CSE 190 Discussion 2

HW1: Where's Waldo 3D

Agenda

- Homework 1 Recap
- OpenGL basics recap
- Assimp usage explained
- Technical tips
- Extra Credit Hints

Homework 1 Recap

- Link to the assignment: http://ivl.calit2.net/wiki/index.php/Project1S18
- Due Date: April 20th 2pm (This Friday!)
  - Present project in person
- Features you need to implement:
  - You will generate a 3D grid of spheres (5 x 5 x 5)
  - Randomly highlight one of them
  - Select it with the Touch Controller
  - More specifications in the assignment page.

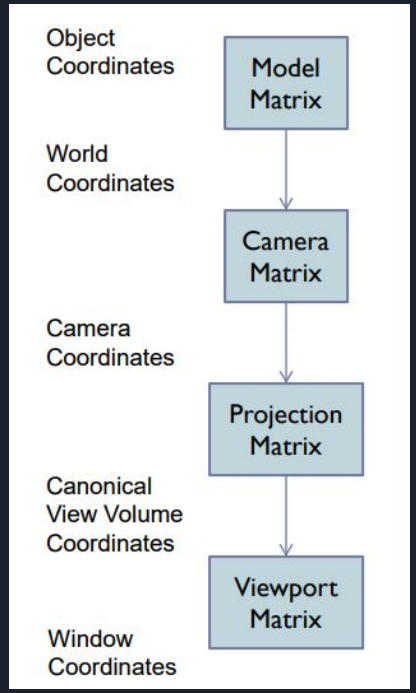
OpenGL Basics

- Rendering Pipeline
- MVP matrix
- Shaders Quick Review

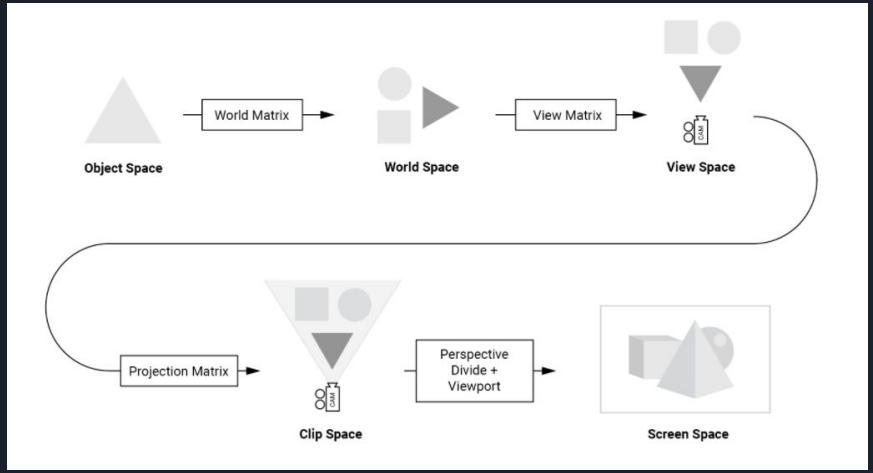
Rendering Pipeline

- a.k.a: graphics transformation pipeline

- Refresh your memory with the rendering pipeline
  - p’ = DPC^-1Mp
  - M is model matrix
  - C is Camera matrix
  - P is projection matrix
  - D is viewport matrix
- How will things be different in VR?
  - More layers of transforms specified for each eye
    - See this link for more information
  - Oculus SDK handles that for you.



MVP Matrices



- A simplified model in OpenGL
  - M is Model Matrix
  - V is the View Matrix
  - P is the Projection Matrix
- Why do I care all these?
  - Simply put, you need to configure the transform of your objects and camera.
  - Quote from OpenGL: *"The Model, View and Projection matrices are a handy tool to separate transformations cleanly. You may not use this. But you should. This is the way everybody does, because it's easier this way."*

MVP Matrices

- Sounds good... but how do I implement them in my program?
  - M: the toWorld matrix in your model class
    - To modify the scale and position of your sphere
  - V: the matrix obtain from HMD orientation and player position
    - Oculus SDK gives you the eyePose orientation
    - You can define the player position if there is a manual movement
  - P: the eyeProjection matrix for each eye (handled by starter code)

```
void renderScene(const glm::mat4 & projection, const glm::mat4 & headPose) override {

    //Get the transformation matrix for user position to apply to headPose.
    glm::mat4 transMat = glm::translate(glm::mat4(1.0f), player->position);

    scene->render(projection, glm::inverse(transMat * headPose));
}
```

