Development Notes

David Eberly, Geometric Tools
Last Modified: October 25, 2020

Contents

This document is a brief summary of development towards getting the Geometric Tools Library (GTL) ported from Geometric Tools Engine (GTE).

1 Introduction

This is the master document for topics I think about that need to be added, deleted or modified in the Geometric Tools Library development track. The sections are an aid in organizing the items for readability. This is preferable to having one long list that has to be scanned visually to find items that perhaps require only minor effort.

2 General

- Switch to using size_t for index types. This has been the single most annoying thing about porting GTE to GTL. I had used int for indexing Vector and Matrix objects, but Microsoft Visual Studio's code analysis complains a lot about 32-bit arithmetic used when generating a 64-bit index. One of the consequences of "size_t correctness" is having to add frequently static casts of integer types.
- Try to be consistent about use of auto. Sometimes for a pointer type I have auto and sometimes auto* or auto const*.
- Use static_cast<T>(var) rather than (T)var.
- Make sure override key word is used in derived classes when appropriate.
- Switch to using the types such as uint32_t rather than unsigned int.
- Replace T variable[N] by std::array<T,N> variable.
- Look at header include statements and determine whether any can be removed. For example, if Math.h and Vector.h are included, this is redundant because Vector.h includes Math.h. On the other hand, when including headers for the C++ Standard Library, sometimes one implementation will indirectly include a header file the GTL code needs, but other implementations do not—in which case I receive bug reports about code not compiling on a Linux distribution not part of my test suite. To avoid the bug reports, I should explicitly in a header file any standard library headers that are used by that header file.
- Some classes have a Boolean member that says whether or not the object has been constructed. When exceptions are thrown, these members are no longer needed. Remove them.
- Search for statements such as &myvector[0] and replace by myvector.data().

3 Projects, Solutions and Top-Level Items

- Build a generic tool to generate the projects and solutions from a file that lists all the GTEngine folders, cpp, h, hlsl, glsl and other files?

3.1 Applications

- Write a PDF about the C# to CLI to C++ solution and projects.
- Replace conditional compilation for testing different configurations in the samples with key-stroke based selection. This makes it easier to test because you do not have to modify a file and recompile.
- Move SetEnvironment to the Application base class and allow samples to provide only the names of the input files. That is, refactor out the common logic in samples into the base class.

3.2 LowLevel

- Try to eliminate slow std::array and std::vector operator[] lookups in Debug builds.

3.3 Graphics

- Add support for DX12 and Vulkan.
- Add the ability to have multiple stream inputs to vertex shaders. The DX11 version is already in the LowLevelStream, which is in GTEngine but was not ported to GTE. I need to figure out how to do this with OpenGL and then expose a common interface.
- Modify the GLSL compile system to preprend the shader inputs/outputs automatically to the shader strings. This needs to mimic what DX11 does with semantics so that the layout numbers are not hard-coded in the GLSL files.
- Revisit the LightingEffect. It is awkward to use, especially when you need the application code to hang onto the light world positions and/or directions.
- GLSL does not have an include-file system when compiling shaders. Add this support manually to the GL45Engine system.
- OpenGL is not thread-safe. Add thread-safe resource construction.
- Use consistent data type for color. Currently, I use std::array in some parts of the engine and gte::Vector4 in other parts. Introduce a color type?
- Finish adding picking support for the currently unsupported index topologies.
- Use auto for creating graphics objects with std::make_shared. Use vertices instead of vertex for read/write of VertexBuffer objects.
- When creating convex mesh factory objects with transparency, try to draw them so that the alpha blending works correctly. Add effects for order-independent alpha (disable depth buffer writing, store all depths for a pixel written multiple times, sort to get minimum depth, enable depth buffer writing, final draw to screen).
- Build an application framework for visualizing distance between objects and intersection between objects. Derive from it for the specific pairs of object types.
- IndexBuffer constructor has a size_t input (last value) that is used to select 16-bit or 32-bit sizes. Replace this input by an enumeration that includes: unindexed, 16-bit, 32-bit.

- Redesign the pvw-matrix update system to make it less verbose and/or to hide the buffer-updates, perhaps by registering an entire update functor?
- In the Resource class, use std::vector and add a ReplaceStorage call to use move semantics rather than copying the data via raw pointers.
- Try to factor the parallax projection matrix to uncouple the view matrix portion from the projective portion.
- The engine now requires OpenGL 4.5. Note that glGenerateTextureMipamap is provided in OpenGL 4.5, so the OpenGL engine can be modified to avoid the manual handling of mipmap generation.
- On Windows machines, the OpenGL samples appear t show a black-textured object for one frame before the correctly renderer object. Figure out why and fix.
- Consider renaming *.hlsl to *.dxsl to avoid the problems MSVS has when adding HLSL files to a folder and disabling HLSL compilation. It takes MSVS two modifications of the project/filter files to get this right.

3.4 Mathematics

- The TriangulateCDT code has been modified and some bugs fixed. The class is complicated enough that a PDF needs to be written to describe its design and use. This will be similar to the PDF for TriangulateEC.
- TriangulateCDT handles coincident vertex-edge and coincident edge-edge configurations. It does not handle edge-edge intersections at edge-interior points. Such configurations are a problem, because the two edges that intersect are both inserted into the constrained Delaunay triangulation, but each one interferes with the other's local retriangulation.
- Modify GenerateUVMesh to allow user to select the boundary uv rather than automatically generating them.
- The setting of BasisFunctionInput members needs to be simplified and/or explained better.
- Need representation of $\pm\infty$ BSNumber and BSRational? Or should this be the responsiblity of applications to handle when such cases can occur in those applications.
- Finish the mesh factory document and post once the meshing subsystem is complete; that is, fully implement and test MeshFactory. (GTL code is currently being written for this.)
- Remove the dependency of Polygon2 on IntrSegment2Segment2. The test for simple should be separate from the primitive definition.
- If I recall, the LCPSolver automatically includes all the epsilon-perturbations and applies arithmetic operations to them. This is slow for rational arithmetic. Redesign the implementation so that an epsilon-perturbation is added only when needed. This should improved performance significantly.
- The LCP solver can use QFNumber objects passed to functions. However, I had to hack the code to pass a lambda function that allows creating quadratic-field numbers with the appropriate d-value. The implementations need to be split up (floating-point, BSNumber, QFNumber).

- Replace the distance and intersection queries operator() by ExactQuery and RobustQuery. That is, there is probably now a need to make some explicit queries when the float and rational versions required significantly different implementations.
- Can ETManifoldMesh and ETNonmanifoldMesh share some code that is put in a base class ETMesh?
- Try not to derive classes in the DCPQuery, TIQuery and FIQuery implementations. The main technical problem is how to access the protected DoQuery calls. Use friend status?
- The test-intersection queries for boxes (aligned and/or oriented) have a Result object whose constructor has an epsilon input. The user cannot currently specify the epsilon via the operator() functions.
- Add erode, open and close operations to ImageUtility3.
- Move the PrimalQuery* files to Arithmetic.

4 Samples

4.1 Basics

4.1.1 ShaderReflection

- Have the sample look up shaders in the Shaders folder? [April 30, 2019][]

4.2 Imagics

4.2.1 VideoStream

- Modify the sample to use thread-safe queue and condition variables. Eliminate polling that consumes a lot of time in the thread. (low priority)