# Geometric Tools Engine Version 1.14
# Installation Manual and Release Notes

Document Version 1.14
June 7, 2015

## Contents

# 1 Introduction

You are about to install Geometric Tools Engine 1.14 (GTE). Version 1.0 of the source code is the companion to the book GPGPU Programming for Games and Science and was developed on Microsoft Windows 8.1 using Microsoft Visual Studio 2013 and Direct3D 11.1. We are now using C++ 11 constructs, so it is necessary to use version 2013 of the compiler. A brief summary of the version modifications is listed next. You can find the file revision logs at the Updates page of our website.

- Version 1.1 involved porting the code *not including the graphics or application code* to Linux and to Macintosh OS X.
- Version 1.2 includes a design change to the `Window` class regarding automatic updating of transformations.
- Version 1.3 includes some new classes for automatic texture coordinate generation.
- Version 1.4 includes a new class to support passing information about available hardware to algorithm implementations. Two of the Wild Magic physics samples were ported, Rope and Cloth, which required porting mesh generation for tube surfaces and rectangle surfaces. Two folders had names starting with a hash symbol (#), which is not friendly to Linux because you have to escape the symbol. These folders were merged with those of the same name but without the hash symbol. The Numerics samples were merged into the Mathematics samples.
- Version 1.5 includes a port of the Wild Magic polygon triangulation by ear clipping, and it includes new code for polygon triangulation using constrained Delaunay triangulation. Sample applications are provided for both classes. You can now use DirectX 10.x feature levels. Ported the Wild Magic class for generating sets of unique vertices and triangles from triangle soup or sets of indexed triangles. Revised the MultipleRenderTargets sample to be clearer about what it illustrates. Ported the Wild Magic class for min-heaps to support priority queues with additional semantics useful for geometric algorithms. Various bug fixes were made.
- Version 1.6 includes an overhaul of the arbitrary precision library to improve its performance and readability, and a new PDF describing the library in great detail is available. It includes a robust implementation of distance between line segments. And it includes some minor fixes and feature additions.
- Version 1.7 removes classes Vector2, Vector3, Vector4, Matrix2x2, Matrix3x3, and Matrix4x4. These classes existed solely to provide constructors for dimensions 2, 3, and 4. The base classes now have constructors to support initialization of specific dimension objects. Moreove, constructors were added that use std::initializer_list. You will need to change constructor calls to use brace syntax.

  Although the derived classes were removed, we use *template aliases* to define those symbols, so you can continue to use, for example, Vector2<float>. With these changes, the mathematics code in GTEngine now requires C++ 11. You must use Microsoft Visual Studio 2013; MSVS 2010 and MSVS 2012 do not have support for template aliases. The decision to change the design allows us to consolidate a lot of code, now including the dimension as a template parameter in various geometric objects and queries. Thus, we have many fewer files that effectively have duplicate logic, which in turn reduces the amount of unit testing code we must write. The consolidation will be an ongoing process, so expect more code changes to the geometric queries.

The macro GTE_MAKE_HLSL_STRING was removed and the built-in effects now have quoted strings for the shaders. The macro does not work on Linux or Macintosh (g++ and LLVM compilers), and there were some adverse interactions between macro expansion and the C preprocessor rules. After these changes and a minor fix, we have ensured that all sample applications work correctly for either choice of multiplication convention (GTE_USE_MAT_VEC or GTE_USE_VEC_MAT) and for either choice of matrix storage (GTE_USE_ROW_MAJOR or GTE_USE_COL_MAJOR).

