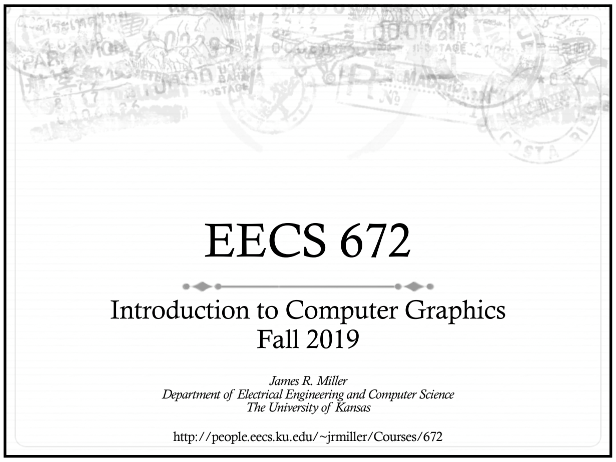
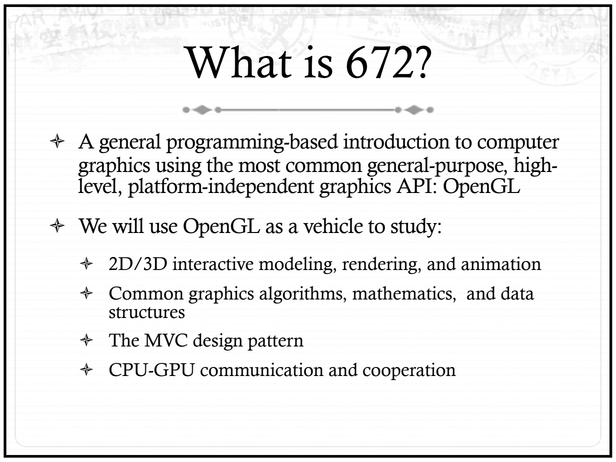


EECS 672

Introduction to Computer Graphics
Fall 2019

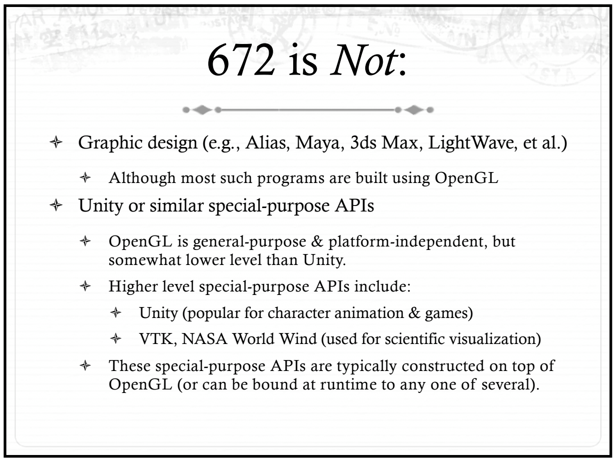
James R. Miller
Department of Electrical Engineering and Computer Science
The University of Kansas

http://people.eecs.ku.edu/~jrmiller/Courses/672



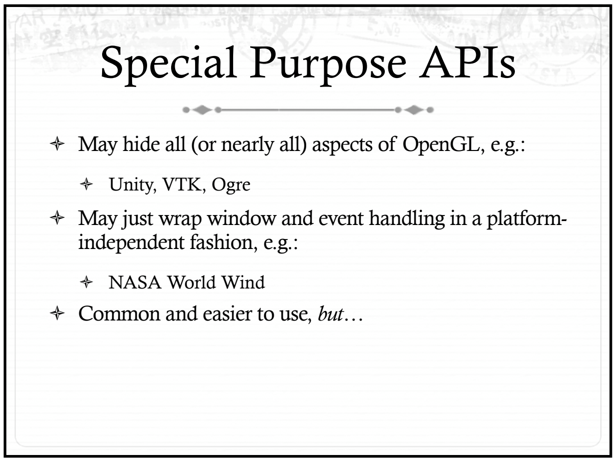
What is 672?

- 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:
  - 2D/3D interactive modeling, rendering, and animation
  - Common graphics algorithms, mathematics, and data structures
  - The MVC design pattern
  - CPU-GPU communication and cooperation



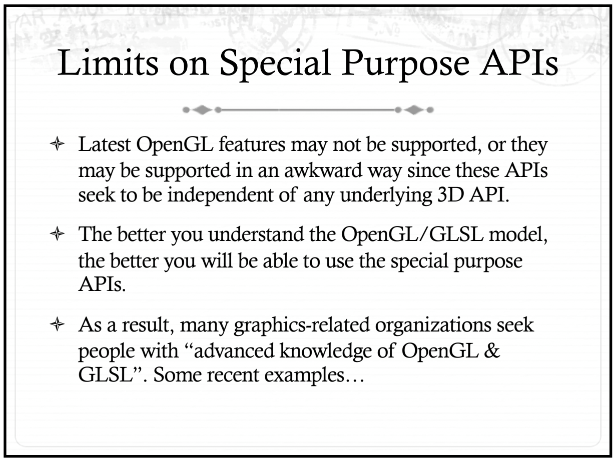
672 is *Not*:

- Graphic design (e.g., Alias, Maya, 3ds Max, LightWave, et al.)
  - Although most such programs are built using OpenGL
- Unity or similar special-purpose APIs
  - OpenGL is general-purpose & platform-independent, but somewhat lower level than Unity.
  - Higher level 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).



