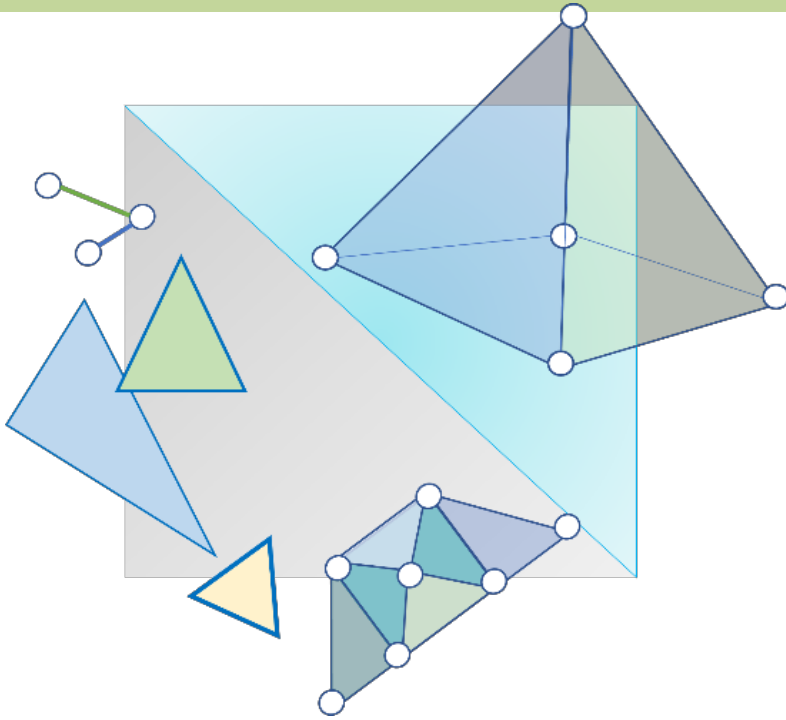
CITS3003 Graphics & Animation

Lecture 2:
Programming with
OpenGL



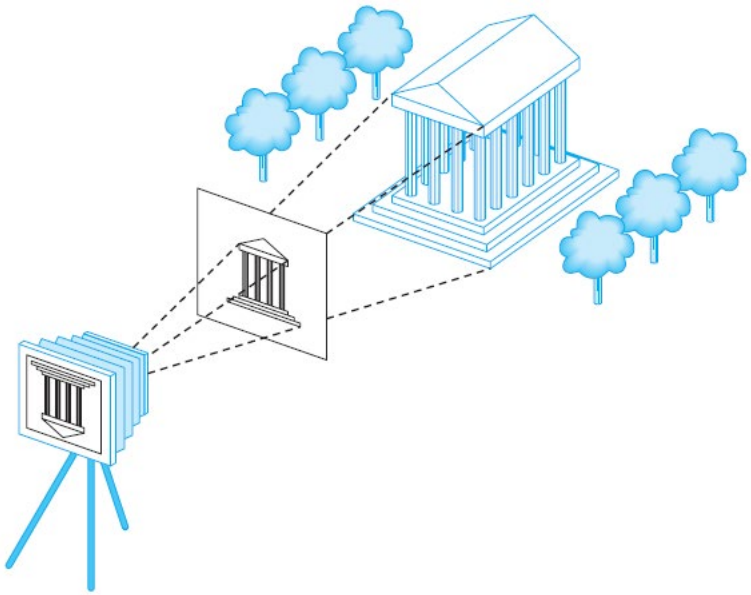
Content

- OpenGL Libraries
- OpenGL State Machine
- OpenGL Variable Types and Functions
- What is GLSL
- A Simple Program

Three-Dimensional APIs

The synthetic-camera model is the basis for a number of popular APIs, including OpenGL and Direct3D.

- If we are to follow the synthetic-camera model, we need functions in the API to specify the following:
  - Objects
  - A viewer
  - Light sources
  - Material properties



What is OpenGL

- Its an API (specifications to be precise)
  - Allows accessing and dealing with the graphics card
  - Contains over 250 functions
  - the latest OpenGL specification 4.6 was released in 2017
- Where do I download OpenGL?
  - Its already there in your graphics driver
- Is it open source?
  - Irrelevant (its essentially just a specification)
- We still treat OpenGL as API

https://www.khronos.org/opengl/
https://registry.khronos.org/OpenGL/specs/gl/glspec46.core.pdf

What is OpenGL (cont…)

- OpenGL is one of many APIs that allow access to the graphics card
  - E.g., Vulkan, Direct 3D 11, Metal
- Why OpenGL
  - Cross-platform
  - Excellent entry point for Graphics learning

