Geometric Tools Engine 8 Update History

Contents

1 Updates to Version 8.1 2

2 Updates to Version 8.0 3

3 Initial Distribution of Version 8.0 6

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

- Version 8.1 posted November 24, 2025.
- Version 8.0 posted May 12, 2025.

The updated files and related notes are provided for the versions in each of the ensuing sections. Modified files are colored gold, new files are colored green, and deleted files are colored red. Source code is colored Violet. Source code modifications that occur in a section *Updates to Version* $x.y$ are pushed to github and uploaded to the Geometric Tools website. However, the zip file is still for version $x.y$. The modified files will be contained in the zip file for version $x.(y+1)$ and posted once that version is ready for shipping.

1 Updates to Version 8.1

February 3, 2026. The GTE_MAJOR_VERSION and GTE_MINOR_VERSION values are out of date. They were set to 7 and 3, respectively, when they should be 8 and 1, respectively.

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

January 13, 2026. Added solution, project, and filter files for Microsoft Visual Studio 2026. Made some minor formatting adjustments to the source file.

GTE/Tools/BitmapFontCreator/BitmapFontCreator.v17.vcxproj
GTE/Tools/BitmapFontCreator/BitmapFontCreator.v18.sln
GTE/Tools/BitmapFontCreator/BitmapFontCreator.v18.vcxproj
GTE/Tools/BitmapFontCreator/BitmapFontCreator.v18.vcxproj.filters
GTE/Tools/BitmapFontCreator/BitmapFontCreator.cpp

Removed tools that were specific to development, testing, and code maintenance.

GTE/Tools/ChangePlatformToolset/*
GTE/Tools/FiniteDifferences/*
GTE/Tools/GenerateApproximations/*
GTE/Tools/GenerateOpenGLWrapper/*
GTE/Tools/PrecisionCalculator/*
GTE/Tools/RotationApproximation/*

January 12, 2026. I added Microsoft Visual Studio 2026 files for solutions, projects, and filters. There are nearly 400 such triples of files (.v18.sln, .v18.vcxproj, .v18.vcxproj.filters).

Set all v17 (MSVS 2022) solution files to use VisualStudioVersion 17.0.32014.148. Fixed the GenerateProject tool to use GL46 instead of the obsolete GL45. Added code to generate v18 (MSVS 2026) solution, project and filter files.

GTE/BuildAll.v17.sln
GTE/BuildAllDX11.v17.sln
GTE/BuildAllGL46.v17.sln
GTE/Samples/CSharpCppManaged.v17.sln
GTE/Tools/GenerateProject/*.*

January 11, 2026. Modified copyright notice to include 2026. All source files were modified. The file version numbers remain the same.

January 3, 2026. Added intersection queries for a triangle and a canonical box or a triangle and an aligned box. The code for a triangle and an oriented box already existed. Modified the sizes and positions of the triangle and oriented box in the sample application; the previous triangle was too small and the previous box was too large.

GTE/GTMathematics.v17.{vcxproj,vcxproj.filters}
GTE/Mathematics/IntrTriangle3CanonicalBox3.h
GTE/Mathematics/IntrTriangle3AlignedBox3.h
GTE/Mathematics/IntrTriangle3OrientedBox3.h
GTE/Samples/Intersection/IntersectTriangleBox/IntersectTriangleBoxWindow3.cpp

December 31, 2025. Despite using Mathematica to verify the mathematical equations and updating the PDF for box-cylinder intersection, I still failed to fix the bug mentioned in the December 26, 2025 report. In fact, the previous report broke the case for (U2,-D). The case for (D,-U2) had an incorrect divisor. These have now been fixed.

GTE/Mathematics/IntrCanonicalBox3Cylinder3.h

December 26, 2025. Fixed the lower and upper calculations for case (U2,-D). Thanks to *OttoX* at github for reporting associated bugs.

GTE/Mathematics/IntrCanonicalBox3Cylinder3.h

2 Updates to Version 8.0

November 24, 2025. Added install targets for the mathematics header-only library. Thanks to *BasketsAlaaf* at github for writing the cmake support and making the pull request.