- We will give some idea about the render hierarchy in later slides

Shader Quick Review

- Vertex Shader
  - Part of the early steps in the graphic pipeline.
  - Nothing is yet rendered at this stage
  - gl_position is the mandatory output variable
- An example of Vertex shader

```
#version 330 core
layout (location = 0) in vec3 position;
layout (location = 1) in vec3 normal;

out vec3 Normal;

// Uniform variables can be updated by fetching their location and passing values to that location
uniform mat4 projection;
uniform mat4 model;
uniform mat4 view;

void main()
{
    // OpenGL maintains the D matrix so you only need to multiply by P, V (aka C inverse), and M
    gl_Position = projection * view * model * vec4(position.x, position.y, position.z, 1.0);

    Normal = mat3(transpose(inverse(model))) * normal;
}
```

Shader Quick Review

- Fragment Shader
  - Same as the pixel shader
  - Part of the rasterization step
  - pixels between vertices are colored and lights are applied
- An example of Fragment shader

```
#version 330 core
in vec3 Normal;
out vec4 color;

uniform int colorValue;


void main()
{
    color = vec4(colorValue,0,0,0.2);
}
```

Shader Quick Review

- Get the reference to your shaderProgram

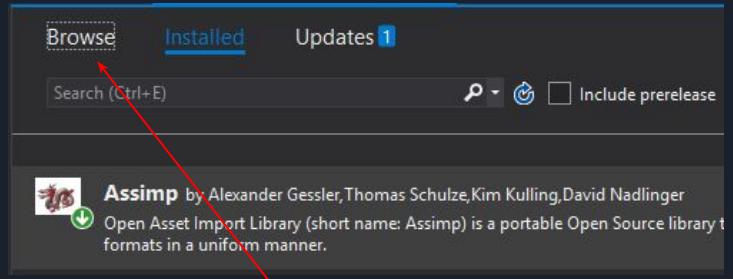
```
#define DEFAULT_VERTEXT_SHADER_PROGRAM "assets/shaders/shader.vert"
#define DEFAULT_FRAGMENT_SHADER_PROGRAM "assets/shaders/shader.frag"
GLuint defaultShaderProgram = LoadShaders(DEFAULT_VERTEXT_SHADER_PROGRAM, DEFAULT_FRAGMENT_SHADER_PROGRAM);
mySphere -> render(defaultShaderProgram, viewPortInfo, position, radius)
```

- Parse variables to shader program
  - Retrieve the reference to shaderProgram and your variable in shader
    - GLint location = glGetUniformLocation(shaderProgram, "#yourVariableName")
  - Parse MVP matrices:
    - void **glUniformMatrix4fv**(GLint location, GLsizei count, GLboolean transpose, const GLfloat *value);
  - Parse numerical value:
    - void **glUniform1f** (GLint location, GLfloat v0);
    - void **glUniform1i** (GLint location, GLint v0);
  - More data types: See the link to glUniform

Assimp Usage

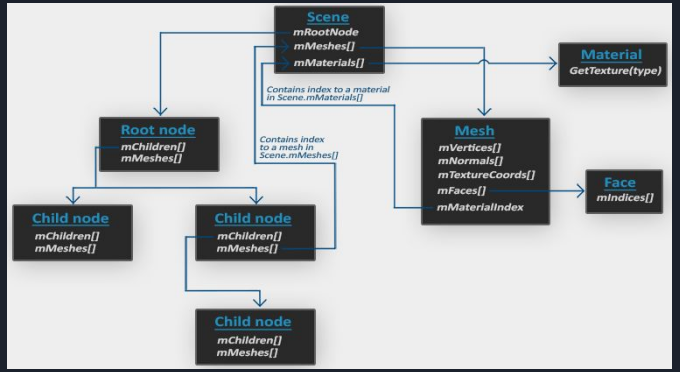
- Import Assimp
- Make it compile and run
- Make use of the method that you actually need