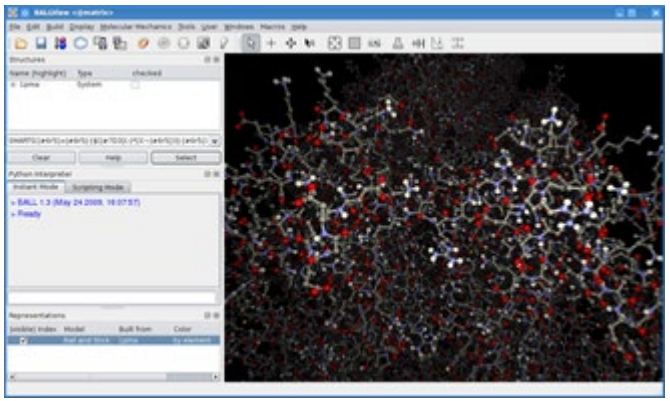
For labs (1-5) Version 3.2+ are all ok!

Modern OpenGL

- Legacy OpenGL (version 3.0 and below) uses set of pre-sets (simple but not flexible)
- Modern OpenGL (version 3.2+) allows the computer program to achieve fast graphics performance by using GPU rather than CPU
- Allows applications to control GPU through programs known as **shaders**
- It is the application's job to send data to GPU; GPU then performs the rendering

OpenGL Applications

- Non exhaustive list of OpenGL applications (link)



Extension Library- GLEW

- Modern versions of OpenGL, such as those found in version 4+, require identifying the extensions available on the GPU.
- It has become standard practice to use an extension library to abstract away the extra coding used to identify and access the OpenGL extensions on a GPU.
- GLEW is an OpenGL Extension Wrangler Library
- GLEW makes it easy to access OpenGL extensions available on a particular system
- Application only needs to include glew.h and run a glewInit()

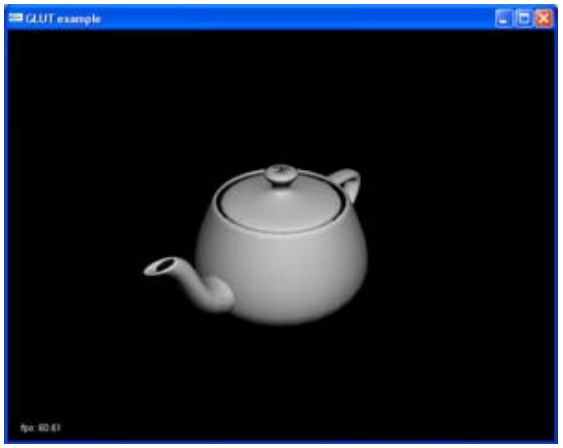
OpenGL/GLUT basics

OpenGL

- OpenGL's function is Rendering (or drawing)
  - Rendering– Convert geometric/mathematical object descriptions into images

- No functions for window management (create, resize, etc)
  -portability across different platforms

- Links with window/windowing system
  - GLX for X window systems
  - WGL for Windows
  - AGL for Macintosh



Credits: Prof. Emmanuel Agu, cs4731, WPI

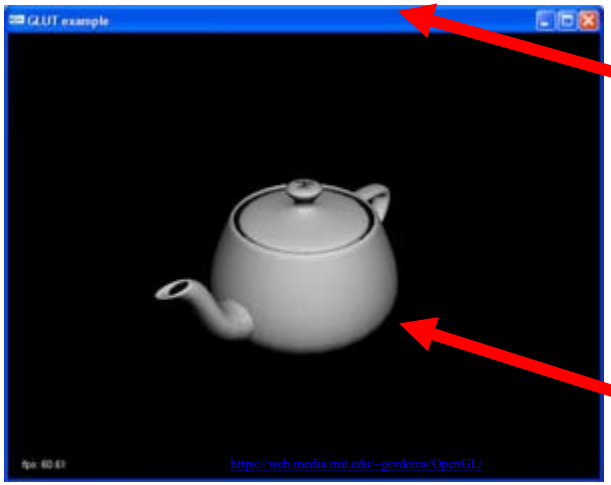
OpenGL/GLUT basics

GLUT

- A window system independent toolkit which:
  - Provides basic functions for window management
  - Interfaces with different windowing systems
  - Offers portability: code is portable between windowing systems.

