CSC 581 - Assignment 1 Reflection Writeup
Ashwin Sapre (asapre)

Services

The main game loop has the following services:

- HIDS (Human Input Detection Service): The game polls for keyboard actions, and on the basis of the detected actions performs predefined game actions such as moving the ball to the left/right/up. Only one event is accounted for, the “close” event.
- Movement service: This service is responsible for moving all of the moving parts on the screen. There are two kinds of moving parts; the ball, user controllable; and the moving platform, automated.
- Collision service: Handles collisions between the ball and the surfaces and the walls.
- Render service: This service renders all the objects present in the game.

Separating all these services will help grow the game later on, as it becomes more and more complex. All these services interact with the game object (since the game object contains all the other artifacts)

Class Design

Game class:

The game class contains the sf::RenderWindow object and all the fixed parameters of the game, such as the dimensions of the ball, surfaces; their initial positions and speeds, etc..

Packaged inside the game class are game objects; the ball, surfaces, walls and their textures. This kind of packaging eliminates the need to pass every game object separately to each service. Instead, we can just pass the game object. This also makes the process of adding new components to the design much easier; it is sufficient to add them to the game class.

The game class is instantiated as a static object. Texture loading, shape initializations are done inside the init() function of the game class. Pointer cleanup occurs inside the shutdown() function, called when the user quits the game.

Circle class:

This class is used to create circular objects by extending the CircleShape class. Only one circular object is used for now (the ball).

Rectangle class:

This class is used to create all rectangular surfaces and walls.

Time class:

This class was supposed to be used for implementing more complex jumping and gravity systems, however I did not find time to do that for this assignment. This class, therefore, serves no purpose currently.

Assets

Textures used in the game (for the walls and the ball) were downloaded from opengameart.org.
Ball: https://opengameart.org/content/pastel-colored-balls-spheres
Walls: https://opengameart.org/content/brick-wall-0

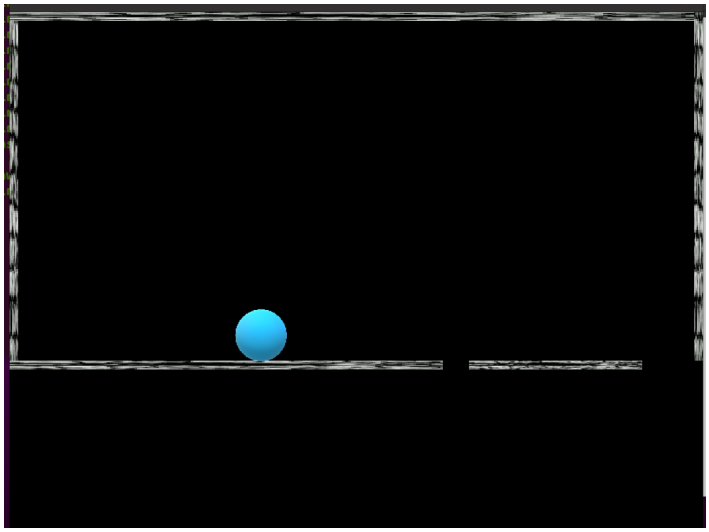
Gameplay

- The game has two window resizing modes - proportional and constant. These can be toggled using the LShift key. Screenshot 1 shows the full 800x600 view. Screenshot 2 shows the “constant” mode view; the window has been resized to a smaller size and as a result, some parts of the game have been cut out of view. Screenshot 3 shows the “proportional” view mode. Starting from the initial 800x600 size, the window has been resized to a smaller size. All of its contents are therefore scaled proportionally.
- When the ball falls down, its position is reset to the starting position. This feature can be extended in the future to add a “lives” component (similar to Super Mario)

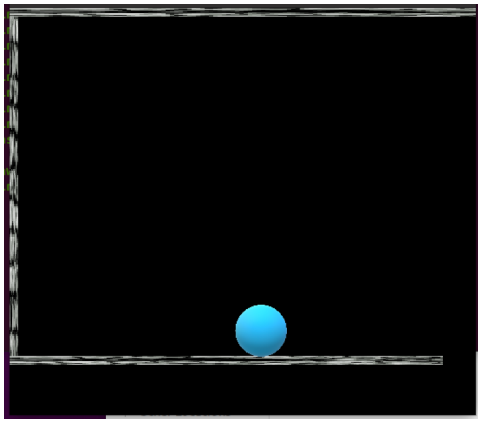
What didn’t work out:

I wanted to make jumping and gravity a bit more sophisticated for this assignment. Currently jumping looks and feels jolty (because, the way it is implemented is just a large upward movement when the up arrow key is pressed. This had to be done because with smaller values, the ball just wouldn’t rise off the floor). This will be improved in forthcoming assignments. I plan to use time to “disable” gravity when a jump has been initiated so that smaller jumping velocities can be used, which in turn will make jumping feel smoother.

Appendix
Screenshot 1



Screenshot 2



Screenshot 3