConsume.{gsl,hsl}
 Samples/Graphics/GeometryShaders/GeometryShadersWindow.{h,cpp}
 Samples/Graphics/GeometryShaders/Shaders/RandomSquaresIndirect{GS,VS}.gsl
 Samples/Graphics/StructuredBuffers/StructuredBuffers.{v12,v14}.{sln,vcxproj,vcxproj.filters}
 Samples/Graphics/StructuredBuffers/StructuredBuffersWindow.cpp
 Samples/Graphics/StructuredBuffers/Shaders/StructuredBuffers{PS,VS}.gsl
 Samples/Imagics/SurfaceExtraction/SurfaceExtraction.{v12,v14}.{sln,vcxproj,vcxproj.filters}
 Samples/Imagics/SurfaceExtraction/SurfaceExtractionWindow.{h,cpp}
 Samples/Imagics/SurfaceExtraction/Shaders/DrawSurfaceIndirect{GS,PS,VS}.gsl
 Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurface.gsl
 Samples/Imagics/SurfaceExtraction/Shaders/ExtractSurfaceIndirect.gsl
 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.h``GEObject.{h,cpp}``GTGraphics.h`
(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,}.gsl
Samples/Geometrics/AllPairsTriangles/Shaders/InitializeColors.gsl
Samples/Geometrics/AllPairsTriangles/Shaders/TriangleIntersection.gsl
Samples/Geometrics/AllPairsTriangles/Shaders/VertexColorIndexed{VS,}.gsl

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,gsl}
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.gsl
Samples/Graphics/ShortestPath/Shaders/InitializeDiagToRow.gsl
Samples/Graphics/ShortestPath/Shaders/PartialSumsDiagToCol.gsl
Samples/Graphics/ShortestPath/Shaders/PartialSumsDiagToRow.gsl
Samples/Graphics/ShortestPath/Shaders/UpdateShader.gsl
Samples/Graphics/ShortestPath/Shaders/WeightsShader.gsl

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](#)).

35 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 UTF8 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 [MouseClicked](#) 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.

36 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

37 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?”

38 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*)

39 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

40 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

41 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.IsParameter[0]` when it should have used `result.IsParameter[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.hsl

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`

42 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 [TIntervalInterval](#) and [FIntervalInterval](#).

```
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 [TAlignedBox3Cone3](#) 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*)

43 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

44 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.hsl
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*)

45 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/Shader/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

46 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}
```

47 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}
```


48 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}
LightSpotPerVertexEffect.{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}

49 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_{ϵ} 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 `<cstdlib>`, 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

IntrAkimaUniform1.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
IntrAkima1.h
IntrAkimaUniform2.h
IntrAkimaUniform3.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 `<stdarg.h>`.

Image.cpp

The files use `std::numeric_limits` and needed to include `<limits.h>`.

```
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