- GLUT lacks the functionality of a high-end toolkit for a specific platform
  - No slide bars

- GLUT is easy to use and learn, and it works quite well for demos and simple applications.
  - Fast prototyping



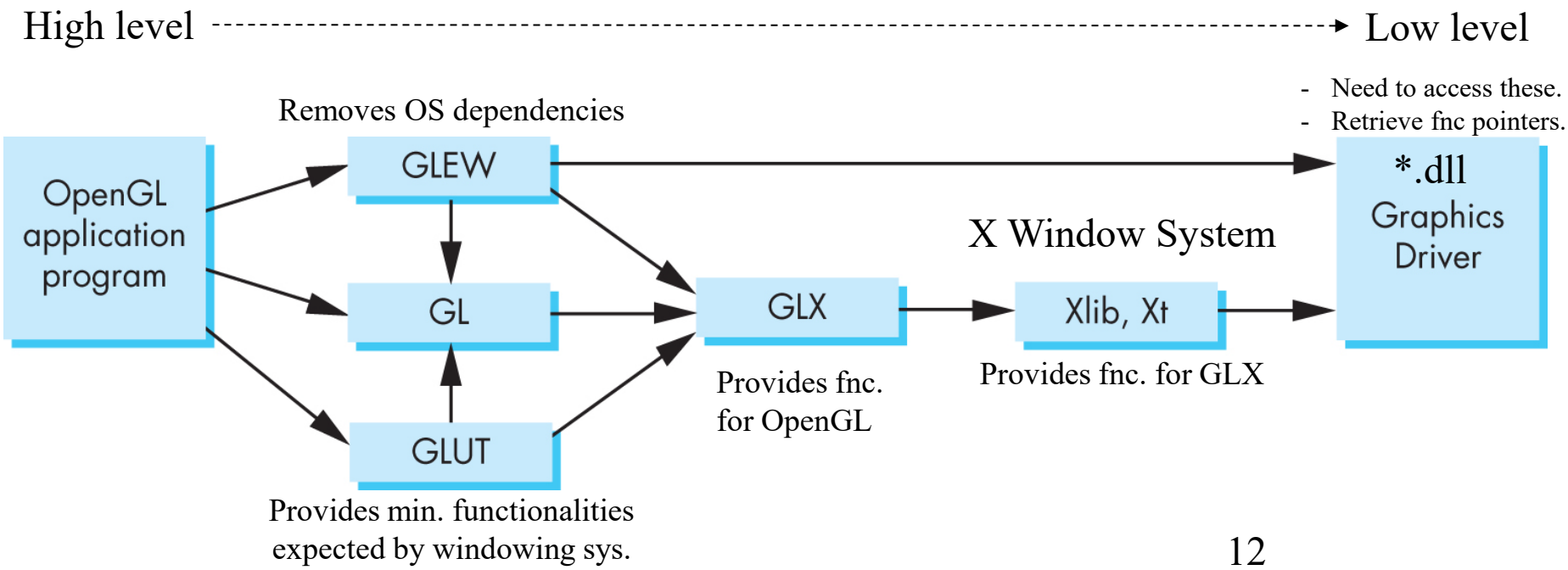
ImageCredits: Prof. Emmanuel Agu, cs4731, WPI

freeGLUT

- GLUT was created long ago and has been unchanged
- freeglut updates GLUT
  - Added capabilities
    - Context checking

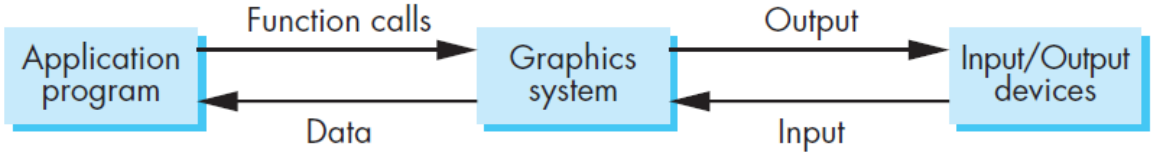
Software Organization

- The application programs can use GLEW, GL, GLUT functions but not directly access to Xlib etc.
- The program can therefore be compiled with e.g., GLUT for other operating systems.

