What is 672?

- A general programming-based introduction to computer graphics using the most common general-purpose, high-level, platform-independent graphics API: OpenGL
- We will use OpenGL as a vehicle to study:
  - Common graphics algorithms, mathematics, and data structures
  - The MVC design pattern
  - CPU-GPU communication and cooperation

672 is Not:

- Graphic design (e.g., Alias, Maya, 3ds Max, LightWave, et al.)
  - Most of these programs are built using OpenGL
- Unity or similar special-purpose APIs
  - OpenGL is general-purpose, high-level, & platform-independent
  - Special-purpose APIs include:
    - Unity (popular for character animation & games)
    - VTK, NASA World Wind (used for scientific visualization)
  - These special-purpose APIs are typically constructed on top of OpenGL (or can be bound at runtime to any one of several).

Special Purpose APIs

- May hide all (or nearly all) aspects of OpenGL, e.g.:
  - Unity, VTK, Ogre
- May just wrap window and event handling in a platform-independent fashion, e.g.:
  - NASA World Wind
- Common and easier to use, but…

Limits on Special Purpose APIs

- Latest OpenGL features may not be supported (or they may be supported in an awkward way).
- Performance, especially for large models.
- The better you understand the OpenGL/GLSL model, the better you will be able to use the special purpose APIs.
- As a result, many graphics-related organizations seek people with “advanced knowledge of OpenGL & GLSL”. Some recent examples…

Sampling of Companies Seeking OpenGL Expertise

- NREL (National Renewable Energy Lab)
  - Utilize advanced immersive environments like CAVEs.
- “nearly all of our work is C++ and OpenGL.”
- LucasFilm (“expert level knowledge”)
- ESRI (huge player in GIS)
- VectorWorks (east coast graphics development company)
- Oblong Industries (interactive visualization systems)
- …
Prerequisites

- EECS 448 (Software Engineering)
- Well-developed programming-documentation-debugging skills
- Ability to read and understand code
- Especially important: OO concepts as implemented in C++
- Linear Algebra
  - We will review, but it’s helpful if you have previously studied:
    - Vector algebra (including +, -, *, dot & cross product)
    - Matrix algebra (including Matrix*Matrix, Matrix*vector)
    - Vector spaces
    - The cryph toolkit implements these and other operations we will need.

Applications

- Interactive Design
- Interactive Games and Simulations
- Interactive Analysis and Visualization of Data
  - Multidimensional
  - Multivariate
  - Time-varying
  - Massive and Distributed

Shader-Based OpenGL

- OpenGL 2.1 versus modern OpenGL (≥3.3; 4.x)
- Modern uses a cooperative CPU-GPU programming model in which generic data are sent from the CPU to the GPU. You write GPU code in GLSL (OpenGL Shading Language) to actually render the data.
- This explicit GLSL programming of GPU enables:
  - Instant ability to develop novel rendering techniques
  - Ability to handle very large data sets with application-dependent attributes at refresh rates
  - Let’s look at one 2D and one 3D example...

Legacy Applications

- Countless legacy academic, commercial, hobbyist, and other OpenGL-based applications exist.
- Hence most vendors continue to support OpenGL 2.1 and earlier applications, for example on request when the Rendering Context is created.
- You may encounter legacy code in web searches.
- All new development should target only new features.
- Using only new features will be a requirement of all projects done for this course.

Our Goals

- Learn modern Shader-Based OpenGL programming
  - Focus: Desktop OpenGL (currently OpenGL 4.x)
  - But we’ll see a bit of: OpenGL ES 2 and WebGL
  - AB 3: shader based & generic vertex attribute array based (although – as of FALL 2016 – WebGL supports only a very old version of GLSL)
  - Learn to write useful shaders using GLSL that run on GPUs
  - Master the mathematics of graphics
    - Points and Vectors in Affine and Projective Spaces
    - Role of Matrices in coordinate system transformations
  - Master the very common MVC graphics program software architecture
Class Attendance

+ Important!
+ Web material & printed references usually focus on:
  + How things work.
+ In class, we talk about “how”, but also:
  + Why we do things one way or another
  + Strategy when developing models and interaction techniques
  + Detailed explanations of our development framework, including
    + How to use it
    + How to extend it

Class Attendance (cont’d)

+ Both projects and exams allow you to practice all of this (how, why, strategy, and extensions).
+ Projects are very applied; exams tend to be more conceptual
+ Experience has shown that you are not likely to do well on exams if you don’t attend class regularly.
+ Finally: Note that the style of programming you will see here is likely different from anything you have seen or done to date.

References

+ General Graphics Text Based on OpenGL
+ Classical Addison-Wesley OpenGL Reference Books
+ OpenGL Reference Web Sites
  + PRIMARY: http://www.opengl.org
  + Window System Interfaces
    + http://www.glfw.org (what we will use)
    + http://freeglut.sourceforge.net (somewhat outdated alternative)