- Version 1.8 moves the inline bodies in the *.inl files to the *.h header files. This greatly reduces the number of files we must maintain, both in source control and for website links. Converted more dimension-specific classes to include the dimension as a template parameter (Rectangle, Cone, Capsule). Continued with query changes based on the new template classes, including adding more template aliases. Added test-intersection query for 3D boxes and cones.
- Version 1.9 has several bug fixes and document updates. A robust point-triangle distance query was added to the engine and the previous code is now used for exact rational arithmetic.
- Version 1.10 has several bug fixes related to classification of the rotation or reflection matrices in the eigensystem solvers. A new tool was added, HLSLToByteCode, that embeds HLSL bytecode in a C++ source file; this allows one not to embed human-readable HLSL strings in the executable. The rotating calipers algorithm was revised in the minimum-area rectangle and minimum-volume box code. The box code now executes completely with exact rational arithmetic by avoiding the non-exact projection of the supporting polylines to the plane of a hull face. The loop invariants of the rotating calipers algorithm are now maintained during the rectangle and box updates. Multithreading support was added to ConvexHull3 and MinimumVolumeBox3; the sample applications were updated accordingly.
- Version 1.11 has updates to documents with associated code changes. A type traits system was added to support compilation of functions based on computation type (floating-point, rational arithmetic). Modified the low-degree polynomial document and code to use rational arithmetic for root classification in order to support robust root finders.
- Version 1.12 fixes a compiler error using Xcode 6.3 and Apple LLVM 6.1.0 compiler. Updated the ellipse-ellipse queries for test-intersection and find-intersection. Implemented the query for area of intersection of ellipses (a very popular document for download). The corresponding PDF documentation was significantly updated.
- Version 1.13 is a roll-up of some general code maintenance, revised PDF documents, and code updates to match the PDF documents.
- Version 1.14 fixes compiler warnings with Microsoft Visual Studio 2015 RC. We included some bug fixes to several intersection queries.

The graphics and application code for Linux and Macintosh platforms are in development, mainly requiring an OpenGL 4.x-based graphics engine.

You should visit our website regularly for updates, bug fixes, known problems, new features, and other materials. The update history page always has a date for the last modification, so you should be able to track what you have or have not downloaded. The source files themselves are labeled with a version number of the form *1.minor.revision*, where *minor* is the minor version in which the file shipped and where *revision* is the number of times the file has been revised. The source files in the initial distribution of Geometric Tools Engine 1.0 are labeled with 1.0.0.

## 2 License

The Geometric Tools Engine uses the Boost License. The license in its entirety is listed next.

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization
obtaining a copy of the software and accompanying documentation covered by
this license (the Software) to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:

The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED ''AS IS'', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

## 3 Copying the Distribution to Your Machine

You may unzip the distribution to a folder of your choice. The top-level folder of the distribution is `GeometricTools` and the subfolder for the distribution is named `GTEngine`. The projects all use paths relative to `GTEngine` and they do not rely on the top-level directory being located at the root of a hard drive.

Some of the folder hierarchy is shown next. The `Include` and `Source` folders contain all the code for the engine. We chose to omit the post-build copy of header and inline files to an `SDK` folder (as done in Wild Magic). If you have projects outside the `GTEngine` hierarchy, you can add a search path in a Microsoft Visual Studio project for the header and inline files using `$(GTE_PATH)/Include`. We use Microsoft Visual Studio's reference system in the project settings for the applications to find the libraries to link to.

```
GeometricTools
  GTEngine                  // Root folder for Geometric Tools Engine, set GTE_PATH to here.
    Include                 // Location for *.h, *.inl, *.hlsl, and *.hlsli files.
    Source                  // Location for *.cpp files.
    Samples                 // Sample applications, many discussed in the GPGPU book.
      Data                  // A small number of data files for the samples.
      Basics                // Basic tutorials for several HLSL concepts.
      Geometrics            // Samples for computational geometry.
      Graphics              // Samples for graphics and video streams (parallel copy).
      Imagics               // Samples for 2D and 3D image processing.
      Mathematics           // Samples for mathematical algorithms and numerical methods.
```

```
  Physics                 // Samples for 2D and 3D physics.
Shaders                   // HLSL shader files (embedded versions are in the engine source).
Tools                     // Several convenient tools.
  BitmapFontCreator       // Generate .h/.cpp file to represent a graphics font.
  GenerateApproximations  // Used to generate the minimax approximations for common functions.
  GenerateOpenGLWrapper   // Source-code generator for gl* wrappers drive by glcorearb.h.
  GenerateProject         // Generate a skeleton (vcxproj, sln, h, cpp) for GTE applications.
  HLSLToByteCode          // Embed HLSL byte code in a C++ source file.
```

The `Samples` subfolders are many. Listing them here would make the displayed hierarchy difficult to read.

