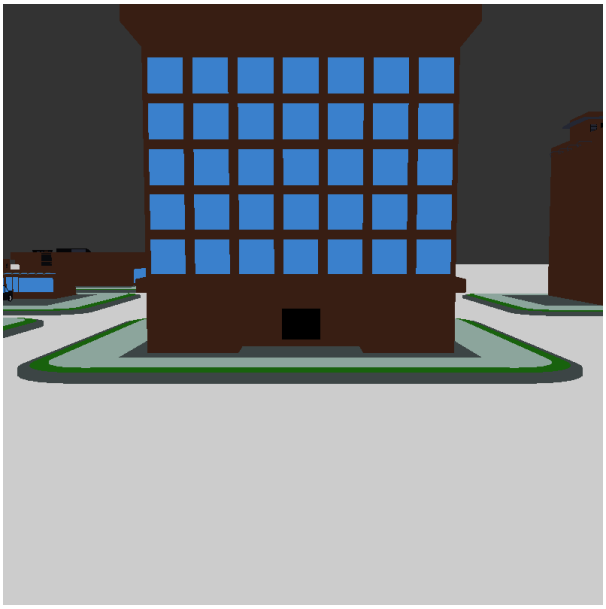
CS33400/ECE30834: Assignment #1 - Project it!

City Roaming (Camera Transformation)

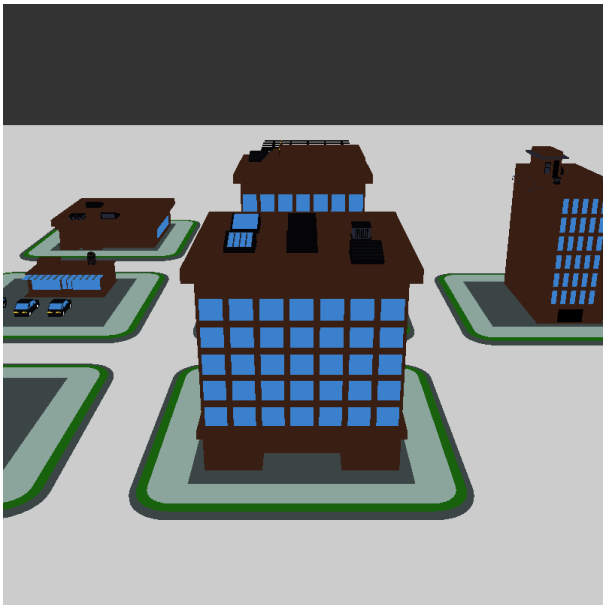
Out: January 24, 2025
Due: February 7, 2025

Objective

The objective of this assignment is helping you understand matrix transformations, coordinate systems in OpenGL and camera projection mechanisms. In this assignment we provide you a tiny city to walk in and you should be able to reach every corner of this city after correct implementation. After working on this assignment, you will learn to use linear algebra to transform the coordinates from one space to another coordinate space, and apply matrix operations to cameras for an interactive application.



(a) Ground Camera



(b) Overhead Camera

Summary

The assignment is to implement an interactive city roaming program. The program includes a low-poly city model[1]. There are also two cameras that can be switched at any time: a ground camera and an overhead camera. The ground camera simulates the view of a person walking in the city. The overhead camera gives you a complete view of the whole scene and it supports the trackball feature (you can use the mouse dragging to rotate the scene around its center).

Your first task is to compute the viewing transformation of the camera. Then you have to implement the function for rotation, and use rotation & translation functions to support the transformations of the camera, including moving forward/backward and turning left/right. Another feature is part of the trackball feature, where you will implement the scroll wheel operation, so that the camera can be moved closer to/farther from the scene.

Specifics

1. **Start with the template from the course website.**
   You may develop under Windows or Linux. For the template, see the `README.txt` for details on compiling and running.

2. **Compile and run the template.**
   Since the model is large, it will take some time to load. The figures above show what you will see when running the template (left: ground view; right: overhead view; you can press "`s`" to switch). The two camera instances `cam_ground` and `cam_overhead` are kept by class `GLState`. **Read the code to understand how it works, and then make the changes below**.

3. **View Transformation.**
   In OpenGL, a view matrix transforms the world coordinates to the view space. The function `Camera::updateViewProj` updates the 4x4 matrices `Camera::view` and `Camera::proj` constantly so the camera views can also be updated. Initially, in the function `updateViewProj`, we are using `glm::lookAt` to calculate the view matrix. Delete it before starting your implementation. We are using the function `Camera::calCameraMat` to replace `glm::lookAt`. Here in `calCameraMat`, you need to construct the view matrix using `eye`, `center`, and `up` vectors. In this function, utility functions are provided to compute normalization, cross/dot product, and transpose (you can also use corresponding `glm` functions).

4. **Rotation.**
   Implement your own version for computing the rotation matrix in `Camera::rotate`. Convert degrees to radians. Condition the result on three cases (rotate around x/y/z axis).

5. **Turning Left/Right.**
   In the ground view, the camera should be able to turn left or right. Implement this in `Camera::turnLeft` and `Camera::turnRight`. Use `Camera::rotate` (implemented in the previous step) for this task. Use `Camera::rotStep` as the rotation speed (how much to rotate between two frames).

6. **Moving Forward/Backward.**
   In the ground view, the camera should be able to move around. Implement this in `Camera::moveForward` and `Camera::moveBackward`. Use the provided `Camera::translate`. Use `Camera::moveStep` as the movement speed.

7. **Scroll Wheel Interactions.**
   In the overhead view, scrolling the mouse wheel should move the camera closer to or farther from the scene. Implement this in the function `GLState::offsetCamera`. Update the coordinates of the overhead camera (`cam_overhead`). Set two cut-offs to prevent the camera from being pushed too close or too far away. If you don't have a scroll wheel, use `I` to zoom in and `O` to zoom out.

8. **Supported Operations.**
   After implementing the above, your program should support the following operations:
   - **Mouse Controls (only in overhead view):**
     - Left click + drag to rotate the camera.
     - Scroll wheel(`I`+`O`) to zoom in/out.
   - **Keyboard Controls (only in ground view):**
     - `A`: Turn left.
     - `D`: Turn right.
     - `W`: Move forward.
     - `X`: Move backward.
     - `Z`: Move down.
     - `C`: Move up.
     - `S`: Switch between the two cameras.
9. **Debugging Functions.**
   Use the provided functions to debug your implementation: `Scene::printMat3`, `Scene::printMat4`, `Scene::printVec3`, `Scene::printMat4`
10. **Notes**
    You are **NOT** allowed to use `glm::lookAt`, `glm::scale`, `glm::rotate`, `glm::translate` anywhere in your code.
    The code lines requiring your implementation are marked "TODO"; under each TODO. there are more detailed instructions guiding you through the tasks. However, depending on your particular implementation there might be other places for you to change code. You are expected to add/modify the code as necessary to make sure the application runs smoothly.

Turn-in Instructions

To give in the assignment, please use Brightspace. Give in a zip file with your complete project (project files, source code, and precompiled executable). The assignment is due BEFORE class on the due date. It is your responsibility to make sure the assignment is delivered/dated before it is due. If you wish to receive confirmation of receipt, please ask by email in advance.

Don't wait until the last moment to hand in the assignment!

For grading, the program will be compiled on Linux and run from the terminal (with Visual Studio as a fallback – please try to avoid platform-specific code (e.g., don't `#include <windows.h>`)), run without command line arguments, and the code will be inspected. If the program does not compile, zero points will be given.

Good luck!

[1] The model is taken from BlenderKit.