CS 211 Lab 5

Welcome to C++

Winter 2021

Today we begin programming in C++ and the GE211 game engine in a minimal example game. The game is quite simple: The player controls two circles on the screen—one with the mouse and one with the keyboard—and when the two circles overlap, one changes color. As you will see, however, it comes with a bug.

Before you can get started, you'll need to install a C++ and GE211 development environment. This means you'll be setting up a C++ compiler, the CLion IDE, and the SDL2 graphics libraries.

Lab setup

Toolchain setup

Like the remaining homeworks, this lab is designed to be done on your own computer, which means you need a C++14 toolchain and the SDL2 libraries installed. Follow these instructions to setup your computer for C++ and GE211.

Project setup

For C++ projects, including this lab, the starter code is provided as a ZIP file for you to download: https://nu-cs211.github.io/cs211-files/lab/lab05.zip. Extract the archive file into a directory in the location of your choosing. Once you have your new directory containing the starter files, you can open it in CLion.

Be careful, as CLion will only work correctly if you open the *main project directory* (which has the the CMakeLists.txt in it). If you open any other directory, CLion will create a CMakeLists.txt for you, but it won't work properly.

The game

Stating a program

To select your build target, use the dropdown menu in the top right of the CLion IDE. You should select "circle_game". To compile and run code, click the green "play" button in the right of toolbar. After compiling, the game window should automatically pop up.

Broken control

Currently, there is an bug in this code. Run the program by clicking the green "play" button in the toolbar. Then try to control the circle with your left and right arrow keys, and the big circle should likely move in the opposite direction of what you intend. A bug!.

Open the project viewer by clicking "Project" on the right side of the CLion IDE and locate the code for this—hint: look in the `src/model.cxx`—and fix it. Run the code again to verify your fix.

There are also test cases for checking the model's movement, so when you are done try running your code against the tests. To build the tests, choose "model_test" from the dropdown menu and again click the green "play" button. This time a window won't pop up, but rather the test results will appear at the bottom of the CLion IDE.

Up and down

As you have seen, the circle that is controlled by the keyboard only moves horizontally right now. There are two *member functions* for moving the large circle up and down, `Model::move_large_circle_up()` and `Model::move_large_circle_down()`, declared as part of the `Model` struct in src/model.hxx. Write their definitions in src/model.cxx, following the pattern of the similar member functions. Then connect them to the keyboard by modifying the `Game::on_key(Key)` member function in ui.cxx to handle additional keys.

Be sure to run the "circle_game" code after you modify it and make sure that it works. You might get the directions wrong on the up and down keys the first time you try!

Click, not hover

Currently, the position of the smaller circle tracks the position of the mouse. However, what if we want the game to only update the position of smaller circle when we click? To detect mouse clicks, you will have to override the `Abstract_game::on_mouse_down(Mouse_button, Position)` function in the `Game` struct.

The function signature is a link to the function's documentation.

You'll need to modify the code in ui.cxx and ui.hxx to remove the `on_mouse_move()` function and instead override `on_mouse_down` with the proper arguments. After you do, run the game again and verify successful behavior.

Testing

The current tests include a few examples that should pass for your code. Fill in the two tests for moving the large circle up and down to

verify it works as expected.

There is a test case that checks that `Model::overlapped()` will return **`true`** when the two circles are touching. Fill in the final test for checking when the circles aren't touching.

Other things to try

Documentation for the Abstract Game Class.

- Make the small circle change colors when it's touching the large circle.
- Make the small circle change to a different color when it's touching the edge of the window.
- Make a circle change size when touching the other.
- Let the user change the colors by pressing different keys on the keyboard.
- Change a circle to a rectangle.