Kale Wicks

12/9/19

CRCP 3305

Final Project Summary

My project is a small, prototype build for a platforming game. The game allows the user to interact with it in three ways: the start screen where the user may choose to start the game or exit (the options button is currently disabled for this stage of the code); an exit screen once the user reaches the end of the place holder stages, and mostly through a small sprite representing the player character. The user moves the player character using the left, right, and up arrow keys to move and jump.

The code is organized through a number of separate classes. The PlayerCharacter class where the methods to move the player character and allow it to stand on platforms. The Stage abstract class which holds basic methods all other stage types will need such as display and transition. The TutorialStage class where the player will gain access to the player character after pressing start. This class exist to allow three screens to teach the player the controls. The DayZone class where the first stage of the game will be played. The StartScreen class which holds the code allowing the game to progress smoothly by linking the classes and their methods together. The Platform class allows for the main focus of the game, providing a rectangular platform for the player character to move and jump on.

The main libraries I used were the PImage, PFont, and PApplet libraries. The PImage class allows for the user to be able to choose which sprite they want within the code. The PFont class allows for text to be displayed to convey messages to the user, such as how to play in the Tutorial stages, where they are in the game, and display text for the start and end screens. The PApplet class provides access to a number of methods such as the keyCoded functions and the width and height of the screen.

Some of the main problems I had while programing each came in the form of difficulties with what I was currently trying to implement at that time. Starting with a plain screen with a rectangle used as a ground constant, my first problem came in the form of the player character continuing to move in the last pressed direction after a direction had been pressed, creating a sort of sliding functions. Another problem came in the form of the jump function. First the player character would not come down after jumping, would continue to move upwards and stay at the top edge of screen. Other problems came in the form of getting the screens to display or having stage text fade out when needed. Some of the problems are documented within the Project Journal, but as the journal was not consistently kept, it does not list every problem had during the development.

If I had more time I would fully implement the random platform generation function, as well as attempt to specify where platforms can be generated so as not to create platforms that harm the user’s gameplay. I would attempt to also clean up the code, as I suspect that once every stage type is implemented it will allow for a method to be implemented which will transition from type to type, connecting each stage rendering the current way of coding to be cleaned up and take less space on the computer. If I could go back, I probably would have attempted to code the platform class after the player character class, rather than trying to get the stages to work. I believe, that would’ve allowed for a smoother and more cohesive coding process due to each stage type class needing it’s own player character and platforms specifically. Overall, from this project I learned the importance of inheritance as well as the usefulness of abstract classes to allow for quicker and more compact coding.