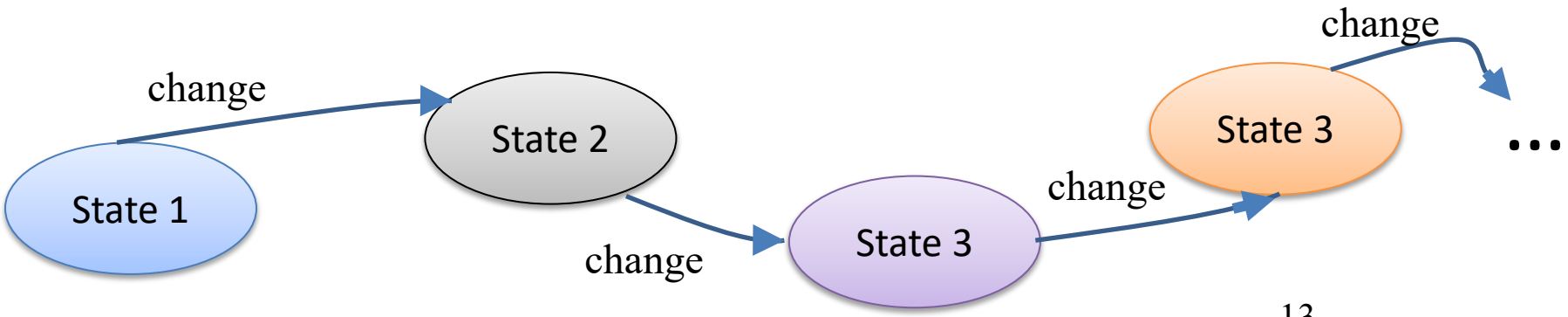


OpenGL as a State Machine

- We can think of the entire graphics system as a **black box** (finite-state machine).



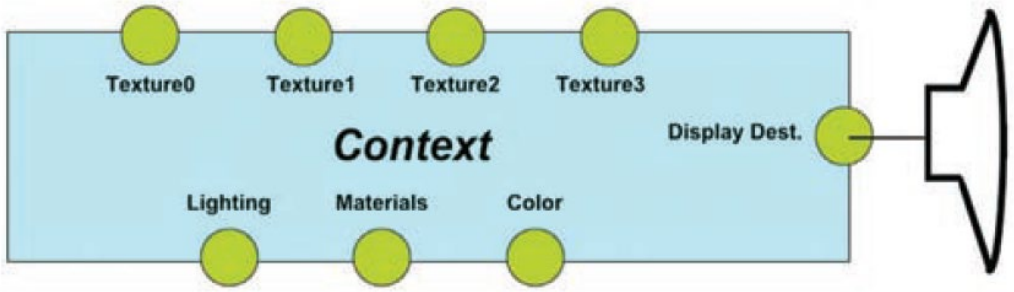
- This black box has inputs coming from the application program.
  - These inputs can change the state (mode) of the machine or can cause the machine to produce a visible output.
- The behavior of Graphics system can be determined by its "state", which can be modified by invoking OpenGL functions



OpenGL Context and State

- OpenGL state machine consists of hundreds of settings that affect various aspects of rendering.
- The state of OpenGL is commonly referred to as the OpenGL context.
- The initial Graphics context has a number of default values for:
  - color or other attributes
  - transformations
  - lighting mode, camera model, textures etc

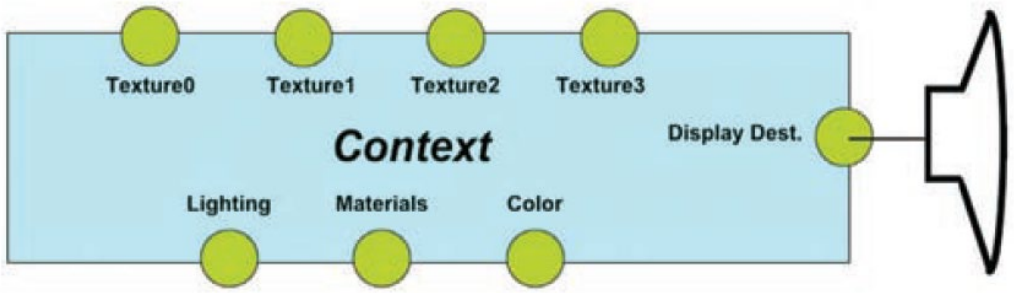
……



The OpenGL state as a graphics context object

OpenGL Context and State

- OpenGL maintains a list for the current values of attributes and other parameters, refers to as “state variables”
  - When we assign a value to one or more state variables, we put the systems into a particular state.
  - State remains in effect until we change the value of a state variable.
  - Each state variable has a default value.



The OpenGL state as a graphics context object

State Machine (cont..)