GTE/Mathematics/CMakeLists
GTE/Mathematics/cmake/gtmathematicsConfig.cmake.in

The private Set function had an if-else statement that was designed to provide fast copying for floating-point Real and object copying for rational Real. The if-else statement provides flow control at run time. However, the separation must occur at compile time. The Ubuntu g++ 13.3.0 compiler generated an error for the separation. I modified the code to use std::enable_if and provide a Set for each case, the other case ignored by SFINAE (Substitution Failure Is Not An Error).

GTE/Mathematics/GaussianElimination.h

The g++ compiler warned about the use of a pragma that is supported by the Microsoft Visual Studio compiler. With treat-warnings-as-errors, the files do not compile. I removed the pragma statements. The g++ compiler also complained about the Edge.T and Triangle.T members shadowing the class template parameter T. I renamed the members of Edge and Triangle to use lower-case v, e, and t.

GTE/Mathematics/MinimumVolumeBox3FloatingPoint.h
GTE/Mathematics/MinimumVolumeBox3Rational.h

The modifications to the minimum volume box code required an update to the sample application. The new template parameter list is different from the old one.

GTE/Samples/Geometrics/MinimumVolumeBox3D/MinimumVolumeBox3DWindow3.cpp

November 23, 2025. The GetExtreme function is a hill-climbing algorithm that visits the vertices of a convex hull. A user reported failure of the minimum volume box code, which I tracked down to failure of GetExtreme. I posted a fix on July 24, 2024. Another dataset showed this did not solve the original problem. I added another fix on August 31, 2024. This also did not solve the original problem. The final fix was to skip hill climbing and search all vertices, posted on September 5, 2024. Over 1 year later, another user of the code reported that the performance degraded significantly. The original datasets that showed failure had a very small number of points. The other user had datasets with thousands of points.

I spent quite a bit of time doing a deep dive of the code. It turns out the actual problem was that a convex hull can have triangular faces where some of the faces are coplanar (or nearly coplanar). Failure can occur when a vertex interior to a union of coplanar faces is visited. The algorithm terminated prematurely. My BSP-based extreme point code failed for the same reason. The minimum volume box code maintains a vertex-adjacency data structure built directly from the convex hull. The solution to the original problem was to remove edges from this data structure that are shared by two coplanar triangles. After such removal, there can be some vertices remaining that have only 2 neighbors. These also had to be removed. Effectively, the data structure represents a wireframe of the convex hull where some of the faces are convex polygons with 4 or more vertices.

I redesigned the class interface slightly, separating out the version that uses floating-point computations and the version that uses rational arithmetic. Both versions must use rational arithmetic to build the convex hull for an exact representation. Without this, the hill climbing can fail. Fortunately, the rational convex hull is not expensive to compute given that the dot products in GetExtreme are the main bottleneck in the computations.

GTE/GTMathematics.v17.{vcxproj,vcxproj.filters}
GTE/Mathematics/MinimumVolumeBox3.h
GTE/Mathematics/MinimumVolumeBox3FloatingPoint.h
GTE/Mathematics/MinimumVolumeBox3Rational.h

November 1, 2025. I had claimed previously that the ExtremalQuery3BSP code was broken. It turns out that the code works as before. The problem was that I had modified the ConvexHull3 code whereby the old code function GetUnorderedHull returned an array of triangles but the new code function GetHull returned an array of indices. The application code was not modified accordingly, so when the application was set to use the BSP query, it would trigger an assert. (I need the BSP query to fix a performance problem I introduced in a bug fix to MinimumVolumeBox3.) The query header was modified to avoid the Visual Studio code analysis warnings that the SphericalArc objects were possibly uninitialized, even though the default constructor initializes all member functions. The application now uses the BSP query by default.

GTE/Mathematics/ExtremalQuery3BSP.h
GTE/Samples/Geometrics/ExtremalQuery/ExtremalQueryWindow3.{h,cpp}

October 5, 2025. The function ComputeNearestOuterPolygonIntersection had an assertion that v0min and v1min were nonnegative. However, they are of type std::size_t, so the assertion is never triggered. This problem was introduced in version 6.0.2023.12.16 when I converted from type std::int32_t to std::size_t. Thanks to Aaron Barany for diagnosing and reporting the problem. I am working on modifications to trap more problems when the input outer and inner polygons are invalid.