Import Assimp



- Add assimp packages to your project:
  - See last discussion slides for details
  - If you have trouble compiling the assimp libraries on your own, simply go to Nuget and search for assimp. Then install and it solves everything for you.
- Include assimp:
  - #include <assimp/Importer.hpp>
  - #include <assimp/scene.h>
  - #include <assimp/postprocess.h>

Make it compile and run



- If you are copying over from the tutorials...
  - Tutorials provide Shader.h, Model.h and Mesh.h
    - Shader.h wraps some methods for accessing shaders
    - Model.h explains how you can retrieve model information from the loaded scene
    - Mesh.h is a class to store the retrieved mesh and textures.
  - You need to replace <glad> with <GL/glew.h> if you are not familiar with it
  - It uses stb library, so you might need to add stb_images.h own your own
    - But you don't need textures for regular points in this assignment

The methods that you actually need

- I have no idea how to use those classes...
  - Don’t panic.
  - It’s natural, since it is a tutorial sample code, not the library for you to use
  - The only things you need are some methods in Model.h
    - `void loadModel(string const &path)`
    - `void processNode(aiNode *node, const aiScene *scene)`
    - `Mesh processMesh(aiMesh *mesh, const aiScene *scene)`
    - And some code snippets for processing materials and textures
  - processMesh()
    - It extracts the vertices and normals from the obj file
    - You might need to modify this method so that you know where to store these information and where to render them

Technical Tips

- Access the controller information
- Summon console
- Count time and random number
- Run code without headset

Access Controller position/rotation

- hmdState.handPoses[isRight] gives you the controller information
- Use ovr::toGlm to convert the coordinates to the world coordinates

```
// get the general state hmdState
double ftiming = ovr_GetPredictedDisplayTime(_session, 0);
ovrTrackingState hmdState = ovr_GetTrackingState(_session, ftiming, ovrTrue);

// Get the state of hand poses
ovrPoseStatef handPoseState = hmdState.HandPoses[isRight];

//Get the hand pose position.
glm::vec3 controllerPosition = ovr::toGlm(handPoseState.ThePose.Position);

// Get hand rotation
glm::quat controllerRotation = ovr::toGlm(ovrQuatf(handPoseState .ThePose.Orientation));
```

Access Controller buttons

- Check ovr_Buttons in OVR_CAPI.h
- Understand ovr_inputState
- Example:

```
typedef enum ovrButton_
{
    ovrButton_A          = 0x00000001, /// A button
    ovrButton_B          = 0x00000002, /// B button
```

```
// Get Current Input State
ovrInputState inputState;
ovr_GetInputState(session, ovrControllerType_Touch, &inputState);
if (inputState.Buttons & ovrButton_A){
        Printf("A is pressed")
}
```

- Fun fact:
  - Note the bitwise operator &, it works here because ovrButton uses one-hot encoding so that & will be evaluated to 0 as long as values on the two sides are different

Summon Consoles

- Consoles can be initialized with this codes:

```
AllocConsole();
freopen("conin$", "r", stdin);          // give access to reading
freopen("conout$", "w", stdout);        // give access to writing to stdout
freopen("conout$", "w", stderr);        // give access to writing to stderr
printf("Debugging Window:\n");          // print a sample message
```

- Try figuring out where to put those codes

Count time using OculusSDK

- Having access to the time is important
  - You need to count a time of one minute in your game
  - It is also needed if you want to implement ovr_avatar that displays player's hand models
- Get the current time
  - You can use the time offered by time.h
  - However, oculusSDK has a handy way for it
    - double currentTime = ovr_GetTimeInSeconds();
  - Simulate a deltaTime using deltaTime = newTime - oldTime
  - Use delta time to compute the time passed.