- From the perspective of the API, there are two types of graphics functions:
  1. Functions that either change the state inside the machine or return the state information
     - glEnable (GL_ LIGHTING);
     - glDisable (GL_BLEND);

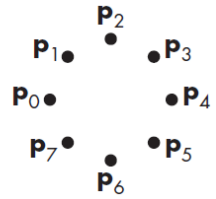
  2. Functions that perform some operations based on the current state of the machine.
     - Primitive functions
       - glColor3f(0.0, 0.0, 0.0);
       - glPointSize(1.5);

OpenGL Functions

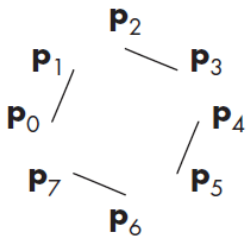
- OpenGL provides a range of functions for specifying:
  - Primitives → the low-level objects or atomic entities that our system can display
    - Points
    - Line Segments
    - Triangles
  - Attributes → the way that a primitive appears on the display
  - Transformations → to carry out transformations of objects, such as rotation, translation, and scaling
  - Viewing → To specify various views
  - Control (GLUT) → to communicate with the window system, initialize our programs, and deal with any errors during the execution
  - Query → to get information about API i.e., how many colours are supported etc.,

What are OpenGL Primitives?

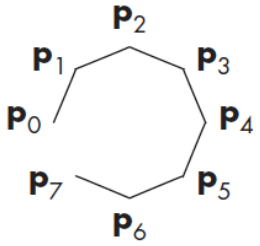
OpenGL primitives are standard set of basic geometric shapes that are used to create complex 3D models in OpenGL. They include points, lines, polylines and polygons.



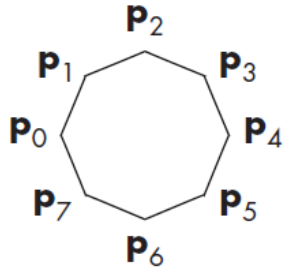
GL_POINTS



GL_LINES

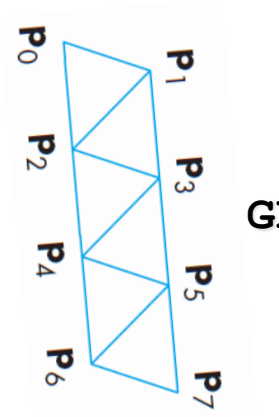


GL_LINE_STRIP

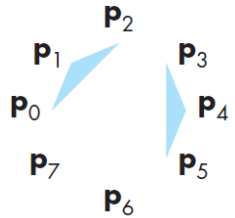


GL_LINE_LOOP

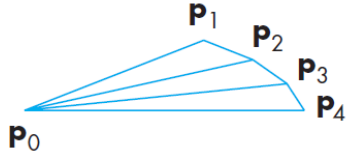
Polygons have a well-defined interior



GL_TRIANGLE_STRIP



GL_TRIANGLES

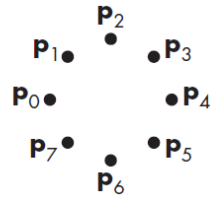


GL_TRIANGLE_FAN

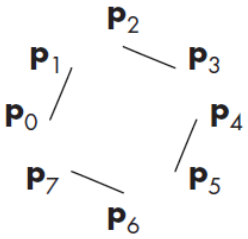
OpenGL primitives are specified by sets of vertices

What are OpenGL Primitives?

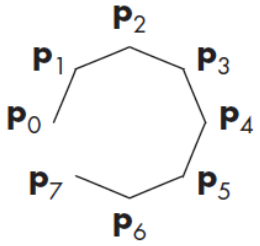
OpenGL primitives are standard set of basic geometric shapes that are used to create complex 3D models in OpenGL. They include points, lines, polylines and polygons.



