Geometric Tools Engine 7 Update History

Contents

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

- Version 7.2 posted November 14, 2024.
- Version 7.1 posted July 27, 2024.
- Version 7.0 posted January 7, 2024.

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 7.2

November 17, 2024. The include path to the parallelepiped class used quotations. To be consistent, the path was modified to use angle brackets.

GTE/Mathematics/DistPoint3Parallelepiped3.h

2 Updates to Version 7.1

November 14, 2024. Modified the ChangePlatformToolset tool to support selection of all compilers allowed by GTE.

GTE/Tools/ChangePlatformToolset/ChangePlatformToolset.cpp
GTE/Tools/ChangePlatformToolset/ChangePlatformToolset.v17.vcxproj
GTE/Tools/ChangePlatformToolset/ReadMe.txt

The Intel C++ Compiler 2025 complained about the static cast of the function pointer returned from wglGetProcAddress. I removed the static cast and used the already existing reinterpret_cast in GetOpenGLFunctionPointer from the return pointer to a void* followed by a reinterpret_cast to the appropriate function pointer type. The compiler also warns that GetOpenGLFunctionPointer can be made static, but that is incorrect. This function is implemented and exposed via extern in the WGL engine and in the GLX engine.

GTE/Graphics/GL45/WGL/WGLExtensions.cpp

November 13, 2024. The Intel C++ Compiler 2024 complained about std::wstring_convert being deprecated. GTE supports C++ 14, where the function is still valid. Intel supports C++ 17 where it is deprecated. However, the Intel compiler ignores the setting in the GTE projects that specifies to use C++ 14. Moreover, Intel ignores the settings that disable specific warnings in the GTE code that are activated by pragma statements.

GTE/Graphics/DX11/DX11.cpp

The Intel C++ Compiler complained about missing override modifiers.

GTE/Mathematics/Samples/Mathematics/ApproximateBezierCurvesByArcs/ApproximateBezierCurvesByArcsWindow2.h

The Microsoft Visual Studio 2022 (version 17.12.0) code analysis tool complained about not initializing mBits in the constructor initialization list. I had added a note to the file that initialization leads to a severe performance penalty (setting large numbers of array elements to 0). I added a pragma to disable the warning for the constructors.

GTE/Mathematics/UIntegerFP32.h

November 12, 2024. The Microsoft Visual Studio 2022 (version 17.11.5) code analysis tool complained that an index is out-of-range. It is not, so I added a pragma to disable the warning for that block of code.

GTE/Mathematics/IntrCircle2Arc2.h

November 11, 2024. The relationship between $\boldsymbol{B}$ and $\boldsymbol{T}$ needed an adjustment to compensate for the translation of the point sets by the average of one of the point sets. A 3D visualization shows that the alignment of point sets is working correctly.

GTE/Mathematics/HelmertTransformation7.h

November 9, 2024. Added an implementation of the 7-parameter Helmert transformation. There is a corresponding PDF at the documents page of the Geometric Tools website.

GTE/GTMathematics.{v16,v17}.{vcproj,vcxproj.filters}
GTE/Mathematics/HelmertTransformation7.h

November 4, 2024. Added hyperlinks to the online PDFs describing the algorithms.

GTE/Mathematics/DistPoint2Parallelogram2.h
GTE/Mathematics/DistPoint3Parallelepiped3.h

November 2, 2024. Added a distance and closest-point query for a 3D point and a 3D parallelepiped. A PDF was added to the documentation page at the Geometric Tools website (DistancePointParallelepiped.pdf). The 2D code was refactored to expose a function GetMinimizer that is used by both the point-parallelogram and point-parallelepiped queries.

GTE/GTMathematics.{v16,v17}.{vcproj,vcxproj.filters}
GTE/Mathematics/Parallelepiped3.h
GTE/Mathematics/DistPoint3Parallelepiped3.h
GTE/Mathematics/DistPoint2Parallelogram2.h

October 25, 2024. The line result.numIntersections = 0 was inside the last loop of the query but needed to be outside the loop. Thanks to github's *owai* for reporting this bug.