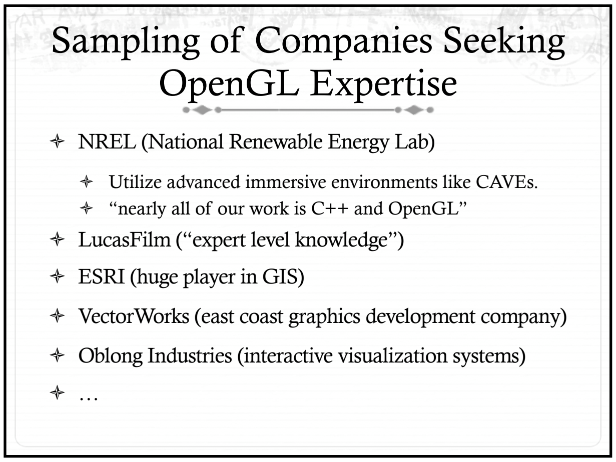
Special Purpose APIs

- 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*…



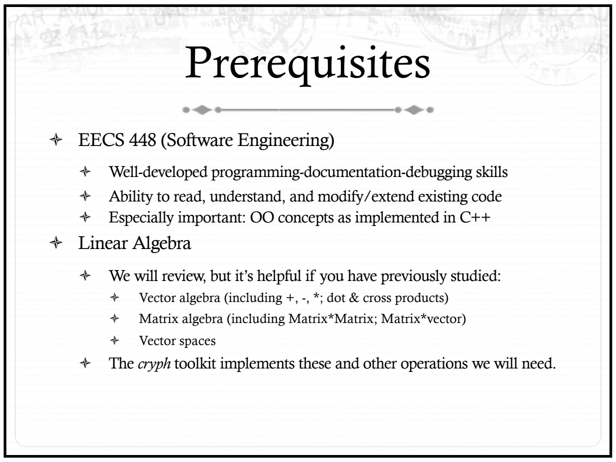
Limits on Special Purpose APIs

- Latest OpenGL features may not be supported, or they may be supported in an awkward way since these APIs seek to be independent of any underlying 3D API.
- 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…



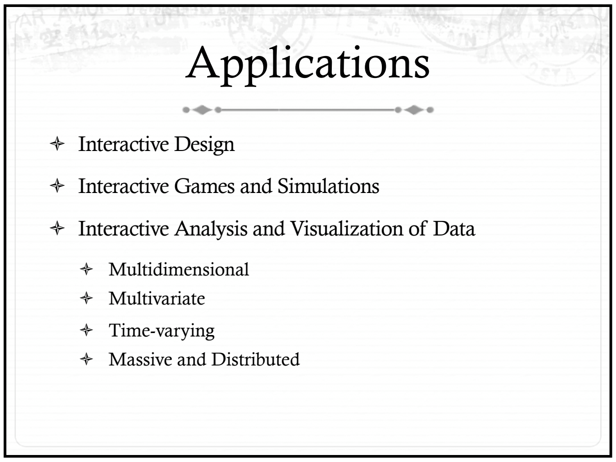
Sampling of Companies Seeking OpenGL Expertise

- 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)
- …



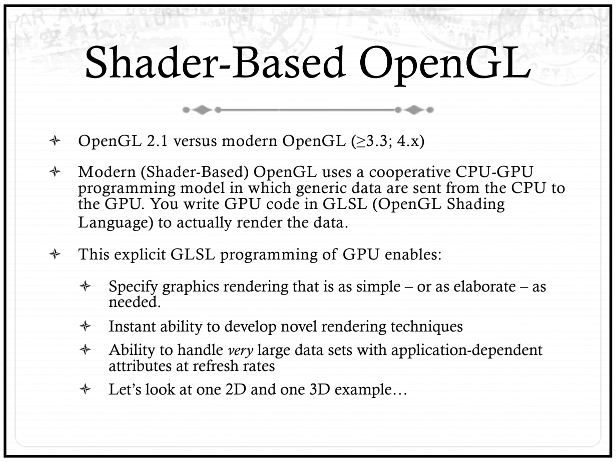
Prerequisites

- EECS 448 (Software Engineering)
  - Well-developed programming-documentation-debugging skills
  - Ability to read, understand, and modify/extend existing 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 products)
    - Matrix algebra (including Matrix*Matrix; Matrix*vector)
    - Vector spaces
  - The *cryph* toolkit implements these and other operations we will need.