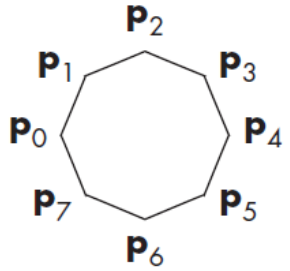
GL_POINTS



GL_LINES



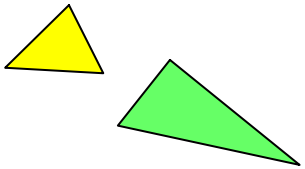
GL_LINE_STRIP



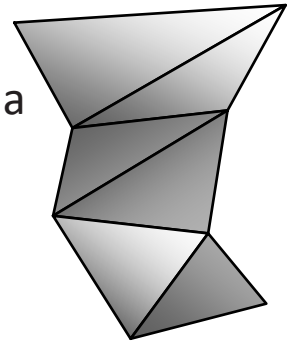
GL_LINE_LOOP

OpenGL primitives are specified by sets of vertices

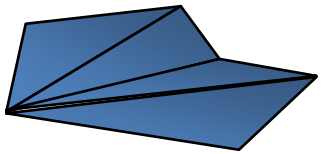
Polygons have a well-defined interior



GL_TRIANGLES



GL_TRIANGLE_STRIP



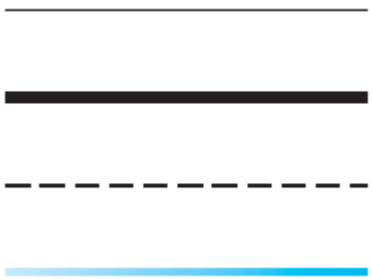
GL_TRIANGLE_FAN

OpenGL Types

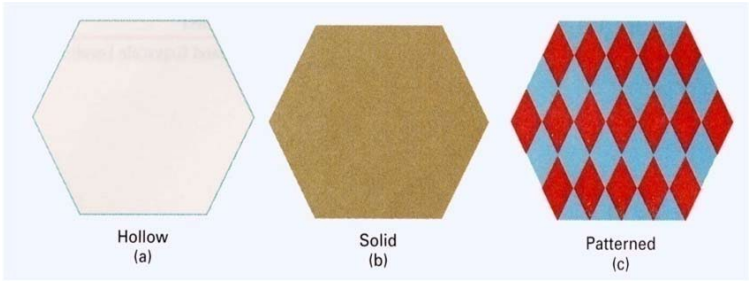
- In OpenGL, we use basic OpenGL types, e.g.,
  - GLfloat,
  - GLdouble,
  - GLint, etc
  - (equivalent to float, double, and int in C/C++)
- Additional data types are supplied in header files vec.h and mat.h from Angel and Shreiner, e.g.,
  - vec2,
  - vec3,
  - mat2,
  - mat3
  - mat4, etc.

What are Attributes?

- Attributes are properties associated with the primitives that give them their different appearances, e.g.
  - Color (for points, lines, polygons)
  - Size and width (for points, lines)
  - Stipple pattern (for lines, polygons)
  - Polygon mode
    - Display as filled: solid color or stipple pattern
    - Display edges
    - Display vertices



