CS 450: Assignment 02

Programming Assignments (95%)

Setup

- **This assignment is in C++.**
- Copy src/app/Assign01.cpp and name it **src/app/Assign02.cpp**
- Replace “Assign01” in the application name and window title with “Assign02”
- Make a copy of the vulkanshaders/Assign01 folder and name it **vulkanshaders/Assign02**
- Modify **CMakeLists.txt** by adding the following line to the end of the file:
    - CREATE_VULKAN_EXECUTABLE(Assign02)
- Make sure the program configures, compiles, and runs as-is

New/Updated Code

- Add the following includes:
    - **#include <assimp/Importer.hpp>**
    - **#include <assimp/scene.h>**
    - **#include <assimp/postprocess.h>**
    - **#include "MeshData.hpp"**
- Create a struct to hold information for a vertex:
    - **struct Vertex {**
    - **glm::vec3 pos;**
    - **glm::vec4 color;**
    - **};**
- Create a struct to hold scene data:
    - **struct SceneData {**
    - **vector<VulkanMesh> allMeshes;**
    - **const aiScene *scene = nullptr;**
    - **};**
- Create a global instance of that struct:
    - **SceneData sceneData;**
- Create a new class **Assign02RenderEngine** that inherits (publicly) from **VulkanRenderEngine**:
    - ***Constructor:***
        - **Assign02RenderEngine(VulkanInitData &vkInitData) : VulkanRenderEngine(vkInitData) {};**
    - **virtual bool initialize(VulkanInitRenderParams *params) override**
        - Call the superclass function first
            - **if(!VulkanRenderEngine::initialize(params)) { return false; }**
        - Return true

- ***Destructor:***
  - **virtual ~Assign02RenderEngine() {};**
- **virtual void recordCommandBuffer(void *userData, vk::CommandBuffer &commandBuffer, unsigned int frameIndex) override**
  - Most of this will be the same as the code in VKRender.cpp, EXCEPT:
    - Cast the userData as a SceneData pointer:
      - **SceneData *sceneData = static_cast<SceneData*>(userData);**
    - Loop through and **recordDrawVulkanMesh()** on each **VulkanMesh** in **sceneData->allMeshes.**

- Create a new function: **void extractMeshData(aiMesh *mesh, Mesh<Vertex> &m)**
  - The overall purpose of this function to grab the vertex positions and shape indices from the given **aiMesh** and store this information in the provided **Mesh** struct.
  - Clear out the Mesh's vertices and indices.
  - Loop through all vertices in the **aiMesh** (**mesh->mNumVertices**):
    - Create a **Vertex**.
    - Grab the vertex position information from **mesh->mVertices[i]** and store it in the **Vertex's** position.
      - Note: **mVertices** is an array of **aiVector3D** structs, so you will have to somehow convert to **glm::vec3** structs
    - Set the color of the Vertex to **any color other than (0,0,0,1) or the background.**
      - You may use different colors per vertex using any reasonable scheme.
      - However, alpha values should always be 1.0f.
    - Add the **Vertex** to the **Mesh's** vertices list.

  - Loop through all faces in the **aiMesh** (**mesh->mNumFaces**):
    - Grab the **aiFace** *face* from **mesh->mFaces[i]**.
    - Loop through the number of indices for this face (**face.mNumIndices**):
      - Add each index for the face (**face.mIndices[k]**) to the Mesh's list of indices.
- **In the main function:**
  - **The model to load will be provided on the command line:**
    - **Use "sampleModels/sphere.obj" as your default model path.**
    - If argc >= 2, grab argv[1] as the model path to load and convert to a string.
    - ***NOTE: Do NOT use cin!! You MUST use the command line arguments!***
  - **Load the model (given by the model path) using Assimp to get an aiScene, making sure to store that scene into the sceneData**
    - *Use the following options (OR'ed together):*
      - aiProcess_Triangulate
      - aiProcess_FlipUVs
      - aiProcess_GenNormals

