DL836 - Y2 Software Project

CA 1 – Casual Game

Paul Doyle - N00193346

Design:

I designed my game to be a top down shooter, taking games such ‘The Binding of Isaac’ and ‘Enter the Gungeon’ as inspiration. The player will have a bird’s eye view of their playable character and environment. The player will be able to move their character on both the x and y axis. The player will have a map of the area that they will be able to view, but they only the area’s that the player has visited will be visible on the map. There will be enemies throughout the environment, these enemies will track and attack the player on site. To defend themselves, the player must shoot and destroy the enemies. The player can shoot on the x and y axis but can only shoot a certain number of bullets at a time, so they must aim their shots carefully to make them count. It will also be possible to avoid enemies by running past them by sprinting, but these enemies will continue to follow the player into the next room and the player risks becoming trapped and overwhelmed by enemies. When the player is hit by an enemy, they will lose health. If the player’s health reaches zero, they will lose the game and be brought to a game over screen. There will be health items hidden through the environment, which on contact will restore a portion of the player’s health. The player must reach the spaceship to reach the end of the level and win the game. Upon doing so, they will be brought to a winning screen with their score displayed. The aim of the game is to get the highest score possible. There are multiple routes a player can take to reach the end of the level. The player’s score will increase every time they shoot an enemy, so to gain the highest score, the player will have to face all the enemies and explore the whole environment. This make the player choose the more difficult task of trying to destroy all the enemies instead of trying to find the exit.

Implementation:

I first started the game by creating a test map, which I would then build upon once I had the basic functionality of the game working. I created this map with a program called Tiled. I first searched for usable assets on sites such as Open Game Art and Itch, from which I could use to build my map. After finding a tile set that I felt suited my vision for the game, I began creating the map. I first created 2 separate layers, a floor layer and a wall layer. The floor layer did not need any properties as the player would just walk on top of it, but I needed to set a collide property on the tiles that made up the wall layer, as I would later need to create a collision between the player and the wall layer.

Once the map was created, I created a new phaser project with a template and began coding. At the beginning of this CA, I greatly struggled with the coding of this game and I still do in some areas. I did not struggle with the aspect of coding or the logic involved, but rather I struggled with the JavaScript syntax as there has been very little JavaScript covered in any of our previous classes or subjects. I felt I may have wasted a lot of time at the beginning of this CA by taking JavaScript tutorials unrelated to this project to get myself somewhat familiar with the language, but also felt that it was necessary.

I started the coding by first creating a boot scene. This boot scene would be used to import all the assets used in the program. I then used this scene to import the JSON file exported from Tiled and the .png file of the tile set used to create the map. Once these files were imported, I created a Game scene which would be used to for the gameplay element of the game. Before creating anything in the game scene, I created a scene start method in the boot scene, so once the assets were loaded, the game scene would start. Now working in the game scene, I created the map with the tile map function. I created different layer and gave the wall layer a physics collision property. I now had to create a player character. I created my own sprite with a character generator and imported its sprite sheet into the boot scene. I then created a create player function, which created a sprite and added physics to it and then added a physics collider between the player sprite and the wall layer. This would mean that the player sprite would not be able to cross the wall layer. I create some input keys in an update scene which allowed me to move the player sprite so I could test the collider. I did not get this function first time, but after reworking some of the code I managed to get the collider working.

Now that I was able to move the player, I decided to animate the sprite depending on what direction it was moving. I did this by creating animation sprites for each direction and by starting an animation loop on certain frames of the sprite sheet. Each direction was then assigned to its corresponding direction. I then implemented the first iteration of shooting mechanic. This was done in the update method, I imported a bullet .png in the boot scene, then in the game scene, created a bullet would appear at the player's x and y position and move corresponding either on the x or y axis depending on which direction to shoot was pressed. While this iteration shot a bullet and had physics, the bullet would not disappear after colliding with an object and would hang around in a map so was not useable. I added an enemy sprite and wrote code to make the enemy track the player. If the player's x position was greater than enemy's x, the enemy would move right towards the player. I used the same logic for less than x and both greater than and less than y. When all these if statements were combined, it would make the enemy follow the player. I then animation states to the enemy depending on which direction it was moving.

To get the bullet working properly I need to create a bullet class. This class contained the parameters necessary to create and move the bullet in any direction. It would also make sure that the bullet would disappear after a certain amount of time. In the game scene, I then created a function to add physics to the bullet so that it would collide with the enemy and timer for the last time a bullet was fired so that the player could not fire multiple bullets at a fast pace. I made sure that when the bullet collided with the enemy, the enemy would lose health and when its health reached zero, it would disappear. Now that I had working bullets and enemy, I need to create more enemies. To do this, I created an enemies class using all the same properties as the function, but this allowed me to create multiple enemies without having to rewrite the code for every enemy. Once I had multiple enemies working, I created a new bigger map with Tiled. I created multiple paths for the player to take and multiple room. With the map imported I created multiple enemies and placed them around the map in separate rooms using the enemy class. I then made different functions for the enemies which took care of collisions between them and the bullets as well as them and the player. I made the enemies lose health whenever they collided with bullets and that they would disappear when all their health was depleted. In the update method, I created a statement for the groups of enemies in each room, so they would not start tracking the player until the player entered the room.

I then created the map/zoom function, by having the camera always zoomed in to the player. If the player wanted to view the map, they would have the option to press a key to zoom out and show a larger area. To hide the areas of the map that the player had not been to, I created masks for each of the rooms. I created these by importing an image of the map into Photoshop and drew black rectangles over each of the rooms. I then imported these squares into the scene and wrote a function so that the squares would disappear when the player collided with them. I ended the end point of the game, by having the game end and start the win scene once the player collided with the spaceship at the end of the level. I created a game over scene which would appear once the player had lost all their health from colliding into them. I then added health items throughout the level, which would refill some of the player's health upon collision. I finished by adding a score to the game, which would increase whenever a bullet hit an enemy.

I then spent the rest of the project creating the different beginning scenes such as the main menu and options menu. I linked all these menus so that the player could navigate through them all.

Testing:

I was constantly testing this game myself throughout its development, keeping an eye out for bugs. When I had the basic structure of the game working, I would ask my friends and classmates to test the game to make sure that the controls were easy to use, and that the objective of the game was clear. This was a good way for me to receive feedback and have information pointed out to me that I may not have noticed as I was working so closely to the project. I would spend a lot of time using the debugger in the web browser as hints to fix bugs such as music not playing correctly, or enemy collisions not working correctly.

Conclusion:

Overall, I enjoyed working on this CA. It was a good experience in problem solving and thinking in terms of what does and does not work when creating code. However as said previously, I wish that we had a bit more practise with using JavaScript before taking on this CA as there is a lot more that I would have liked to implement into the CA, but is beyond my capabilities due to lack of experience and time. If I had more time, I would have implemented more features in the game, such as multiple enemy types, multiple levels, and a boss enemy at the end of every level. I would have also liked to create more sprites and filled the levels with more background elements. While this game is not perfect, I feel it is a good attempt for my first game and what I've learned throughout the process will greatly benefit my future programming projects.