Section 1: Your First SFML Project

- This section, along with section 0, was arguably the most difficult part of the project. I initially tried to set up my environment using Multipass. I received several errors with this and eventually decided to switch to using NCSUs VCL with the help of a friend. I received several library errors when I first started coding my main file. This troubled me until I discovered I needed to download the SFML libraries and have them sit in VS code.
- After much trouble setting up my environment, I was finally able to start programming my window. I followed the SFML tutorial exactly as stated on the website to open a window. After this, I still faced a difficult time compiling my code since I was getting several permission errors accessing my files. I tried using “chmod -x main” and running main after and would still experience errors. It was not until after experimenting and researching several commands I was reminded to use “make clean” which helped me resolve my issue.
- At this step, I was told I should have a black window screen, but for some reason, my window was clear and would look very glitchy if I tried to resize it. I looked at several SFML documentations, and switched a few things such as using “RenderWindow” instead of “Window” and I also learned I needed to use “window.display()” and “window.clear(sf::Color::White);” which is what I didn't get from the initial tutorial but found later and eventually helped me display my window.

Section 2: Drawing Objects

- For drawing objects, I experienced minimal issues. I first wanted to test my ability to spawn a basic circle shape in my main file. After this was successful I realized these shapes were going to represent all my objects in the game. I wanted to create an abstract class that would be implemented by my character class and physical object class. I realized that I had no idea how to do abstraction in C++ and decided to start with a character class and go back to it later.
- I knew my character class would need to take on a shape and some position/ spawn point at its very least. I also wanted to simplify my draw() by incorporating a draw() method since I would already be taking in the shape. This worked very nicely when executing this in my window. Secondly, I decided to add a new argument to my constructor to take in a texture for the character. If the filename was unavailable then I would default the shape to being all red.
- For creating my game object I copied my actor class since I wanted to get all the same information from it. I probably could have made them the same but for the sake of future-proofing, I thought it would be better to have them separate.
- When I eventually created my abstract Object class, I made my GameObject and Actor class inheirit the constructor, draw(), getShape(), and move() which I will explain in section 3.

Section 3: Handling Inputs

- Section 3 was probably the easiest section for me to complete for this homework. I basically followed the code given in the move tutorial. I initially had these movements called in my event handler, but realized I could achieve smoother transitions outside.
- However instead of using move() SFML gives, I decided to create a move() function in my Object class that would then call move. I created this method for when I wanted to handle physics later on. Right now my move() function uses setPostition() to set the new position of the object to see if that is better using velocities but I may switch back depending on performance

Section 4: Collision Detection

- For my collisions, I thought it would be smart to implement what could be gravity later on. So I wanted to add an isTouchingFloor() function. This took a lot longer than anticipated because I wanted to get the lower end of my spirite's hitbox so the top or side of it would not count as it touching the ground. I thought it would be fun to test this by displaying the object's hitbox, which will appear in my shape by pressing H. This function currently compares against one object but I would like to redesign it to compare against any object without doing that manually in a loop. With my isTouchingFloor() my character transforms downward until its lower bounds intersect with another shape.