UNIVERSITY OF CALIFORNIA, MERCED

BERKELEY · DAVIS · IRVINE · LOS ANGELES · MERCED · RIVERSIDE · SAN DIEGO · SAN FRANCISCO SANTA BARBARA · SANTA CRUZ

CSE 165: Introduction to Object-Oriented Programming
Lab 5

In this lab you will be creating a prototype for a Space Invaders type game, where you will be putting together many of the concepts covered so far. At the very least, your app should have a shape representing a spaceship (a simple rectangle is sufficient for that), that can move in 4 directions on the 2D place, namely up, down, left, and right. You may assign the arrow keys on your keyboard to each direction or you may use the W A S D keys.

In addition to your ship, there must be at least one enemy, which can be represented by a simple rectangle, and it may be stationary. Ideally, the enemy would be in the top half of the screen, and your spaceship at the bottom.

Your spaceship needs to be able to fire a projectile, when a specific key is pressed (spacebar is a good candidate). The projectile should originate from the position of the ship, and should travel upwards in a straight line.

As the projectile moves upwards, it will either colide with the enemy, or it will fly off screen. If your projectile collides with the enemy, then the enemy is defeated, and it should disappear. A message "YOU WIN" should be displayed on the screen.

If your ship colides with the enemy, then you lose, and a message "GAME OVER" should be displayed on the screen.

This is a very heavily watered-down version of the game. Basically, one stationary enemy with collision detection is all that is required. If you want to do some coding for fun, there is a large number of features you can add, such as multiple enemies that move and fire their own projectiles at your ship (or they just fire randomly in multiple directions), a points counter, limited lives, power-ups, etc.

At this point everything on the screen (spaceship, enemies, projectiles, etc) can be represented by a simple rectangle. Next week, we will see an easy way of attaching textures to your rectangles, which will allow you to easily create sprites, and add backgrounds to your apps.

As for the implementation side of things, you are required to use the new GlutApp template (included in the Lab 5 folder), and you have to make appropriate objects for everything. At the very least, you should have a Game object that contains everything pertaining to the game. In your `App.h` file, you should declare an instance of your Game object, and in the `App.cpp` you should initialize it correctly, you should display the game in the `App::draw` function, and you should pass the keyboard events received from the user, onto your Game object, which should handle everything. There should be no other logic in your `App` files.

When implementing your game, you should also break it up into meaningful components, for example we would like to see an object called `Ship`, that has a `fire` action, along with all the other appropriate things.