Attributes for lines



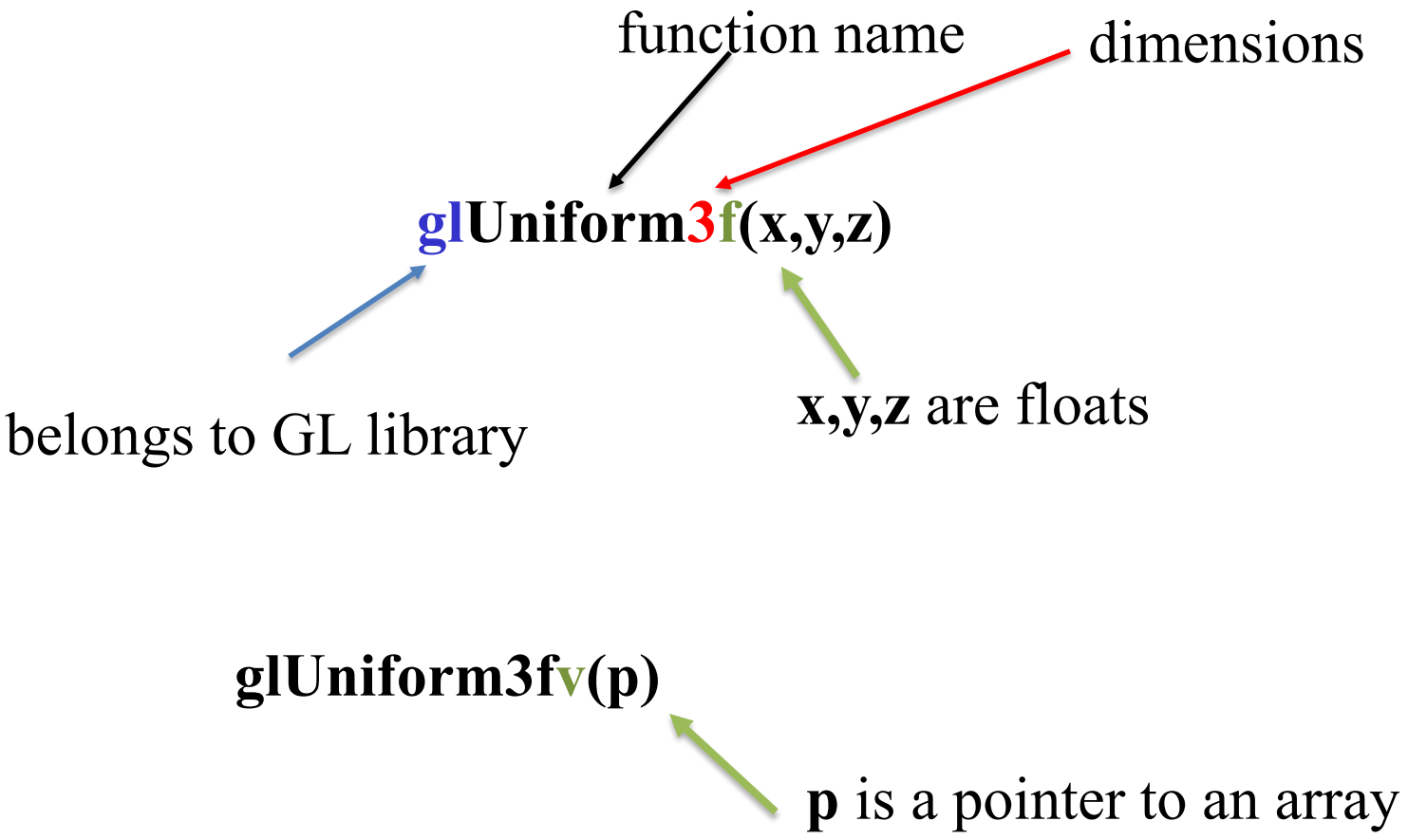
Attributes for polygons

OpenGL Functions:
Lack of Object Orientation

- OpenGL is not object oriented so that there are multiple forms of a given logical function, e.g., the following are the same function but for different parameter types: (no overloading)
  - **glUniform3f**
  - **glUniform2i**
  - **glUniform3fv**
- The major reason is efficiency (Don't wrap everything in classes when it is not required)

Format of OpenGL Functions

glUniform — Specifies the value of a uniform variable for the current program object



Uniform variables are used to communicate with vertex or fragment shaders from outside. We will come to the details later.

OpenGL #defines

- Most constants are defined in the include files **gl.h**, **glu.h** and **glut.h**
  - Note **#include <GL/glut.h>** should automatically include the others
  - Examples: the functions **glEnable** **and** **glClear** are both declared in **gl.h**
- The OpenGL data types **GLfloat**, **GLdouble**,…. are also declared in **gl.h**

slido

Which of the following options is correct?

ⓘ Start presenting to display the poll results on this slide.