GTE/Mathematics/TriangulateEC.h

September 22, 2025. Embarcadero 13 uses Clang 15, where std::is_signed<T> is now a compiler intrinsic rather than a template class. As such, a specialization of std::is_signed<T> is not allowed. I removed the specializations from the rational arithmetic classes.

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

August 20, 2025. Implemented the occlusion query for the OpenGL graphics engine.

GTE/Graphics/GL46/GL46Engine.{h,cpp}

August 8, 2025. The function OpenGLReportListener was incorrect regarding throwing exceptions. The throw occurred inside the default case of the switch statement. It had to be outside the switch statement and inside the code block for when code is not zero. After making this change, the Intel Graphics driver on my laptop threw an exception in the GLSL reflection code when checking for a GL error in glProgramInterfaceiv for programInterface is GL_TRANFORM_BUFFER. I have seen this behavior over the years. I thought that the OpenGL specification required this to be handled. I added a block of code to trap this special case and not call ReportGLError. I also commented out the preprocessor definition GTE_GL46_THROW_ON_REPORT_LISTENER_WARNING in case other such GL errors are unexpectedly generated.

GTE/Graphics/GL46/GL46.cpp

July 17, 2025. Modified the default version of OpenGL from `#version 430` to `#version 460`. Modified comments to reference OpenGL 4.6 instead of OpenGL 4.5.

GTE/Graphics/GL46/GL46Engine.h
GTE/Graphics/GL46/GL46ProgramFactory.{h,cpp}

July 14, 2025. Removed two unused class members.

GTE/Samples/Graphics/MultipleRenderTargets/MultipleRenderTargetsWindow3.h

June 26, 2025. Fixed a typographical error in a comment.

GTE/Mathematics/DistAlignedBoxAlignedBox.h

June 24, 2025. After removing support for MSVS 2019, some files in the GenerateProject tools were deleted. However, I had failed to remove the project references to those files, and the project failed to compile.

GTE/Tools/GenerateProject/GenerateProject.v17.vcxproj
GTE/Tools/GenerateProject/GenerateProject.v17.vcxproj.filters
GTE/Tools/GenerateProject/ProjectTemplate.cpp
GTE/Tools/GenerateProject/GenerateProject.cpp

June 8, 2025. For the OpenGL trace system, I added wrappers to hide the Microsoft OpenGL 1.0 and 1.1 DLL functions. This is a hack for now which uses the gte namespace. Fighting the gl* names for the DLL functions when the remaining extensions have wrappers around them with gl* names was painful. A later modification of GTE OpenGL will rename all the wrappers Gl* to avoid the hack.

GTE/Graphics/GL46/GL46.h
GTE/Graphics/GL46/GL46.cpp

3 Initial Distribution of Version 8.0

May 12, 2025. The main modifications are listed here.

- According to the Microsoft Product Fixed Lifecycle Policy, the Mainstream Support for Microsoft Visual Studio 2019 ended in April 2024. I have removed the solutions and projects for Microsoft Visual Studio 2019, now supporting only Microsoft Visual Studio 2022. My contract load is sufficiently large that reducing development and support time for my freely available GTE is necessary. If you need to build the code using Microsoft Visual Studio 2019, you can download the Community Edition and select the platform toolset to be v142 (MSVS 2019) instead of the default v143 (MSVS 2022).
- The GTE OpenGL graphics code supported only through OpenGL 4.5. The features for OpenGL 4.6 are extremely small, so I added support for them. Any names involving GL45 now use GL46 instead. This includes class names and file names. I downloaded the latest headers from Khronos (such as glcorearb.h) and use them instead of the older ones that I was using for OpenGL 4.5.

- I added the ability to generate traces for OpenGL function calls. You can enable this by adding to your OpenGL projects the preprocessor symbol GTE_ENABLE_GLTRACE. Alternatively, you can expose the define of this symbol in GTE46.cpp (line 139). When enabled, the trace is written to the file GLTrace.txt in the working directory (when running inside Visual Studio) or the executable directory (when running outside Visual Studio). I found the traces to be useful when rapid prototyping using GTE but then extracting the native OpenGL code for clients to use.