CS33400/ECE30834: Assignment #1 - Project it! “Urban Roaming: Free Camera”

Out: September 1, 2023
Back/Due: September 15, 2023, 11:59 PM

Objective:

This assignment will help you grasp matrix transformations, coordinate systems in OpenGL, and camera projection mechanisms. You'll be given a small city model to explore, and your task is to control the movement of a free-moving camera within this model. By completing this assignment, you'll become familiar with using linear algebra to change coordinates between different spaces and applying matrix operations to create interactive camera effects.

Summary:

There are two cameras:

i) Ground Camera, which will be freely moved in the city.
ii) Overhead Camera (Bird's eye view): It will give you the top view of the whole city from above. It camera should be able to zoom in and out and rotate around the whole city.

These two cameras can be switched at any moment by pressing the key ‘s’, which is already implemented in the base code. After executing the program, the initial camera will be the “Ground Camera”. Pressing ‘s’ will take it to the “Overhead” state.

Specifics:

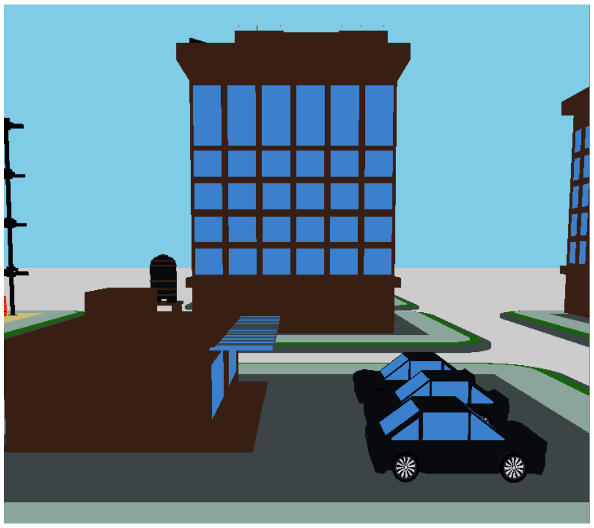
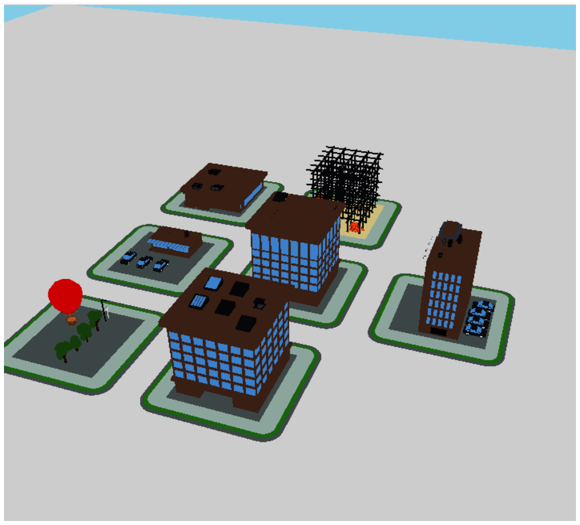
1. Start with the template from the course website. You may develop under Windows or Linux. For the template, see the README.txt for details on compiling and running.

2. Compile and run the template. You will see the ground camera. Which can be switched with the overhead camera by pressing ‘s’.
   The ground of the city is xz plane. It is your responsibility to understand the orientation of the ground and overhead/ground camera from the camera parameters defined in ‘glstate.cpp’. This is crucial for this homework.

Tasks

View transformation.	In OpenGL, a view matrix converts world coordinates to view space. The Camera::updateViewProj function consistently updates the Camera::view and Camera::proj 4x4 matrices for ongoing camera view updates. Initially, within the updateViewProj function, glm::lookAt is used to compute the view matrix. Please remove this usage before proceeding with your implementation. We're replacing glm::lookAt with the Camera::calCameraMat function. Within calCameraMat, construct the view matrix using eye, center, and up vectors. This function also offers utility

	functions for normalizing, computing cross/dot products, and transposition (glm functions can also be used).
Rotation	Complete the Camera::rotate Function to get the rotation matrix given the rotation angle and axis of rotation. You can assume that the axis will be either x/y/z axis.
Vertical Flip of the camera	In the ground view, the camera should have the capability to be oriented upside down or do a vertical flip. Implement this in the function Camera:: invert **Please note that this is not just flipping the sign of the y-component of the up vector.
Turning the camera **left/right, up/down, and tilting to right/left**	In the ground view, your camera should be able to turn left/right. Implement this in *Camera::turnLeft* *Camera::turnRight*. Camera::lookup Camera::lookdown Camera::tiltleft Camera::tiltright
Moving the camera **up/down, forward/backward**	In the ground view, your camera should be able to go up and down. And also go forward and backward. Implement this in *Camera::moveForward* *Camera::moveBackward* Camera::moveUp Camera::moveDown
Rotate Clockwise and Anti-clockwise	In the overhead view, the camera should be able to rotate around the city in a clockwise or anti-clockwise manner. Functions to update GLState::circle_anticlock GLState::circle_clockwise
Move toward/away from the center of the city.	In the overhead view, the camera should be able to move toward/away from the center of the City. You can assume that the city's center is the geometric origin (0,0,0). Function to update GLState::movetocenter **Also, ensure the camera cannot pass through the ground.**

(Ground view on the left and Overhead view on the right)

Key-Maps:

Left Arrow: Turn left
Right Arrow: Turn right
Up arrow: Move forward
Down Arrow: Move backward
D: Move down
U: Move up
R: Look up
F: Look Down
G: Tilt Camera Right
U: Tilt Camera Left
I: Invert the Camera
S: Switch between the two cameras (a ground camera and an overhead camera)
Left Click: Rotate around the city Anti Clockwise
Right Click: Rotate around the city Clockwise
Scroll wheel - move toward/away from the center

For reference, "D: Move down" is implemented in the base code.

Important Note:

We have also included several debugging functions for your convenience. You can easily utilize Scene::printMat3, Scene::printMat4, Scene::printVec3, and Scene::printMat4 to display matrices and vectors.

Please refrain from using glm::lookAt, glm::scale, glm::rotate, or glm::translate anywhere within your code.

The sections of code that require your implementation are indicated with the label "TODO." Beneath each TODO label, you'll find detailed instructions guiding you through the tasks. It's important to note that depending on your specific implementation, there may be other areas where adjustments are needed. Your responsibility is to add or modify code as necessary to ensure the seamless functioning of the application.

Turn-in:

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. You are responsible for ensuring 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!

The city model is taken from
https://www.blenderkit.com/get-blenderkit/4c00d910-5258-4205-935f-5858d3e855c0/.