Justin Mabry

Making a Side-Scroller

For my final project, the application I chose to create was a 2D, side-scrolling game reminiscent of games from the past. I've always been interesting in game design and have previously created games in c# and Java, with the latter being a game based off of the famous game Oregon Trail. The former however, was much more similar to this with it having a window with GUI elements and sprites. It was a very different experience then any other programming for games I've done with it being more similar to how we made applications in class. With most of the applications I've made, particularly the ones that involve GUIs, there is an editor window where you're able to place and manipulate sprites and other objects to give it the looks you want before you get to programming the program, however, making window-based applications in Python is very different.

Every single part of the window, game, and sprites, and had to programmed in without the use of a GUI editor, in the beginning of me making the game, the background, ground, walls, and every other part of the game was coded to be manually created every time the game would load, that also meant I had to program the collision for every single one when it was being created. Fortunately to streamline this process and to my joy, there's an open-source application for Python Arcade that allows you to design and edit maps, as well as give them different layers before implementing it into the game, then all you have to do is take the name of the layers and specify them inside your application so you know what to look for when programming the games collision. With changing this it also made me change the way those objects are read in the program. Every time I changed that, I had to change the spatial hashing for the objects to make sure there were no problems between the player and the other sprites on the screen.

I went through multiple phases of maps and because of this, the final map that I'm submitting isn't the only one that there is code for, meaning if I chose to load in another map, I'll have the options for that map already programmed in. Python Arcade allowing me to import a map I created didn't make anything easier though, it more so just changed the way I had to program the rest of the game.

There were a lot of challenges while programming this. Although at first, it's similar to the solitaire game I made for the project earlier in the semester. It was a completely different process with the only similarities really being the fact that both games are built inside of a main game class. One of the biggest challenges for me was the camera and movement. Although it was somewhat similar to how it was done in c#, the movement went through many different iterations during my development of this game. Everything was initially housed inside the key press and key release functions, however once I implemented animations into the game, I had to completely rewrite the way that movement was handled within the game. Another challenge for me was actually the animations mentioned above. When I started implementing animations into the game, much of my code was once again changed. Initially everything for the character had all been housed inside the same class that my game was. However, when I wanted to implement the animations into the game, I had to create an entirely new class just for the player character. The movement for the character is still housed within the game class, however, the player class is where the textures were loaded into the game, as well as the majority of the code for switching between the animations and knowing which ones to switch to.

While testing my game, another thing that was missing was the music and sound effects, games are fun and all, but without music and effects, the game will almost always be lackluster unless it takes a truly unique idea and transforms it. My first objective in this area was to setup the sounds for when a collectible was picked up, when you died, or when you jump. After those were put into place, I found some background music to play in the background while the user is playing the game, once again I ran into a problem while troubleshooting. Initially, when the program was running everything worked as it should, however, when I closed out of the application the music would still continue playing even though the music wasn't running anymore. At first, I wasn't sure why this was happening, but from what I could find online it had to with the fact that the music was still being played through the code in Spyder. I'm not sure if it would still have the same problem if you were running it straight from an executable because you wouldn't have Spyder running the background, but nonetheless it was something that needed to be fixed. I found a workaround by importing the OS framework into my project and creating an IF statement inside my *main* method at the end of my code where if the application was completely closed out, then I would have the music stop playing.

Overall, it was a very interesting project that has me excited to try making another game inside Python to further see what the framework is truly capable of. Something I would like to try implementing in another game is something that I have working here, but with the map I've given it's not able to be properly shown-off. Cameras are something that I had never worked with before this project and it's still something I'm not sure I fully understand, but there had to be two different cameras put into place. One for the character, and one for the score that you can see at the bottom left of the screen. The camera is what handles what happens to the screen when the character moves to somewhere off the current view of the map. With bigger maps I had used while making the game, it worked fully as intended and would pan to where you could see what the player is moving to. You can see it when you move towards the right of the screen in the current map after starting off, but it just doesn't show much due to the design of the current map.

Although there are other parts that aren't implemented in the map, I chose this map because I think it's the best example to show what I've learned while working on this program since it has many of the elements I had to learn while I was making it. I feel like the project I'm submitting shows off much of what I've learned while working on the game, however much of my initial code and the code that was shown off during my presentation has changed. I'm going to submit the one from the presentation as well as the final so you can get a better understanding of how much of the code was changed, however, I would like to add that there are still a couple hundred lines of code that was either modified or taken out through this programs evolution during my time working on it.