Warmup 1

Due: Feb. 14, 2021

Introduction

Welcome to CS1950U! In this assignment you'll be creating the basic framework of the game engine you will be developing for the rest of the semester. It will introduce first and third-person movement in a 3D world, basic application and game world organization, and some important graphics concepts. By the end of these two weeks, you'll also have a simple 3D game built on top of your engine!

All assignments in this course will follow a similar format of requiring both engine features and a game built on the engine.

Warmup 1 will get you used to working in 3D space. Though there won't always be a ton of requirements related to gameplay, this is ultimately a class about making games, so have fun with these assignments, and feel free to build more than is required! What is listed here is just the bare minimum for getting a complete on a checkpoint.

Support Files and IDE

Getting Started

If you haven't yet, read the first few pages of the CS1950U Setup Guide on the Docs page. It contains instructions for setting up a work environment on your personal computer as well as more detailed descriptions of the support code.

To get the support code for this and all future assignments, copy the contents of `/course/cs195u/asgn/engine` to your project directory for Warmup (probably something like `~/course/cs195u/engine`). This should give you a basic Qt project including a Qt pro file that configures and helps build your project, as well as directories containing some starter code and resources. This code should compile and run right away, give you a black window and a framerate counter.

You'll use the Qt Creator IDE for this course. Open Qt Creator (run the `cs195u_qtcreator` command if you are on a department machine) and open `cs195u_engine.pro` to load the project. Files ending in .pro are text files containing the project configuration (a list of sources, compiler flags, and platform-specific build commands). Here are several shortcuts you can use in Qt Creator:

- Ctrl+R: Build and run your project (standard out appears in the Application Output pane)
- Ctrl+K: Quickly open any file in the project by name (in addition to any class or function)
- Ctrl+Click: Jump to the definition of any symbol (variable, function, macro, etc.)
- F4: Switch between *.h and *.cpp files with the same name

Support Code

A major part of game engine development is being able to design large and complex software systems. For this reason, the support code for CS1950U is minimal. You will need to implement the majority of each project from scratch, so make sure to allocate time for design.

That being said, we realize you have limited time and want you to focus on what's interesting, so we have provided a few support files to get your started:

- `view.{h,cpp}`: Defines a View widget extending QGLWidget. This is a starting point for your game engine; it sets up a full screen window with mouse capture and a variable-update game loop. Every update of the game loop calls tick() to handle game updates and triggers paintGL() to redraw the view. You will want to fill in these methods when implementing Warmup.
- `mainwindow.{h,cpp,ui}`: Initializes a main window containing a View widget. You should not need to modify these files.
- `main.cpp`: Starts the program.
- `util/CommonIncludes.h`: Contains `include` statements for universally needed classes and libraries, such as glm. You may add whatever you want to it, however it will be included in many, many files, so try to keep it as small as possible.
- `engine/graphics/Graphics.h`: Contains an implementation of a graphics object, which includes various functions for drawing, and OpenGL state management. It also manages graphics resources such as Textures, Shaders, Shapes, Materials, Fonts, and Framebuffers.
- `engine/graphics/Camera.h`: A default implementation of a Camera object, which describes a view on a 3D world.
- `engine/graphics/*`: Contains other graphics helper classes which you'll hopefully find useful!

Note: when you create new folders, you may want to add them to INCLUDEPATH and DEPENDPATH in warmup.pro so you can #include files inside them directly.

Resources

In addition to the support code, we may also provide resources such as textures and models. In this assignment, the only resource we provide is a grass texture entitled `grass.png` (from `opengameart.org`). Feel free to find other textures or models while doing your projects, but keep in mind that you cannot use any associated code.

Design Check

- List the steps involved with setting up a first person camera. How will you get or compute the parameters you need?
- How will you define an application? What about a screen?
- In broad terms, describe the steps and OpenGL calls necessary to render the floor.
- How will you implement gravity, the floor, and jumping?

Basic Requirements

Basic requirements describe what you need in order to have a basic, playable demo for a checkpoint. For Warmup 1, they are as follows:

- Handin never crashes
- Your game renders a quad in 3D
- Moving the mouse pans the camera
- Pressing certain keys moves the camera

Engine Requirements

The following will all be a part of your "Common" engine. For now, you don't need to know what this means. Just know that the following features should be applicable to almost any game, as a result, should be logically separated from, but utilized by, your game code:

- Virtual Application class representing a whole game which supports:
  - Timed updates (tick)
  - Render events (draw)
  - Input events (mouse and keyboard)
  - Window size updates (resize)
  - Adding and removing screens
  - Switching screens
- Virtual Screen class representing a logical subscreen of a game which (minimally) supports:
  - Timed updates (tick)
  - Render events (draw)
  - Input events (mouse and keyboard)
  - Window size updates (resize)
- Your engine uses the provided Camera object, or you've built your own Camera object
- Your engine uses the provided Graphics object, or you've built your own Graphics object

Game Requirements

For this week, you will not implement any real gameplay. Your handin should allow the player to walk around a world using mouse and keyboard inputs to change the camera. The player will be able to jump and not fall through a textured floor, but there will be no "point" to the game.

Your game must fulfill the following set of requirements:

- The player must be able to move using standard first person camera controls, meaning:
  - Horizontal mouse movements change the yaw of the camera
  - Vertical mouse movements change the pitch of the camera
  - Standard WASD keyboard controls (W moves forwards, S moves backwards, A strafes left, D strafes right) change the eye of the camera
- The player must never fall through the ground at $y = 0$
- The player must be able to jump off the ground using the spacebar or a mouse button
- The player can only jump when on the ground
- Gravity must act downwards on the player
- The ground must consist of planar geometry with a tiled grass texture. This means the floor is a series of 1x1 quads each with the same texture, not a single quad with a stretched texture
- The game must have **at least two** screens, one of which requires player input to get to the other