

CS 450: Assignment 04

Programming Assignments (95%)

- Copy `src/app/Assign03.cpp` and name it **`src/app/Assign04.cpp`**
 - o Similar to before, make sure the shaders are loaded from the **`shaders/Assign04`** folder (instead of `shaders/Assign03`)
- Make a copy of the `shaders/Assign03` folder and name it **`shaders/Assign04`**
- Modify **`CMakeLists.txt`** by adding the following lines to the end of the file:

```
add_executable(Assign04 ${GENERAL_SOURCES} "./src/app/Assign04.cpp")
target_link_libraries(Assign04 ${ALL_LIBRARIES})
install(TARGETS Assign04 RUNTIME DESTINATION bin/Assign04)
install(DIRECTORY shaders/Assign04 DESTINATION bin/Assign04/shaders)
```

- Make sure the sample configures, compiles, and runs as-is

Assign04.cpp

- **Add the following includes (if they are not already included):**
 - o `#include "glm/gtc/matrix_transform.hpp"`
 - o `#define GLM_ENABLE_EXPERIMENTAL`
 - o `#include "glm/gtx/transform.hpp"`
 - o `#include "glm/gtx/string_cast.hpp"`
 - o `#include "glm/gtc/type_ptr.hpp"`
 - o `#include "Utility.hpp"`
- **Add a global 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(glm::vec3 offset)`
 - o 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
 - o Return the composite transformation

- Add the following function for rendering a scene *recursively*:


```
void renderScene(
vector<MeshGL> &allMeshes,
aiNode *node,
glm::mat4 parentMat,
GLint modelMatLoc,
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/Utility.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(pos) to get a proper local Z rotation: *R*
 - Generate a temporary model matrix model matrix as:
 - *glm::mat4 tmpModel = R * modelMat*
 - Use glUniformMatrix4fv() to pass in *tmpModel* as the model matrix
 - For each mesh in the NODE (*node->mNumMeshes* meshes total)
 - Get the index of the mesh: *int index = node->mMeshes[i]*
 - Call drawMesh() on each mesh *allMeshes.at(index)*
 - Call renderScene() on each child of the NODE (*node->mNumChildren* children total)
 - List of meshes and modelMatLoc: same as passed in
 - Node: *node->mChildren[i]*
 - Parent matrix: *modelMat*
 - NOT tmpModel!
 - Level: *level + 1*
- 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 *rotAngle*
 - GLFW_KEY_K
 - Subtract 1.0 from *rotAngle*

- In the main function:
 - Call `glfwSetKeyCallback()` to appropriately set the key callback function
 - AFTER the creation of the shader program but BEFORE the rendering loop:
 - Get the model matrix location using `glGetUniformLocation()`
 - INSIDE the drawing loop:
 - INSTEAD of the loop with `drawMesh` calls, call `renderScene()`
 - Node: *scene->mRootNode*
 - Parent matrix: *glm::mat4(1.0)*
 - Level: *0*

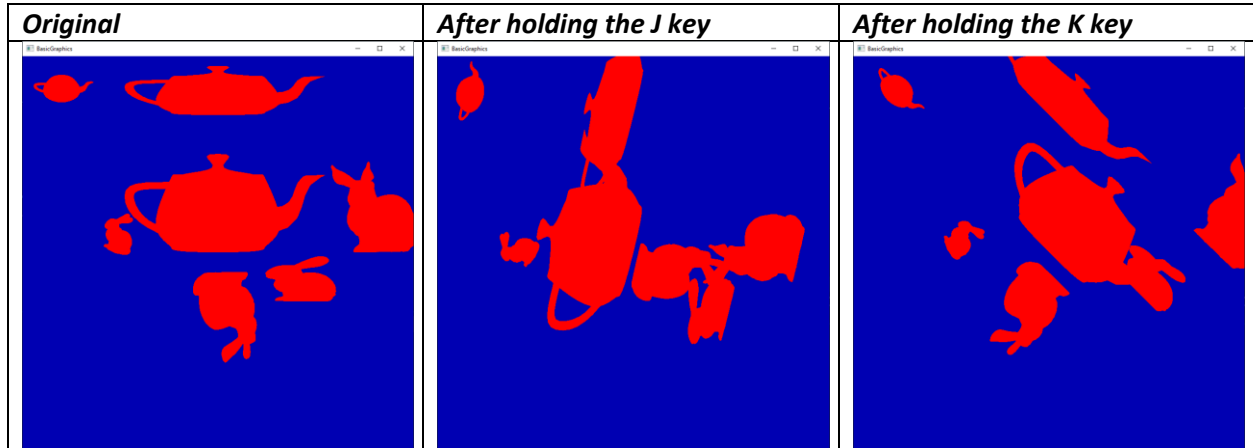
Basic.vs

- Add uniform variable *modelMat* of type `mat4`
- For `gl_Position`, multiply `modelMat` by `objPos` (IN THAT ORDER).

Screenshot (5%)

The previous models should load as before (and should rotate properly). For the screenshots, you will load **bunnyteatime.glb**, which has a more complex scene graph.

For this part of the assignment, **submit THREE screenshots** of the application window; your screenshots should look like the following (barring the specific color choices of the objects and background):



Grading

Your OVERALL assignment grade is weighted as follows:

- 95% - Programming
- 5% - Screenshots