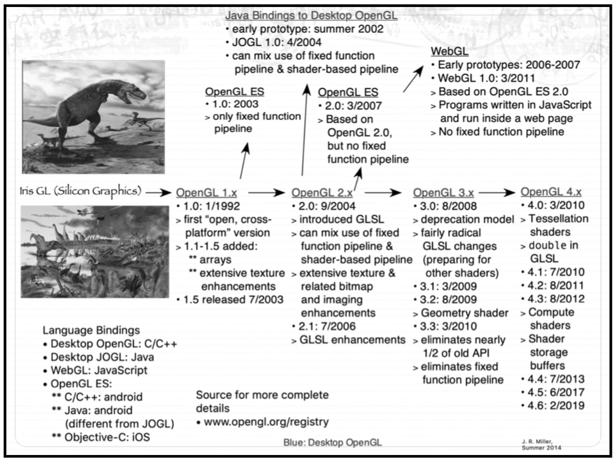
Applications

- 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 (Shader-Based) OpenGL 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:
  - Specify graphics rendering that is as simple – or as elaborate – as needed.
  - 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…



Java Bindings to Desktop OpenGL
- early prototype: summer 2002
- JOGL 1.0: 4/2004
- can mix use of fixed function pipeline & shader-based pipeline

OpenGL ES
- 1.0: 2003
  - > only fixed function pipeline

OpenGL ES
- 2.0: 3/2007
  - > Based on OpenGL 2.0, but no fixed function pipeline

WebGL
- Early prototypes: 2006-2007
- WebGL 1.0: 3/2011
  - > Based on OpenGL ES 2.0
  - > Programs written in JavaScript and run inside a web page
  - > No fixed function pipeline

Iris GL (Silicon Graphics) → OpenGL 1.x → OpenGL 2.x → OpenGL 3.x → OpenGL 4.x

OpenGL 1.x
- 1.0: 1/1992
  - > first "open, cross-platform" version
  - > 1.1-1.5 added:
    - ** arrays
    - ** extensive texture enhancements
- 1.5 released 7/2003

OpenGL 2.x
- 2.0: 9/2004
  - > introduced GLSL
  - > can mix use of fixed function pipeline & shader-based pipeline
  - > extensive texture & related bitmap and imaging enhancements
- 2.1: 7/2006
  - > GLSL enhancements

OpenGL 3.x
- 3.0: 8/2008
  - > deprecation model
  - > fairly radical GLSL changes (preparing for other shaders)
- 3.1: 3/2009
- 3.2: 8/2009
  - > Geometry shader
- 3.3: 3/2010
  - > eliminates nearly 1/2 of old API
  - > eliminates fixed function pipeline

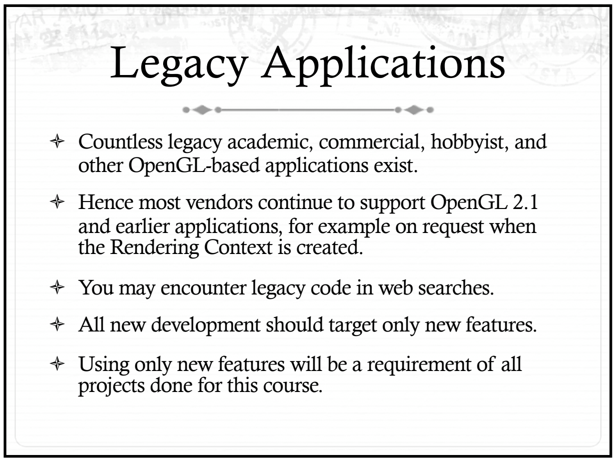
OpenGL 4.x
- 4.0: 3/2010
  - > Tessellation shaders
  - > `double` in GLSL
- 4.1: 7/2010
- 4.2: 8/2011
- 4.3: 8/2012
  - > Compute shaders
  - > Shader storage buffers
- 4.4: 7/2013
- 4.5: 6/2017
- 4.6: 2/2019

Language Bindings
- Desktop OpenGL: C/C++
- Desktop JOGL: Java
- WebGL: JavaScript
- OpenGL ES:
  - ** C/C++: android
  - ** Java: android (different from JOGL)
  - ** Objective-C: iOS