GTE/Mathematics/IntrCircle2Arc2.h

October 21, 2024. The inclusion of the header file for Parallelogram2.h was exposed via quotes but needed to be <Mathematics/Parallelogram2.h>. I also added includes of C++ Standard Library headers whose symbols are accessed in the distance query header.

GTE/Mathematics/DistPoint2Parallelogram2.h

Modified the order of vertices returned by GetVertices so that the ordering is consistent with that of OrientedBox2.

GTE/Mathematics/Parallelogram2.h

October 15, 2024. Added a distance and closest-point query for a 2D point and a 2D parallelogram. A PDF was added to the documentation page at the Geometric Tools website (DistancePointParallelogram.pdf).

GTE/GTMathematics.{v16,v17}.{vcproj,vcxproj.filters}
GTE/Mathematics/Parallelogram2.h
GTE/Mathematics/DistPoint2Parallelogram2.h

September 18, 2024. Added an implementation for computing the minimum spanning tree of a vertex-edge graph. The implementation also computes the back edges for the graph relative to the minimum spanning tree.

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

September 6, 2024. Added new samples for approximating 2D point sets by an ellipse and 3D point sets by an ellipsoid. These were backported from GTL end-to-end testing.

GTE/BuildAll.{v16,v17}.sln
GTE/Samples/Mathematics/CMakeLists.txt
GTE/Samples/Mathematics/ApproximateEllipse2/*
GTE/Samples/Mathematics/ApproximateEllipsoid3/*

Fixed typographical errors that mentioned incorrect Boolean values for the inputs useEllipseForInitialGuess and useEllipsoidForInitialGuess to the operator() functions.

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

September 5, 2024. One more time. The GetExtreme function still failed. The essence of the function is a hill-climbing algorithm using a queue. My poorly written queue-like code, and in particular the visited[] management, was still not correct. Thanks to github's AnsonInTheAstroworld for reporting the bug once again, and for providing a dataset for which the implementation failed. Although a modification was suggested using queues (with storage std::vector), I used the simplest algorithm, which is to iterate over all the vertices and compute the maximum height. I used Intel oneAPI VTune for profiling. It turns out that the bottleneck in computing are the memset calls in the constructors for mBits of UIntegerFP32<2561>. MinimumVolumeBox3 has a significantly large number of constructor calls; the initialization was approximately 80% of CPU time (the latter number on the order of 1 minute). I removed the initializations. On my Intel i9-10900 CPU which has 20 logical processors, the profiling showed that with a single-threaded execution (in the main thread), the queue-based approach used 4638 milliseconds for the provided dataset. My simple algorithm used 3121 milliseconds. Using 8 threads, the queue-based implementation used 1028 milliseconds and my simple algorithm used 722 milliseconds. The numbers are sufficiently small enough that the simple algorithm should suffice.

GTE/Mathematics/MinimumVolumeBox3.h
GTE/Mathematics/UIntegerFP32.h

August 31, 2024. The GetExtreme function still did not work correctly after the July 24, 2024 modification. The initial value of dMax for vertex mVertices[0] is computed before the loop over the vertices to find potentially larger dMax values. An example showed that the initial dMax is 0. The adjacent vertices to mVertices[0] all have zero-valued d-values. The adjacent vertices to mVertices[1] all have zero-valued d-values, and one of those vertices is mVertices[0]. This causes vMax to be assigned to 0 again, which invalidates the search for maximum dMax. I added another line before the loop that marks mVertices[0] as visited. Thanks to github's AnsonInTheAstroworld for reporting the bug and providing an example that caused failure.

GTE/Mathematics/MinimumVolumeBox3.h

August 26, 2024. Major rewrite of the DistanceCircle3.pdf file. Modified DistLine3Circle3.h accordingly. Added implementations of ray-circle distance queries (DistRay3Circle3.h) and segment-circle distance queries (DistSegment3Circle3.h).

