Discussion 1
CSE 167
Outline

- Brief introduction to OpenGL, GLFW, GLEW and GLM
  - What are they?
  - How to use them?
- Homework 1 walk through and demo
What is OpenGL?

- Open Graphics Library.
- Render 3D graphics efficiently.
- Cross-language and cross-platform.
- More like a specification...
- Read more here: https://learnopengl.com/Getting-started/OpenGL
What is GLEW?

- OpenGL Extension Wrangler Library.
- A C/C++ library that loads modern OpenGL.
- Can only use old OpenGL without it.
- Since Apple maintains their own version of OpenGL, you don’t need GLEW on macOS.
What is GLFW?

- Graphics Library Framework
- A C++ library that handles creating windows and interacting with the windows.
- Provide an OpenGL context such that we can draw in the window.
What is GLM?

- OpenGL Mathematics
- A header only C++ math library.
- Follow the OpenGL Shading Language (GLSL) specifications.
- To succeed in this course, you should learn it well!
GLEW and GLFW usage: minimal example

`glfwInit();` // Initialize GLFW.

`GLFWwindow* window= glfwCreateWindow(640, 480, "Hello World", NULL, NULL);` // Create a 640x480 window named “Hello World”.

`glfwMakeContextCurrent(window);` // In short, let you access OpenGL.

`glewInit();` // Initialize GLEW. Don’t need this step on macOS.

......
GLEW and GLFW usage: minimal example

......
while (!glfwWindowShouldClose(window)) { // Main loop.

    glClear(GL_COLOR_BUFFER_BIT); // Clean the window.

    ...... // Do the rendering here.

    glfwSwapBuffers(window); // Display the rendered content.

    glfwPollEvents(); // Handle events such as keyboard inputs.
}
Window::cleanUp(); // Clean up objects
glfwDestroyWindow(window); // Deallocate window
glfwTerminate(); // Terminate GLFW.
GLFW: key callback

```c
void keyCallback(GLFWwindow* window, int key, int scancode, int action, int mods); // Where you handle keyboard events.
```

- Is the key pressed: `action == GLFW_PRESS`
- Is the key F1: `key == GLFW_KEY_F1`
- Is SHIFT pressed: `mods == GLFW_MOD_SHIFT`
- Do something when the user press “A”:

```c
if (action == GLFW_PRESS && key == GLFW_KEY_A && mods == GLFW_MOD_SHIFT) ......
```
GLM usage

Frequently used data types:

\texttt{glm::vec3, glm::vec4, ...} // Vector or point

\texttt{glm::mat3, glm::mat4} // Matrix

Some examples:

\texttt{glm::vec3 a(1, 2, 3)} // A vector [1, 2, 3]

\texttt{glm::mat3 b(5)} // A 3x3 diagonal matrix diag(5)
GLM usage

a.x // The first element of the vector a.

b[0] // The first COLUMN of matrix b, which is a vec3 since it is a column of mat3

b[0][1] or b[0].y // The second element of the first column of b.

Read more here (highly recommended):

GLM usage

glm::vec3 a, b;
glm::mat3 p, q;

a + b // Vector addiction;

a * 42 // Vector-Scalar multiplication

p * a // Matrix-Vector multiplication

p * q // Matrix multiplication
GLM usage

Transformation matrices:

glm::translate, glm::rotate, glm::scale, ...

You will learn more about them in class. For HW1, you don’t have to use them, although you can if you want.

Read about them here:

https://glm.g-truc.net/0.9.2/api/a00245.html
Homework 1 Rendering Point Clouds

http://ivl.calit2.net/wiki/index.php/Project1F20
OBJ File Structure

- **v** (Vertex)
  - General Format: v v_x v_y v_z r g b
  - v_x, v_y, v_z are the vertex x, y, and z coordinates
  - r g b are colors

- **vn** (Vertex Normal)
  - General Format: vn vn_x vn_y vn_z
  - vn_x, vn_y, vn_z are the vertex normal x, y, and z coordinates

- **f** (Faces)
  - General Format: f v1 // vn1 v2 // vn2 v3 // vn3
  - v1, v2, and v3 are index of a vertex
  - vn1, vn2, and vn3 are index of a normal vertex
Part 2) The Point Cloud

A set of points in 3D

- No meshes, no triangles, just points

Your job is to read them into memory

- Namely: into a PointCloud object
- Check out the parser code in the project description for ideas
Part 3) Displaying your point cloud

- Dynamically allocate a new point cloud and pass in the .obj file name, then set the current object to that point cloud. Don't forget to deallocate it later!

```
// Create a point cloud consisting of cube vertices.
cubePoints = new PointCloud("foo", 100);

// Set cube to be the first to display
currObj = cube;
```

- The draw call to currObj and inheritance will do the rest

```
// Render the objects
currObj->draw(view, projection, shaderProgram);
```

- Check out Window::keyCallback for an idea how to change the rendered object and more by key presses
- For changing the point size, check out glPointSize
Good modern OpenGL tutorials

https://learnopengl.com/

https://www.opengl-tutorial.org/beginners-tutorials/
Any questions?
Most common and difficult to solve post-discussion questions:

1. **OSX user Error:**

   ```
   dyld: Library not loaded: @rpath/libglfw.3.dylib  Referenced from:
   /Users/xxx/Library/Developer/Xcode/DerivedData/CSE_167-xxx/Build/Products/Release/CSE 167  Reason: no suitable image found.  Did find:
   /usr/local/lib/libglfw.3.dylib: code signature in (/usr/local/lib/libglfw.3.dylib) not valid for use in process using Library Validation: mapped file has no cdhash, completely unsigned? Code has to be at least ad-hoc signed.
   ```

   check project -> target -> Signing & Capabilities, and remove Headened Runtime. Update Signing Certificate in Signing to "Sign to Run Locally". This seems to happen when you specify a team name, please try to setup the project with minimal specifications.