CMSC 335
COMPUTER GRAPHICS

LAB 1

• LIBRARY OVERVIEW
• PROGRAM 1 STARTER PROGRAM
• DESIGN RECOMMENDATIONS
PROGRAMMING LANGUAGE CHOICE – C++

• Modern choice for efficiency! Mainly because it is compiled and not interpreted.

• Please use modern features (C++11, 14, 17) to make programming easier

• Historically in graphics
  • Assembly – until 90s
  • C - 90s – 2000s
  • C++ starting in 2000s
• OpenGL and DirectX are graphics libraries that allow 3D rendering

• We will use OpenGL because it is open source and cross-platform
  • Open GL is a C library! Style is a bit archaic, but it plugs in nicely to C++
  • Open GL is only the graphics component and does not include a windowing system
  • Shading language is called GLSL (GL Shading Language)
OPEN GL INFORMATION

• All functions named as gl*, e.g., glClear(*), because C does not have namespaces

• All constants/enums named as GL_* , e.g., GL_TRUE

• GL type aliases are GL*, e.g., GLint
  • I recommend a "when in Rome" approach

• Function names commonly end with character/number code, e.g., glUniform3f(*). This describes the number and types of parameters, because C does not allow overloading
• GL Utility Toolkit (GLUT) – Windowing system that is deprecated
• Graphics Library FrameWork (GLFW) – Alternative windowing system
• GL Extension Wrangler (GLEW) – OpenGL 4 method wrappers
• GL Mathematics (GLM) – Linear algebra support
• Simple OpenGL Image Library (SOIL2) – Importing images for texturing
• Open Asset Import Library (Assimp) – Import 3D models

• Honestly, feel free to use whatever, e.g., Qt for windowing, or write your own, e.g., Mathematics
In project 1, you will create a simple software based renderer using ray tracing. We will cover this algorithm later this week.

However, I provided a program to show a framebuffer to help you get started.

Let's go over it now, find it on the course website.
RECOMMENDATIONS

• Use test-driven-development
• Be very careful in your design
  • Start early
  • Refactor early and often
  • I will lightly grade design in programming assignments and provide points for refactoring
• Use an external input format (XML or JSON) for program input
  • Never hardcode, try to make things parameters
• Program with efficiency in mind, i.e., place timers in your code to find bottlenecks
• OpenGL contains error handling methods, i.e., use them!
• Turn-in programming assignments using GitHub