        - aiProcess_JoinIdenticalVertices
  - **Check to make sure the model loaded correctly**
    - *Print error and exit if ANY of these are true:*
      - The sceneData.scene object is null
      - sceneData.scene->mFlags & AI_SCENE_FLAGS_INCOMPLETE
      - sceneData.scene->mRootNode is null
  - **Comment out the current code that creates hostMesh, the VulkanMesh mesh, and the list of allMeshes.**
  - **Before your drawing loop:**
    - Make sure your **VulkanRenderEngine** is an instance of **Assign02RenderEngine**:
      - **VulkanRenderEngine *renderEngine**
        = new Assign02RenderEngine(vkInitData);
    - For each **mesh** in the scene (**mMeshes** with **mNumMeshes**):
      - Create a **Mesh<Vertex>** object inside the loop
      - Call **extractMeshData** to get a **Mesh** from each **sceneData.scene->mMeshes[i]**
      - Call **createVulkanMesh** to get a **VulkanMesh** from that **Mesh**
      - Add the **VulkanMesh** to your **vector of VulkanMesh's** in your **sceneData**
  - **In your drawing loop:**
    - Pass in the address of sceneData to drawFrame():
      - **renderEngine->drawFrame(&sceneData);**
  - **After your drawing loop:**
    - Comment out previous **cleanupVulkanMesh()** call
    - Loop through all of your **VulkanMesh** objects and call **cleanupVulkanMesh**()
    - Clear out your list of **VulkanMesh** objects in your **sceneData**

Screenshot (5%)

Sample models have been provided for this assignment. To pass in command line arguments:

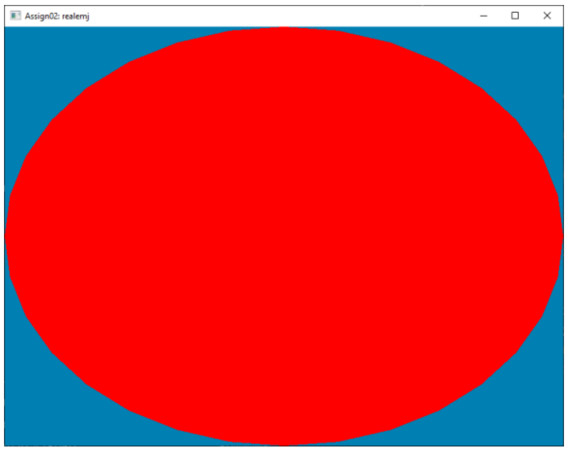
- Go to .vscode/launch.json
- Change "args" entry to ["./sampleModels/teapot.obj"],

For this part of the assignment, **upload TWO screenshots** of the application window (note that window title should be "Assign02: yoursitnetid") :

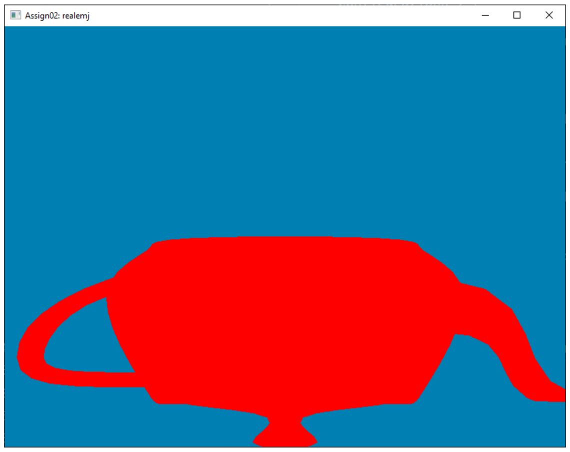
- **Assign02_sphere.png**
- **Assign02_teapot.png**

Copy the images to the **screenshots/** folder.

Assign02_sphere.png (color need not match this image):



Assign02_teapot.png (color need not match this image):



Grading

Your OVERALL assignment grade is weighted as follows:

- 95% - Programming
- 5% - Screenshot

Hints

Quick refresher on C++ vectors:

- *Add item:* myVector.push_back(thing);
- *Number of items:* myVector.size()
- *Get item:* myVector.at(i) OR myVector[i]