Geometric Tools Engine Version 2.3
Installation Manual and Release Notes

Document Version 2.3
March 10, 2016

Contents

1 Introduction **2**

2 License **2**

3 Copying the Distribution to Your Machine **3**

4 Compiling the Source Code **4**

4.1 Microsoft Windows 4

4.2 Linux 4

4.3 Macintosh OS X 4

4.4 OpenGL 4.3 Support for Microsoft Windows 4

5 Running the Samples (Windows 7, 8, 10) **5**

1 Introduction

You are about to install Geometric Tools Engine 2.3 (GTE). Version 1.0 source code was the companion to the book GPGPU Programming for Games and Science and was developed on Microsoft Windows 8.1 using Microsoft Visual Studio 2013, C++ 11, and Direct3D 11.1. Version 2 is close enough to what is in the book that you should have no problem navigating the code as you read the book. The source code has been reorganized using subfolders of `Source` and `Include`, and the header includes are now slightly different. We have provided projects for both MSVS 2013 and MSVS 2015. Version 2.1 now has a graphics engine that requires minimally OpenGL 4.3 in order to support compute shaders. Version 2.3 removes all the raw calls to new and delete.

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 *2.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.

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. Create an environment variable named `GTE_PATH` that stores the absolute directory path to the folder `GeometricTools/GTEngine`. For example, if you unzipped the distribution to the root of the C drive, you would set `GTE_PATH` to `C:/GeometricTools/GTEngine`.

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 files.
      Applications              // Support for Microsoft Windows applications.
      Graphics                  // Platform-independent graphics files are in this folder.
        DX11                    // DX11-specific graphics files are in this subfolder.
        GL4                     // OpenGL-specific graphics files are in this subfolder.
      Imagics                   // Image processing files.
      LowLevel                  // Several low-level utility files.
      Mathematics               // The bulk of the engine consists of mathematics support.
        Algebra
        Approximation
        Arithmetic
        ComputationalGeometry
        Containment
        CurvesSurfacesVolumes
        Distance
        Functions
        GeometricPrimitives
        Interpolation
        Intersection
        NumericalMethods
        Projection
        SIMD
      Physics                   // Some physics support, not all WM5 physics code has been ported.
    Source                      // Location for *.cpp files.
      Applications              // Support for Microsoft Windows applications.
      Graphics                  // Platform-independent graphics files are in this folder.
        DX11                    // DX11-specific graphics files are in this subfolder.
        GL4                     // OpenGL-specific graphics files are in this subfolder.
      Imagics                   // Image processing files.
      LowLevel                  // Several low-level utility files.
      Mathematics               // The bulk of the engine consists of mathematics support.
      Physics                   // Some physics support, not all WM5 physics code has been ported.
    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/GLSL 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 VS2013/2015 vcxproj, sln, h, and cpp for GTE applications.
```

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

4 Compiling the Source Code

4.1 Microsoft Windows

Microsoft Visual Studion 2013 uses Version 12 of the compiler and Microsoft Visual Studio 2015 uses Version 14 of the compiler. The project and solution names have embedded in them v12 or v14; that is, we support both versions of the compiler. The engine solutions are `GeometricTools/GTEngine/GTEngine.v12.sln` and `GeometricTools/GTEngine/GTEngine.v14.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 solutions `GTBuildAll.v12.sln` and `GTBuildAll.v14.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.

4.2 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.

4.3 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.

4.4 OpenGL 4.3 Support for Microsoft Windows

GTEngine now contains an OpenGL graphics system (GL4Engine) that requires at least OpenGL 4.3. The application layer can use this on Microsoft Windows. The port to Linux and Macintosh will occur next. All the Visual Studio projects have configurations Debug and Release that include compiling the Direct3D 11

source code. New configurations have been added, DebugGL4 and ReleaseGL4, that additionally compile the OpenGL source code. **NOTE:** In the new configurations, both graphics systems are enabled. It is possible to use DX11Engine and GL4Engine in the *same* application and with the *same* window. However, the intent is to support applications that use one system for GPU rendering (for example, GL4Engine) and the other system for GPGPU computing (for example, DX11Engine).

The GenerateProject tool now creates projects with Debug and Release configurations (Direct3D 11 only) and DebugGL and ReleaseGL4 (OpenGL 4.3 but Direct3D 11 still available). If you create your own projects, you must define the preprocessor symbol GTE_DEV_OPENGL and add opengl32.lib to your library link options.

The OpenGL functions are exposed via the files GteOpenGL.h and GteOpenGL.cpp. The cpp file is automatically generated from OpenGL header files using the tool

GeometricTools/GTEngine/Tools/GenerateOpenGLWrapper

Whenever glcorearb.h is updated by the Khronos Group, this file can be downloaded and GteOpenGL.cpp can be regenerated so that it exposes any new features.

Because the GTEngine OpenGL graphics system relies on the OpenGL headers, they must be accessible to the Visual Studio projects in some standard header file path. The default settings of the projects are to target Windows 8.1, even though the code runs on Windows 10. The standard path we use is

C:/Program Files (x86)/Windows Kits/8.1/Include/um/gl

and the files copied to this path are glcorearb.h, glext.h, glxext.h, and wglext.h. The GenerateOpenGLWrapper project has these files in the project folder; you should copy the header files to the aforementioned gl folder.

5 Running the Samples (Windows 7, 8, 10)

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_*.dll` in your path, where `*` is the version number of the shader compiler. You might have to modify your `PATH` environment variable to include the path. With latest Windows, the DLL should be in a Windows Kit bin folder.