A note about generating random number:

int RandomNum = rand()%[the range of random number] + [the smallest number in range]

Run code without the headset

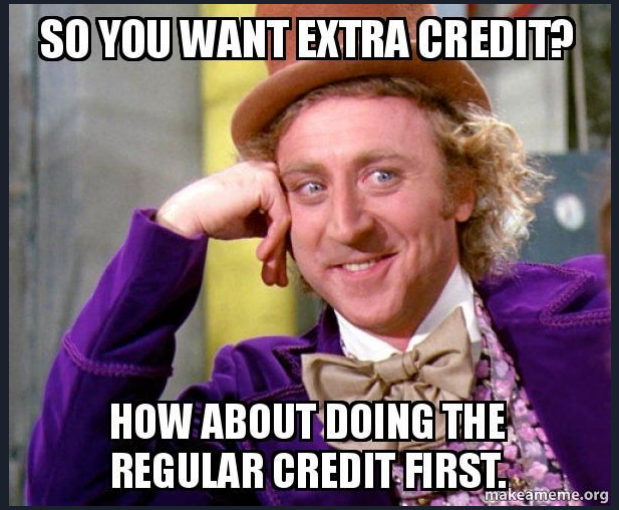
- Note: It may not work after Oculus App 1.14.x (still confirming with Oculus support)
- Since Oculus does not allow downloading/running with the lower-version runtime other than up-to-date version, running code without the headset may not be a feasible way to everyone.

> 3.2 Oculus Runtime Software. Subject to these Terms, including the license provided in Section 3.1, you may access, install, and use the Oculus Runtime Software ("Runtime"). In order to maximize your enjoyment, safety, and overall experience through our Services, the Runtime may only be used with Oculus approved hardware devices and with software developed using the Oculus Rift Software Development Kit, as specified in the Oculus Rift Software Development Kit license agreement. We also require that you use only the then-current version of the Runtime. You acknowledge that the Runtime incorporates proprietary information, and that you will not disclose it to any other person or entity.

- There is an unofficial way to downgrading to Oculus 1.3 (in reddit), but try it with your own risk in your own pc if that's really you want, and it takes time. (Note: VR Lab PC is not capable to do that since school restrict the Administrator authority on it)
- After you installed Oculus 1.3, refer to this thread to initialize the OclulusDebugTool.exe

Extra Credit

- It is for bonus.
- Do the regular part first.
- We tend not to provide detailed help for it, so as in future assignments

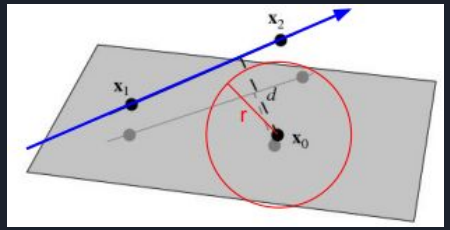


Extra Credit Tips
Oculus Avatar

- Download Avatar SDK:
  - Avatar SDK
- Official documentation on how to use Avatar SDK:
  - Getting Started with Avatar SDK
- You can find some simple online samples on how to use the Avatar SDK in the link above
- How do I import?
  - Refer to discussion 1 slides about how to add packages and resolve linker dependencies.

Extra Credit Tips
New Game Play

- Be creative!
  - New ways of selection/manipulation?
  - Make the environment more interesting?
  - Make the set of spheres dynamic?
    - A dodgeball game ?
  - How would you implement a laser pointer?
    - Point-line distance reference



Extra Credit Tips
Display Text in the VR headset

- Text Rendering
  https://learnopengl.com/In-Practice/Text-Rendering
- Text rendering with OpenGL and C++
  https://rdmilligan.wordpress.com/2018/03/21/text-rendering-with-opengl-and-c/
- Drawing Text in a Double-Buffered OpenGL Window
  https://msdn.microsoft.com/en-us/library/windows/desktop/dd318305%28v=vs.85%29.aspx
- Rendering things to the Oculus Rift
  https://developer.oculus.com/documentation/pcsdk/latest/concepts/dg-render/

Extra Credit Tips
Wrap Waldo as texture for the sphere

Waldo's Picture (link) -> Sphere's texture

- Some help resource:
- Texturing a Sphere
  https://www.khronos.org/opengl/wiki/Texturing_a_Sphere
- How to wrap an image around a sphere in opengl
- https://stackoverflow.com/questions/22980246/i-want-to-wrap-an-image-around-a-sphere-in-opengl
- GLSL Programming/GLUT/Textured Spheres
  https://en.wikibooks.org/wiki/GLSL_Programming/GLUT/Textured_Spheres

QUESTIONS?