GTE/GTMathematics.{v16,v17}.{vcproj,vcxproj.filters}
GTE/Mathematics/DistLine3Circle3.h
GTE/Mathematics/DistRay3Circle3.h
GTE/Mathematics/DistSegment3Circle3.h

Somehow this file slipped through the instantiation tests. The calls to ConvexHull2 still had the last parameter (an epsilon value for fuzzy arithmetic), but that should have been removed with the August 3, 2024 entry.

GTE/Mathematics/SeparatePoints2.h

The use of auto in IntpQuadraticNonuniform2 to obtain maximum indices and then also to initialize a loop variable worked with the Microsoft compiler but not g++ on Linux. Removed unused variables and types in the sample application.

GTE/Mathematics/IntpQuadraticNonuniform2.h
GTE/Samples/Mathematics/Interpolation2D/Interpolation2DWindow3.h

I forgot to increment the minor version from 0 to 1 in the CMakeLists.txt files.

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

August 3, 2024. Removed the obsolete mEpsilon member, which changes the signature to the operator() functions. Added an operator() function whose input is a std::vector<Vector2<T>>. Back-ported (most of) the GTL ConvexHull2 code to GTE. Fixed compiler breaks that resulted from the signature change. As is the case with other reworked GTE files, I replaced the template parameter Real with T as a signal that I have revisited the code (usually for GTL and back-porting to GTE).

GTE/Mathematics/ConvexHull2.h
GTE/Mathematics/ConvexHull3.h
GTE/Mathematics/MinimumAreaBox2.h
GTE/Mathematics/MinimumWidthPoints2.h
GTE/Samples/Geometrics/ConvexHull2D/ConvexHull2DWindow2.cpp

August 2, 2024. The convex hull is computed using rational arithmetic. The class IntrinsicsVector2 is first used to determine the intrinsic dimension of the input points, but these computations use floating-point arithmetic. The direction between the extreme points is obtained as a normalized difference of points. The floating-point rounding errors can lead to the incorrect conclusion that the hull is 2-dimensional, but the hull (using rational arithmetic) is actually 1-dimensional. The dataset that exposed the bug was passed to the minimum-area box code, and the misclassification of the intrinsic dimension leads to a crash caused by an out-of-range index into a std::vector. The dataset has more than 3 double-precision points that happened to be exactly on a line; rational arithmetic verified this. If exactly 3 double-precision and colinear points are passed to the hull finder, there is no crash. I removed IntrinsicsVector2 and set mDimension based on the rational computation of the hull. As it turns out, I had already done the same in the GTL code but for some reason did not back-port that code to GTE. Thanks to Richard Bouwman for reporting the bug.

GTE/Mathematics/ConvexHull2.h

3 Updates to Version 7.0

July 27, 2024. When drawing a 2D disk using ImageUtility2::DrawCircle with a Boolean input solid of true, for each x-value the callbacks are executed for a range of y-values. For a callback as simple as drawing only the pixel, this leads to many cache misses because of the row-major ordering for the image. The code has been modified to select y-values and execute the callbacks for a range of x-values.

GTE/Mathematics/ImageUtility2.h

July 24, 2024. The GetExtreme function was incorrect when all the adjacent vertices of the initial vertex have the same projection in the specified direction. Geometrically, the convex hull has a collection of coplanar triangles sharing the initial vertex. GetExtreme bails after examining the adjacent vertices and incorrectly concludes that the initial vertex is the extreme. The comparison dCandidate > dMax must instead be dCandidate >= dMax to allow the hill climbing to continue away from the flat region of the initial vertex. And it is now necessary to use visited flags to ensure the hill climbing does not oscillate between two vertices of the same projection value. The visited flag array must be local to GetExtreme because in the multithreaded case, a class member array is no longer protected from concurrent access. Thanks to github's AnsonInTheAstroworld for reporting the bug.