Components of a OpenGL application

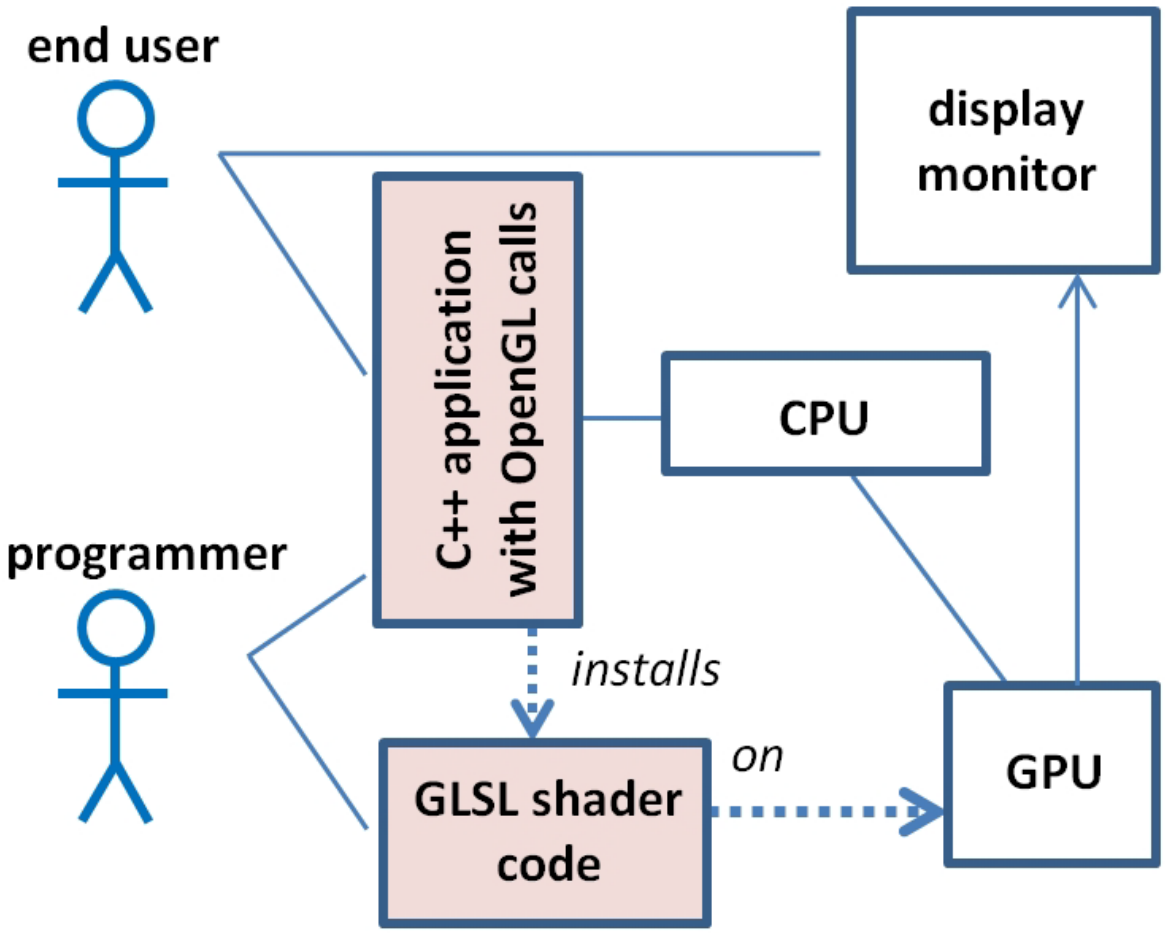


Image credits: Computer Graphics Programming in OpenGL with C++, Scott Gordon and John Clevenger

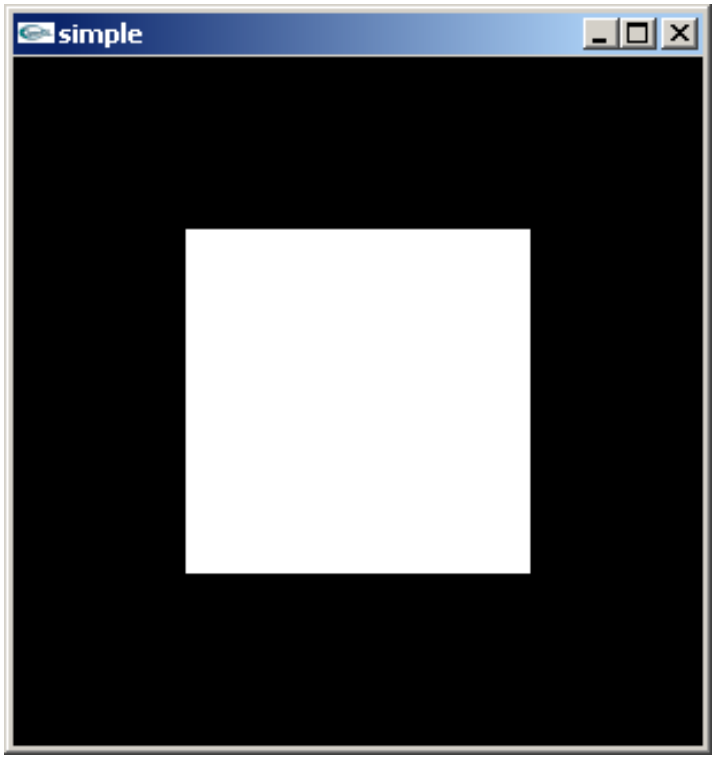
What is GLSL

GLSL is short for OpenGL Shading Language

- It is a C-like language with:
  - Built-in Matrix and vector types (2, 3, 4 dimensional)
  - C++ like constructors
- It is similar to Nvidia's Cg and Microsoft HLSL
- Supports loops, if-else constructs, but recursion is not allowed
- GLSL codes are not stand-alone applications, they require an application program that uses OpenGL API
- More on GLSL in later lectures

A Simple Program

simple.cpp - Generates a white square on a black background



For the above task, following are the **Rendering Steps:**

1. Generate vertices (2 triangle = 6 vertices)
2. Store the vertices into an array
3. Create GPU buffer for vertices
4. Move array of 6 vertices from CPU to GPU buffer
5. Draw 6 points from array on GPU using glDrawArray

Further Reading

"Interactive Computer Graphics – A Top-Down Approach with Shader-Based OpenGL" by Edward Angel and Dave Shreiner, 6th Ed, 2012

- Sec. 2.4 Primitives and Attributes (up to Sec. 2.4.1)
- Sec. 2.3.1 Graphics Functions
- Sec. 2.3.2 Graphics Pipeline and State Machine
- Sec. 8.10 Graphics and the Internet
- (Advanced) Appendix A.2

Graphics Shaders, Theory and Practice, 2nd Edition, Mike Baily, Steve Cunningham