CS 355 Lab #4: Simple 3D Graphics with LWJGL

Overview

This lab introduces you to simple 3D graphics by having you set up the world-to-camera, projection (perspective or orthographic), and viewport transformations. It will again use Java, but we will be using the Lightweight Java Game Library (LWJGL), which provides simple Java wrappers around OpenGL, to do the rendering.

OpenGL was originally in C, so you'll find that the design is pretty much straight imperative (i.e., commands) rather than object-oriented. It also doesn't involve GUIs, though there have been GUI libraries written for it. You won't have event listeners but will instead poll the keyboard to see if keys are pressed.

In this lab you will move around a *very* simple virtual world that consists of a wireframe model of a small house. (If you wish, you can play around with the model if you want to render other scenes.) In this world you can turn left or right in the $x - z$ plane of the world; move up or down in the y direction, or move forward, backward, left, or right relative to the $x - z$ plane *and relative to whichever direction you are currently facing*. You should be able to switch the projection between perspective and orthographic.

When moving, you should move one unit in the world coordinates *relative to the way you are currently facing*. When turning, you should turn one degree left or right.

User Interface

Your only interaction will be through the keyboard. Each of the following keys should perform the corresponding actions.

a	Move left
d	Move right
w	Move forward
s	Move backward
q	Turn left
e	Turn right
r	Move up
f	Move down
h	Return to the original "home" position and orientation
o	Switch to orthographic projection
p	Switch to perspective projection

Downloading LWJGL

You will likely not have LWJGL already installed on your computer, so you'll need to go to `http://www.lwjgl.org` and download it from there. To install it, follow the instructions at `http://www.lwjgl.org/wiki/index.php?title=Downloading_and_Setting_Up_LWJGL` for whichever type of system you have. (It runs on Windows, Mac OS X, and Linux.) Note that there is both a JAR file for the Java part and a native code piece. You don't have to "install" these per se—just set your class path and VM options as directed on the installation page, or configure your IDE to use these options. If you get stuck with this, ask the class Google group for help.

The most common cause of problems getting something to render is not installing LWJGL directly, especially setting the appropriate paths to the native code portion of it.

There are instructions on how to set up LWJGL with the IntelliJ IDEA at `http://lwjgl.org/wiki/index.php?title=Setting_Up_LWJGL_with_IntelliJ_IDEA`.

Implementation Notes

You will be given a shell again for this program. There are various incantations you have to do to get OpenGL and LWJGL set up correctly and working in an Java frame, and we want you to be able to focus just on the rendering instead of all this setup. (If you want to see the ugly stuff, it's in the `LWJGLSandbox` source file.)

For this lab the "model" in the model-view-controller sense is static. If we were interactively manipulating the model, placing virtual lights, etc., that would obviously change. Note that this model includes the `HouseModel` class but also our own `Point3D` and `Line3D` classes as part of it.

This lab uses a combined controller / view, which you implement using the `CS355LWJGLController` interface we provide. This interface has just four methods that you need to implement, as described in the following four subsections.

A stub that implements this interface and has some of what you need in it can be found in `StudentLWJGLController.java`. You'll notice that at the top of the file the TA has explicitly imported only the OpenGL commands you'll actually need for the lab. There's a *lot* more to OpenGL than what we're using, so let this import list help guide you in what commands to use.

If you don't know how to use a particular command or what its parameters are, go find the documentation on the internet. (It's not hard to find.)

A good place to start is `http://www.glprogramming.com/red/index.html`. Chapter 3 contains most of what you need to know for this lab.

resizeGL

This method will be called just once when the program opens, and it gives you a chance to set up the viewport parameters. It also allows you to initialize the `MODELVIEW` (world-to-camera) and `PROJECTION` transformations, as well as any other state data you want to keep. See the constants defined in the shell for the size of the screen.

update

You don't have to do anything in this method for this lab, but it is where you could update other data as needed. It is called every time through the main "run" thread.

updateKeyboard

Here you should poll the keyboard using the `Keyboard.isKeyDown` method (look at the documentation for this), and respond appropriately. If a movement key is pressed, you should update your state data of where you are (in (x, y, z) world coordinates) and what direction (rotation angle around the y axis) you're facing. You should then accordingly update the `MODELVIEW` (world-to-camera) transformation for OpenGL. Once you set this matrix up using appropriate transformations, OpenGL will remember it for every time you render again, at least until you change it.

If a key to switch projection mode ("o" or "p") is pressed, you should set the `PROJECTION` type and parameters accordingly. (Hint: take a look at the `glOrtho` and `gluPerspective` commands.) Once you set the projection type and parameters, OpenGL will also remember these settings for every time you render.

render

This is essentially the "View" portion of the code. It should transform the model from world to camera coordinates according to where the camera currently is and is facing. It should then draw the sequence of lines from the `HouseModel` class. You may set the line-drawing color to whatever you wish—just make sure they can be seen against the background.

Note that if you update the `MODELVIEW` and `PROJECTION` appropriately as you move around or switch projection type, all you have to do in this portion is actually draw the lines in 3D. OpenGL will handle transforming the lines, projecting them to 2D, and then drawing the appropriate 2D lines.

Extra Credit

Once you have down rendering one house, use the basic house model and additional transformations (combination of translation and rotation) to position multiple copies of the house. *You must do this by including this transformation in the* `MODELVIEW` *matrix you use to draw each house, not by changing or explicitly transforming the values of the coordinates in the model.* Use this to draw a small street of differently colored houses.

Suggestion: you might find the ability to push and pop matrices from the transformation stack to be useful. See the pattern in the slides for Transformation Hierarchies.

Submitting Your Lab

To submit this lab, again zip up your src directory to create a single new `src.zip` file, then submit that through Learning Suite. If you need to add any special instructions, you can add them there in the notes when you submit it.

Rubric

This is tentative and may be adjusted up to or during grading, but it should give you a rough breakdown for partial credit. Please be aware that it may be hard to see enough to get partial credit unless a reasonable amount of the lab is complete.

- Correct perspective projection (30 points)
- Correct orthographic projection (20 points)
- Correct camera movement (30 points)
- Generally correct behavior otherwise (20 points)
- Extra credit: Drawing a street of multiple houses (20 points)

TOTAL: 100 points (120 with extra credit)

Change Log

- May 22: initial version for Spring 2015