CSC481 Homework 1

Casey Byrnes

9-13-2023

Design Decisions

My first design decision was to create a window to run the game in. I decided on a window size of 800 by 600 pixels, that way it can run on any modern monitor resolution. I also decided that it should open in the upper right corner of the screen every time the game runs. This ensures that the window will not be off-screen when opening it. I chose to incorporate a close button into the title bar as well so that a user can easily close the window. I currently have the frame limit set to 60 fps, but this will likely change once I implement timelines. Many movement functions are tied to the frame at the moment, so a limit was needed, but moving forward with timelines this may not be needed.

My next design decision was to create platforms. I created a static platform class by extending the RectangleShape from the SFML library. In the constructor for it I added many extra arguments that will be useful when making a platform, including color and a start position. In the future I plan on adding some additional functions to this class to allow the platforms to move left or right when needed, since this is for a platformer and the level may extend beyond the limits of the game window itself. By adding this function, I would be able to move the platforms left or right when the character reaches a certain point in the screen and give the player the perception of the world moving left or right, which is common in platformer games.

I then made floating platforms by extending the newly created platform class. This allowed me to use the additional arguments I made for platform, but I also added more arguments to allow for direction the platform moves, the distance it moves, and the speed at which it should move. After I implemented this the way I did, I thought that it may have been easier to have the floating platform extend the RectangleShape, then have the static platform extend the floating platform class. I will likely change this in the future because I think it will make more sense going forward. I also intend on adding the same left/right movement to this class as the static platform class for the same reason. Since these will be a part of the level itself, they need to be able to shift left and right when the world “slides” right or left.

After designing my platform classes I designed a character class that extends from CircleShape. I chose the circle for the character to help distinguish it from the rectangles of the platforms and I also added a texture to it that is a little face with a white outline around it. This helps the player know further separate the shape from the platforms and other future objects in the game. The character class uses many other variables to track things for the character, such as jumping, falling, jump height, etc. I also implemented functions to allow for jumping and other movement, as well as checking if the character is falling.

Once I had these three main shapes, I was able to create them after the game window, but before the game loop because these objects need to be tracked outside of the game loop. Inside my game loop I start by checking for any user input. I used the sf::Keyboard::isKeyPressed() function to get real-time inputs from the player. At the moment only four possible keys can be pressed. Although the game can be closed with the “X” button in the title bar, I also wanted to be able to close it with a

keyboard key so if “Esc” is pressed, then the game closes as well. The other options that currently exist are “A” for left, “D” for right and “Space” for jump.

After checking for player input, I attempt to apply gravity to the character. I do this by using the check for falling function I made in the character class. If the character is falling then gravity is applied, if the character is in the process of jumping, however, then gravity is not applied.

For collision detection, I am currently checking for collision with the character and each of the other objects in the game (the static platform and the floating platform). To check for each of these I am comparing the bottom of the character with the top of each of the platforms. If gravity pulls the character down into/passed the top of either platform, then the character is moved to the top of that platform. This only happens if the character has an X-axis position that is within the platform. Once the player has intersected with it and been moved to the top of the platform as needed, then the player is marked as “allowed to jump” by using one of the created functions for the character class. By controlling the character’s jump ability this way, I am preventing the player from being able to jump in succession without ever touching the ground/platform. One thing that is not working as I would prefer, is the floating platform collision. The character is able to stand on it without issue, but if the character is under it while the platform is low enough, the character is automatically set on top of it (See Figures 1 and 2). After hours of troubleshooting, I believe this is due to the use of the “intersects” function and I plan on changing this in the future if I am able to find a better way to handle this collision scenario.

As a part of the collision detection system currently, I am also checking the bounds of the window itself. If the player tries to move past the window, then collision is triggered and the character position is set to the edge of the screen. This will also change in the future if I am able to implement a level “sliding” system that allows for larger levels. I plan on doing a collision check more in the middle of the window and if the character passes that point in the X-axis, then the level will shift left/right as needed.

After everything is ready, the window is cleared on each loop iteration, then each of the objects are drawn and the window is displayed.

I tried implementing a level class that would allow me to simply use ASCII characters and load up a level design with a different ASCII character for each object in the game. For example, a “.” Would mean nothing, a “#” could be used as a static platform and a “P” could be the player start location. I struggled with figuring out how to implement this and create the actual objects. I think more characters will be needed to indicate more variables to actually create the objects. I will revisit this though because I think it would make the game engine much more useful, especially if I can successfully implement the expanded level “sliding” mechanic I want to implement. I am still trying to figure out the design of it, but I think that using numbers for floating platforms would work, and allow me to have start and travel distances (See Figure 3). This still would not give me any custom colors though, so I may need to change either the import system, or the platforms themselves to just use a default color.

In the future I also would like to implement a running mechanic. After much thought on it, I think I will try to use a velocity modifier when the “Shift” key is pressed and then revert back to default speed when it is released. Because of the current movement being tied to the frames, I decided to wait on this until I get proper timelines enabled because I think if I implemented it now, I would end up needing to refactor it anyway.

Appendix

Figure 1:

Figure 2:

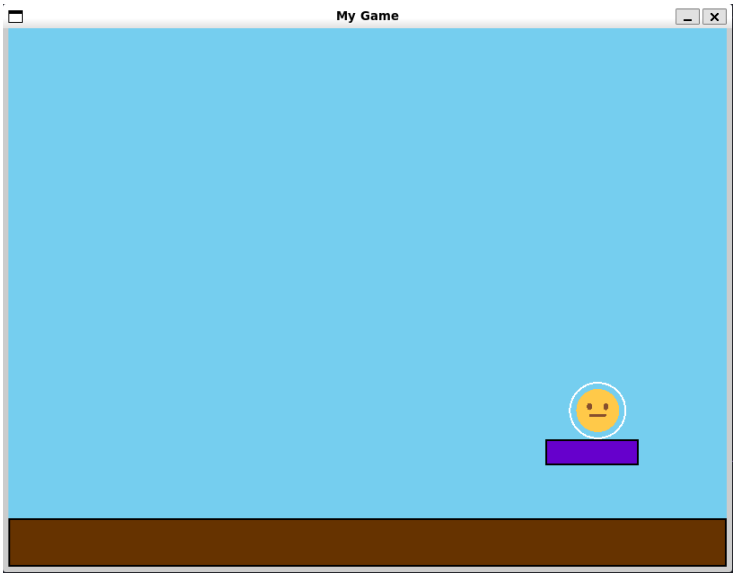

Figure 3:

```
..........
..111..2..
.P.....2..
##########
```