# 4 Environment Variables

## 4.1 Microsoft Windows

Data files are found using an environment variable that you must create; its name must be `GTE_PATH` and its value must be the path to where you installed `GTEngine`. For example, if your user account is in the default location and you installed the `GeometricTools` folder to your user account folder, you will create an environment variable

```
GTE_PATH = C:\Users\YourAccount\GeometricTools\GTEngine
```

To create an environment variable, you need to launch the System window that is accessible through the Control Panel. The simplest way to do this is via the Windows shortcut: `Windows Key + Pause/Break`. Figure 1 shows the top-half of the window.

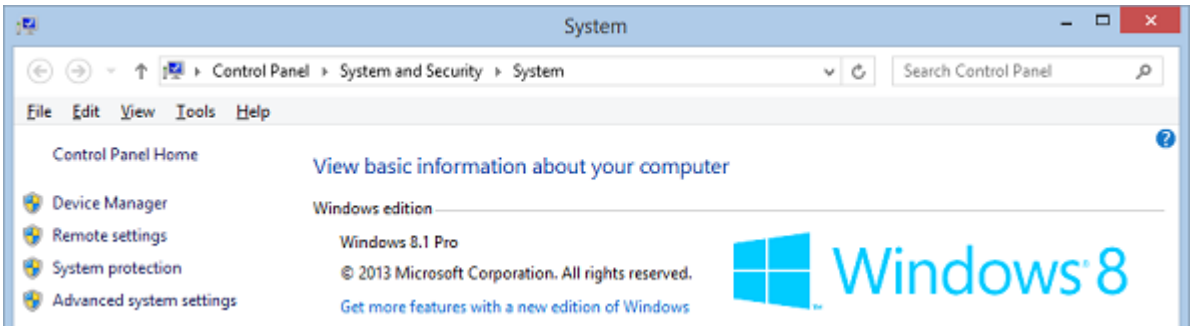


Figure 1. The System dialog window.

Select `Advanced system settings` to launch the dialog shown in Figure 2(a).

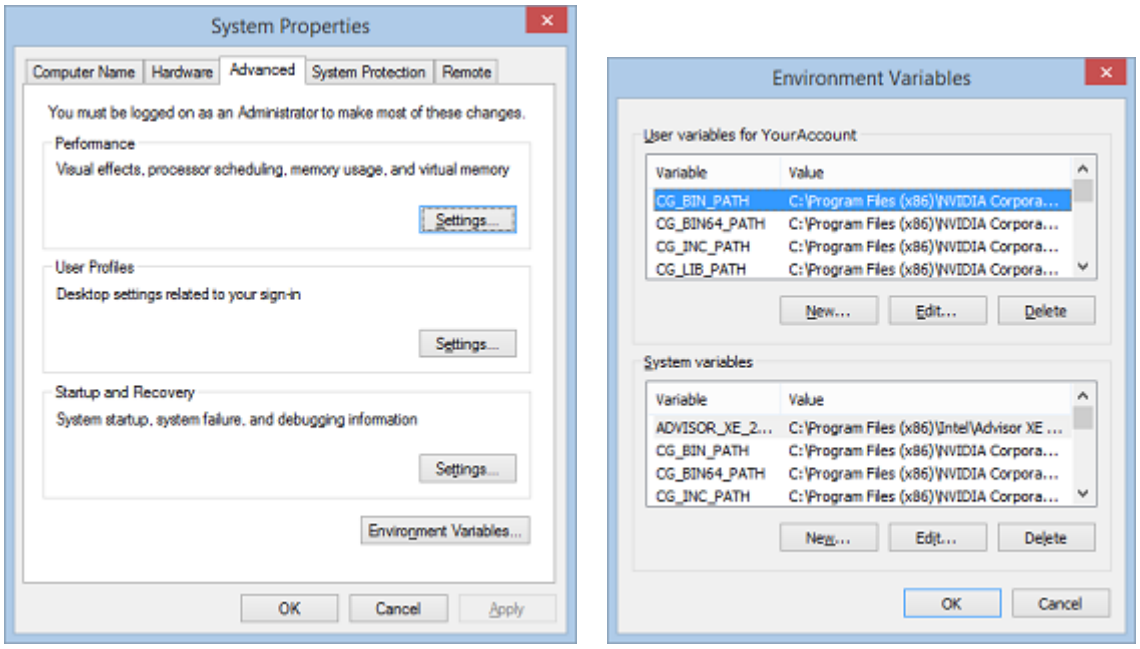


Figure 2. (a) The Advanced System Settings dialog window. (b) The Environment Variables dialog window.

