CS 450: Assignment 03

Setup

- **This assignment is in C++ and GLSL.**
- Copy src/app/Assign02.cpp and name it **src/app/Assign03.cpp**
- Replace “Assign02” in the application name and window title with “**Assign03**”
- Replace the name "Assign02RenderEngine" with "**Assign03RenderEngine**"
- Make a copy of the vulkanshaders/Assign02 folder and name it **vulkanshaders/Assign03**
- Modify **CMakeLists.txt** by adding the following line to the end of the file:
    - CREATE_VULKAN_EXECUTABLE(Assign03)
- Make sure the program configures, compiles, and runs as-is

Assign03.cpp

- **Add the following includes (if they are not already included):**
    - #include "glm/gtc/matrix_transform.hpp"
    - #define GLM_ENABLE_EXPERIMENTAL
    - #include "glm/gtx/transform.hpp"
    - #include "glm/gtx/string_cast.hpp"
    - #include "glm/gtc/type_ptr.hpp"
    - #include "VKUtility.hpp"

- Create a struct to hold vertex shader push constants: **UPushVertex**
    - Add one field for the model matrix: **alignas(16) glm::mat4 modelMat**

- Add to the **SceneData** struct: **float *rotAngle* to hold current local rotation angle in degrees (default value 0.0f).**

- Add the following function for generating a transformation to rotate around the LOCAL Z axis: **glm::mat4 makeRotateZ(float rotAngle, glm::vec3 offset)**
    - Generate transformation matrices (with glm) and form a composite transformation to perform the following IN ORDER:
        - Translate by NEGATIVE offset
        - Rotate *rotAngle* around the Z axis
            - **REMEMBER TO CONVERT rotAngle to RADIANS!!!!**
        - Translate by offset
    - Return the composite transformation

- Modify **Assign03RenderEngine**:
  - Override the following:
    virtual vector<vk::PushConstantRange> getPushConstantRanges() override
    - Return the appropriate vector push constant ranges
  - Add the following function for rendering a scene ***recursively***:
    void renderScene(vk::CommandBuffer &commandBuffer,
    SceneData *sceneData, aiNode *node, glm::mat4 parentMat, int level)
    - Get the transformation for the current node, which is an aiMatrix4x4: ***node->mTransformation***
    - Convert the transformation to a ***glm::mat4 nodeT*** using the aiMatToGLM4() function
      - This function is defined in include/VKUtility.hpp/cpp
    - Compute the current model matrix: ***glm::mat4 modelMat = parentMat*nodeT***
    - Get location of current node by:
      - Grabbing the last column of modelMat, which is a vec4
        - Remember that glm matrices are stored in column-major format, so modelMat[3] gives you the last column.
      - Convert this vec4 to a vec3 pos
    - Call makeRotateZ(sceneData->rotAngle, pos) to get a proper local Z rotation: ***R***
    - Generate a temporary model matrix model matrix as:
      - ***glm::mat4 tmpModel = R * modelMat***
    - Create an instance of **UPushVertex** and store ***tmpModel*** as the model matrix
    - Use **commandBuffer.pushConstants()** to push up the UPushVertex data
      - NOTE: The pipeline layout is stored in **this->pipelineData.pipelineLayout**
    - For each mesh in the NODE (***node->mNumMeshes*** meshes total)
      - Get the index of the mesh: ***int index = node->mMeshes[i]***
      - Call recordDrawVulkanMesh() on each mesh ***sceneData->allMeshes.at(index)***
    - Call renderScene() on each child of the NODE (***node->mNumChildren*** children total)
      - Command buffer, scene data: same as passed in
      - Node: ***node->mChildren[i]***
      - Parent matrix: ***modelMat*** (NOT tmpModel!)
      - Level: ***level + 1***
  - Change **recordCommandBuffer()**:
    - INSTEAD of loop with recordDrawVulkanMesh() calls, call renderScene() ONCE:
      - Node: ***scene->mRootNode***
      - Parent matrix: ***glm::mat4(1.0)***
      - Level: ***0***

- **Add a GLFW key callback function:**
    - If the action is either GLFW_PRESS or GLFW_REPEAT, check for the following keys:
        - GLFW_KEY_ESCAPE
            - Call glfwSetWindowShouldClose()
        - GLFW_KEY_J
            - Add 1.0 to *sceneData.rotAngle*
        - GLFW_KEY_K
            - Subtract 1.0 from *sceneData.rotAngle*

- **In the main function:**
    - **Call glfwSetKeyCallback() to appropriately set the key callback function**

shader.vert

- Add the appropriate push constant struct
- For gl_Position, multiply pc.modelMat by the input vertex position (IN THAT ORDER).

Screenshot (5%)

For the screenshots, you will load **bunnyteatime.glb**, which has a more complex scene graph.

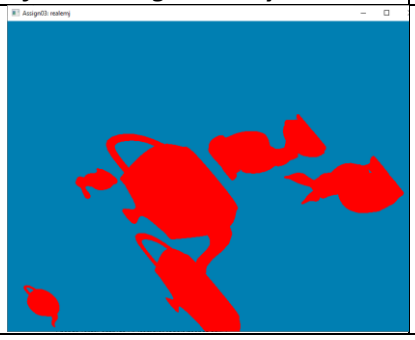
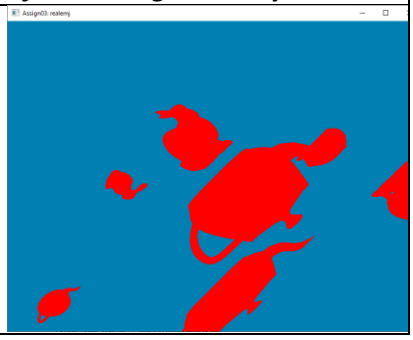
- Go to .vscode/launch.json
- Change "args" entry to ["./sampleModels/bunnyteatime.glb"],

For this part of the assignment, **upload THREE screenshots** of the application window:

- **Assign03_orig.png**
- **Assign03_afterJ.png**
- **Assign03_afterK.png**

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

Your screenshots should look like the following (barring the specific color choices of the objects and background):

Original	***After holding the J key***	***After holding the K key***
		

Grading

Your OVERALL assignment grade is weighted as follows:

- 95% - Programming
- 5% - Screenshots