DAREDEVIL

Stefon Martin

11/26/2018

*Note: the main objective of documentation is to provide a guide for someone that is taking over development of your project

1. Introduction

The minigame daredevil has reached a phase where it has a full system in place to run basic gameplay. Players now have a game where there is a sense of thrill and faced paced thinking. Future developers have a game that is fairly easy to read, optimize and add more creative features.

2. User Interface

The initial start of the game begins with the main menu. The main menu has two option play and settings, settings currently only has the option to tune the volume. After play has been selected you will start daredevil's gameplay. At the top left of the screen the player can see the score change in real time as you are sky diving, at the top right lies the players best score. The best score is locally stored on the computer you are using, later we hope to be able to saves scores on a server, so each player can know their high score on any machine. Players of the game have the initial ability to move left or right to initiate the "sky diving" phase. In the sky diving phase, the player is still only capable of moving left or right but with a slightly faster move speed to reenact movements of high agility to dodge the flying objects. The scoring system is based on the elapsed time that the player has started skydiving until the player runs out of lives. Three lives are initially given and are reduced when the player runs into enemy birds. There are two additional items that float throughout the sky and are used as perk. The first item is the active coin which is a rare item that is used to add to the incremental active coin. This will hopefully affect the amount of time spent of the game because the player will want to play focused and long enough to get the coins that pop up infrequently. The second item is a perk that is used to manipulate the level so the player can have a better chance at surpassing the high score. The game difficulty and speed of objects increases as the player survives longer, this gives a bit of variety for the game so that the gameplay isn't completely vanilla.

3. Components/Resources/Textures

As explained in the previous section daredevil has only two scenes. The main menu and the actual game. The game scene has a background that scrolls to reenact falling through the sky, the scrolling doesn't begin until the player has moved off the platform. With prefabs you can store game objects that you have added components and scripts so that you only have to drag and drop to your scene. The prefabs stored are the enemy birds, the scrolling background, the player and the pick up items that float throughout the level. In the resources folder the gameplay music is stored, this is the folder that should hold all of the sound effects and miscellaneous files.

4. Code

The backbone of the game relies heavily on the scripts. Scripts are used to manipulate game objects and it incorporatess lots of object-oriented programming. The main script being used to collect information about everything happening in the game is the Game Controller script. This script can be used as an api to send information to the incremental. Things like the score, high score, difficulty, camera and coins collected are stored. The player object is controlled by the ddPlayer script, this script is responsible for mostly controlling the player and detecting collisions. When the player layer collides with a game object the Collison method determines what effect will happen to the player. Each enemy object is being controlled by the flying bird script which just uses an algorithm to use force upon the rigidbody of the game object to send it flying upward. The different birds all have individual speeds and spawn rates to add a bit of variety to the game play. Each enemy and item object gets used as a variable in the spawn script to determine the location and spawn rate of each object. The background is split upon six different sections and each are using the same BGScroll script which is the script that reenacts the player falling. The script takes the rigidbody of the background and forces it upwards until it reaches a certain maximum Y coordinate which after, gets set back to initial position, continuing this in the update loop makes the sliding background.

5. Learning outcome

This project interested me because it had to do with game development and I have always enjoyed creating games in my free time. I took it upon myself to use this project as a learning experience for how to properly do that. Throughout this project I have learned several things that contribute to me being a better programmer. Planning is a fundamental protocol before jumping into developing a game. With planning you are creating yourself a framework to work from and it will help keep things organized and working properly in the end. I began with drawing a visual illustration of how the gameplay would look and from there I could make a diagram of all the components that I would need to make the gameplay work. Along with planning I learned how important it is to document and clean your code. Simple things like commenting what a function does goes a long way when someone else must read your code or when you are trying to remember what everything does. Even things like unused variables can become a nuisance to deal with because you are trying to figure out how everything is being pieced together. When I get hired by a company to be a programmer I will most likely be given a frame to work off of someone else's code, this project has given me the experience to be able to read someone else code and to be able to add new features and fix bugs. Staying updated with the code is also important because you must have knowledge about how the system has changed and how it will affect the things you do for your code.