GTE/Mathematics/MinimumVolumeBox3.h

July 17, 2024. Unit testing of TriangulateCDT in the GTL development track exposed a subtle bug in BSNumber when GTE_BINARY_SCIENTIFIC_SHOW_DOUBLE is defined. The functions BSNumber::SetSign, BSNumber::SetBiasedExponent, and BSNumber::SetExponent each recompute mValue. The problem is that if the source object represents 0 and SetSign is called with a nonzero value, the target object is not yet a valid BSNumber because SetBiasedExponent and bit copies have not yet occurred. This leads to failure in the mValue recomputing of SetSign. The GTE version of TriangulateCDT does not fail because 0 is never the source of the incremental construction, whereas the GTL version has a case where the source is 0. GTE has several blocks of code that incrementally create a BSNumber object using SetSign, SetBiasedExponent, and a copy of the integer bits from the source object. I added updates of mValue after those blocks.

GTE/Mathematics/BSNumber.h
GTE/Mathematics/ConstrainedDelaunay2.h
GTE/Mathematics/Delaunay2.h
GTE/Mathematics/Delaunay3.h
GTE/Mathematics/IncrementalDelaunay2.h

July 15, 2024. Fixed bugs in the interpolator. The original Akima code was in Wild Magic 5, and it was ported correctly to GTE 1.0. The incorrectly repeated FX calls were introduced in GTE 2.2 (2/2/2016). The lack of parentheses for (bound-1) expressions was introduced in GTE 5.14 (11/11/2021). The Akima uniform 3D interpolator had been unit tested in Wild Magic, but I dropped the ball on testing it during the evolution of the GTE code. Thanks to Russel Wenzel for reporting the errors.

GTE/Mathematics/IntpAkimaUniform3.h

July 14, 2024. The merge of a pull request added a second pragma-once statement to ContTetrahedron3.h, which I removed. The pull request also fixed typographical errors in the comments, and a Python script was added to removed files from the GTE directory tree. The TrackObject class was given setters for x-size and y-size that are used if the application window is resized. I updated the file version numbers for these modifications.

GTE/Mathematics/BoxManager.h
GTE/Mathematics/ContTetrahedron3.h

GTE/Mathematics/IntrOrientedBox3OrientexBox3.h
GTE/Mathematics/OBBTree.h
GTE/Samples/Graphics/LightsWindow3.cpp
GTE/clean.py
GTE/Applications/TrackObject.h
GTE/Applications/Window3.cpp

June 24, 2024. Removed SetParallaxProjection based on the revised document PerspectiveMappings.pdf. Such a perspective projection matrix cannot be constructed when the view frustum viewport is a non-parallelogram convex quadrilateral.

GTE/Graphics/Camera.{h,cpp}

May 21, 2024. Added a new PDF (LieGroupsAlgebras.pdf) that describe the mathematics and implementation of LieGroupsAlgebras.h. The minimax approximations had to be adjusted for the cases when $|\theta| > \pi$. Also, the adjoint representations were those of Ethan Eade's document (mentioned in the bibliography of the PDF), but they were based on his choice for the order of the generators. My implementation uses a different order, which leads to adjoint representations that are permutations of his.

GTE/Mathematics/LieGroupsAlgebras.h

April 27, 2024. Added a new file for computing using Lie groups and Lie algebras. For now, the supported groups are SO(2), SE(2), SO(3), and SE(3). Minimax approximations are used for rotation-related functions to avoid the removable singularities at angle zero.

GTE/Mathematics/LieGroupsAlgebras.h
GTE/Mathematics/RotationEstimate.h
GTE/GTMathematics.{v16,v17}.{vcproj,vcsproj.filters}

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

GTE/Mathematics/NearestNeighborQuery.h

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

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

March 23, 2024. My template instantiation system needed serious modification for when I deprecated a class and provided a new implementation. The system failed to instantiate the new implementations. I

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

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

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

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

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

GTE/Mathematics/ApprParabola2.h

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

GTE/Mathematics/InscribedFixedAspectRectInQuad.h

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

GTE/Mathematics/InscribedFixedAspectRectInQuad.h

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

GTE/Samples/Imagics/ExtractRidges/ExtractRidgesConsole.cpp

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

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

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

GTE/Mathematics/Matrix.h

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

GTE/Mathematics/HermiteTriquintic.h

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

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

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

GTE/Samples/Imagics/ExtractRidges/ExtractRidgesConsole.cpp

January 7, 2024. New major version posted, no updates yet.