Source for more complete details
- www.opengl.org/registry

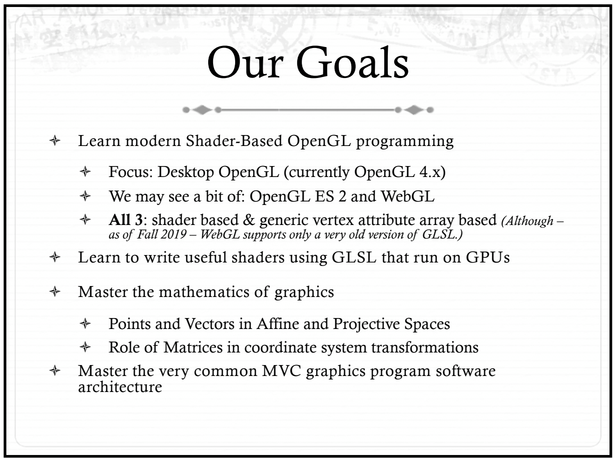
Blue: Desktop OpenGL

J. R. Miller,
Summer 2014



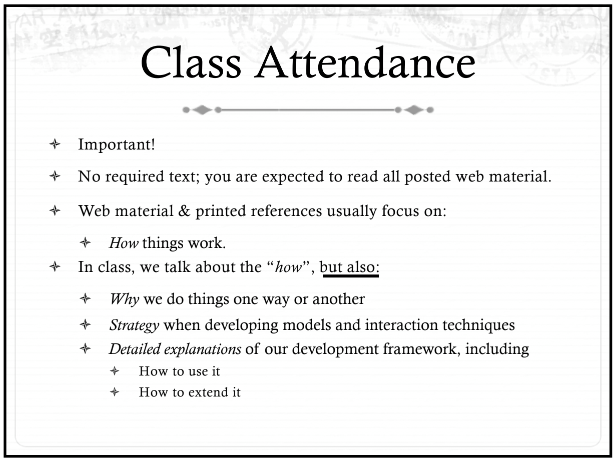
Legacy Applications

- 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.



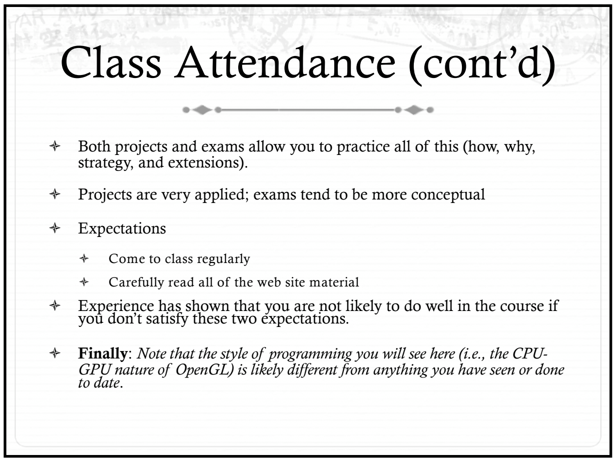
Our Goals

- Learn modern Shader-Based OpenGL programming
  - Focus: Desktop OpenGL (currently OpenGL 4.x)
  - We may see a bit of: OpenGL ES 2 and WebGL
  - **All 3**: shader based & generic vertex attribute array based *(Although – as of Fall 2019 – 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



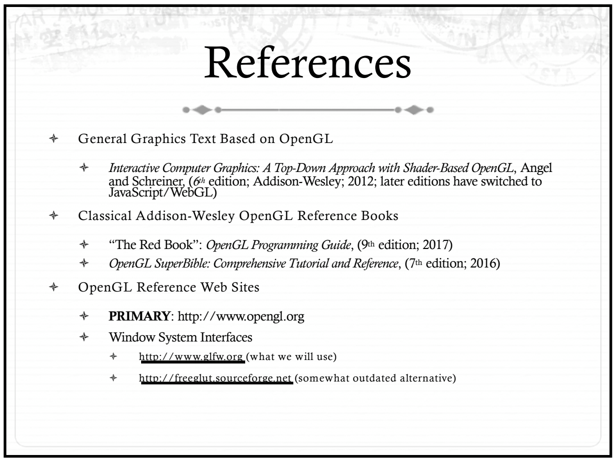
Class Attendance

- Important!
- No required text; you are expected to read all posted web material.
- Web material & printed references usually focus on:
  - *How* things work.
- In class, we talk about the "*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
- Expectations
  - Come to class regularly
  - Carefully read all of the web site material
- Experience has shown that you are not likely to do well in the course if you don't satisfy these two expectations.
- **Finally**: *Note that the style of programming you will see here (i.e., the CPU-GPU nature of OpenGL) is likely different from anything you have seen or done to date.*



References

- General Graphics Text Based on OpenGL
  - *Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL*, Angel and Schreiner, (*6th* edition; Addison-Wesley; 2012; later editions have switched to JavaScript/WebGL)
- Classical Addison-Wesley OpenGL Reference Books
  - "The Red Book": *OpenGL Programming Guide*, (9th edition; 2017)
  - *OpenGL SuperBible: Comprehensive Tutorial and Reference*, (7th edition; 2016)
- 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)