Select the `Enviornment Variables ...` button to launch the dialog shown in Figure 2(b). Under `User variables for YourAccount`, select the `New ...` button to launch the dialog shown in Figure 3.

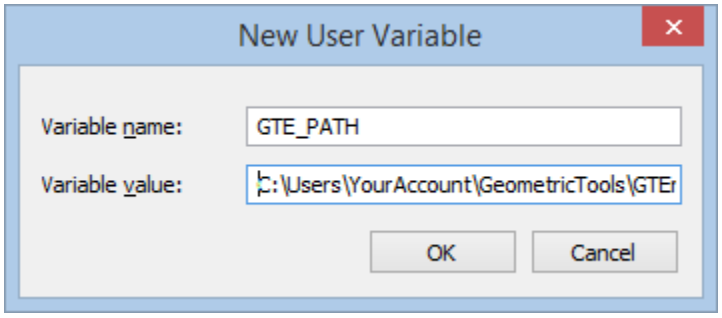


Figure 3. The New User Variable dialog window.

Enter the name and path as shown in the figure. Naturally, you will use the path to your installation. The dialog cannot be resize, so the environment variable value is not fully displayed. Press the `OK` button.

## 4.2 Linux

Until we port the application source code, there is no need for an environment variable named `GTE_PATH`. For now the Linux distribution is effectively a large mathematics library.

### 4.3 Macintosh OS X

Until we port the application source code, there is no need for an environment variable named `GTE_PATH`. For now the Macintosh OS X distribution is effectively a large mathematics library.

## 5 Compiling the Source Code

### 5.1 Microsoft Windows

The engine solution is `GeometricTools/GTEngine/GTEngine.sln`. Each sample application or tool has its own solution with all dependencies listed, so it is possible to open a sample application and compile and run it without explicitly building the engine solution first. The folder `GTEngine` contains the solution `GTBuildAll.sln` if you want to build the engine, samples, and tools rather than building the projects separately.

Currently, the engine solution generates static libraries. We have the hooks in place for dynamic libraries but have not yet created the build configurations. We will provide the dynamic library configurations eventually.

### 5.2 Linux

The makefile to build the GTEngine library is

```
GeometricTools/GTEngine/makefile.gte
```

Both static and shared library builds are supported. From a terminal window execute

```
make CFG=configuration -f makefile.gte
```

where `configuration` is `Debug` or `Release` for static libraries or is `DebugDynamic` or `ReleaseDynamic` for shared libraries.

### 5.3 Macintosh OS X

We have provided an Xcode project

```
GeometricTools/GTEngine/GTEngine.xcodeproj
```

that allows you to build any of four configurations. The `Debug` and `Release` configurations generate static libraries. The `Debug Dynamic` and `Release Dynamic` configurations generate shared libraries.

# 6 Running the Samples (Windows 8.x and Windows 7)

You can run the samples from within the Microsoft Visual Studio development environment. Samples that access data files use the `GTE_PATH` environment variable to locate those files; code is in place to assert when the environment variable is not set. If you run from Microsoft Windows, presumably double-clicking an executable via Windows Explorer, the environment variable is still necessary.

Many of the samples compile HLSL shaders at run time. This requires having `D3Dcompiler_47.dll` in your path. When running through Microsoft Visual Studio, the environment has a path to

```
C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin
```

that stores a copy of the DLL. It also has macros

```
$(WindowsSDK_ExecutablePath_x86), $(WindowsSDK_ExecutablePath_x65)
```

that contain paths to the Windows Kits that are part of Windows 8.x. These paths are

```
C:\Program Files (x86)\Windows Kits\8.x\bin\x86
C:\Program Files (x86)\Windows Kits\8.x\bin\x64
```

where `x` is 0 or 1, depending on which version of the operating system is installed. Each has a D3D compiler DLL. Windows Kits are not installed by default on Windows 7, but they are created when you install Microsoft Visual Studio 2013.

When running from Windows Explorer or a command window, you need to have one of these DLL locations as part of your path; otherwise, you get an error dialog stating the DLL cannot be found. On our 64-bit machines, we have added to the system `PATH` environment variable the Windows Kits location for the `x64` binaries.