**Halioth - Computer Studies Summative Written Report**

**Requirements**

For the first stage of the software development cycle, my partner and I discussed possible ideas for the kind of game that we wanted to program. We made a Google Docs file and shared our ideas with each other that way. Ideas included mazes, ball shooting, and scrolling platform games, but a space exploration game was decided upon.

Next, we considered expanding on specific details on the game, especially its objective. One of our ideas included dodging objects thrown at various speeds, while another involved firing at asteroids under a time limit. Eventually, through a fair compromise, the two ideas were merged, and the final objective that was decided upon was: A player controlling a spaceship tries to shoot as many mobs (enemies and asteroids) as required for a level while taking minimal damage, until the player’s lives run out.

**System Analysis**

Next, we began to map out possible inputs from the user, and what outputs from the program would be appropriate. This mainly included keys that were needed for gameplay, which was created from the game’s overall logic. Below is a list of inputs and their outputs for the entire program.

Controls During Gameplay

|  |  |
| --- | --- |
| **Input** | **Output** |
| Left and right arrow keys | Spaceship moves left and right |
| Spacebar (hold for continuous fire) | Spaceship fires forward towards mobs |
| ‘P’ key | Pause in game |
| ‘C’ key | Continue game after pausing |
| ‘Esc’ key | Quit game |

Collision Inputs and Outputs

|  |  |
| --- | --- |
| **Input** | **Output** |
| Player’s bullet hits enemy | Enemy ship explodes and player’s points increase |
| Enemy ship fires at player’s ship | Player’s ship loses health |
| Mob collides with player | Player’s ship loses health |
| Mob reaches bottom of the screen | Player loses points |
| Powerup collides with player | Firepower: Player has two streams of rapid firing for 10 seconds  Speed: Player has double its initial speed for 10 seconds  Health: Player gains full health on health bar |

**System Design**

Once goals and mandates were created, the challenge that our program would display was high. Different sections of the game needed to be created to work flawlessly with each other. My partner and I decided to work on separate ends towards a specific section, increasing efficiency of work. Afterwards, the translation of the system design into Python/Pygame code would be merged with each other and tuned appropriately. Below is a general design plan of the program.

Step 1: Display the title screen, allowing the user to choose ship and initiate the game.

Step 2: Display the game screen with player’s spaceship and info bar (including level, health, lives, points, and enemies left).

Step 3: Spawn Mobs until player eliminates a certain number of them (this amount will increase for each level).

Step 4: Take player inputs from keyboard to move player’s ship and display bullets that fire to the top of the screen (display explosion if there is a collision between bullet and enemy mob).

Step 5: Spawn power ups that either provide extra firepower, speed, or health if collision between the player’s ship and powerup occurs.

Step 6: Initiate new level once player has eliminated the necessary number of mobs, while giving player an extra life (a maximum of 6 lives at any given time).

Step 7: Repeat steps 2-5 with more enemy mobs that are faster and more powerful until player loses all their lives.

Step 8: Display game over screen and determine if user made a top 10 score. If so, prompt user to enter a username.

Step 9: Display leaderboard with the player’s rank of all time scores if the player has completed a a top 10 score.

Step 10: Display options to restart the game or quit the program.

I was mainly responsible for the game fundamentals, including the majority of the objects that had to be spawned, and how each level was created. I decided that the difference for each level depended on different parameters that could be changed and injected into the game like a math function formula. I also had to plan out what info should the player see and where would it be placed.

**Implementation**

Implementing the game and its various elements undoubtedly took the longest in the project. My partner and I worked on separate sections of the project according to our system design roles and kept each other in touch when we were done coding specifics. I worked on many of the classes, as well as the game loop and levels.

While working on the game screen, I mainly had to decide on details such as a good position for the ship relative to the screen height, the position of the info above or below the player, and the spawning plus movement of the mobs. Mob movement was the most challenging. Using the random module with Python, I based the movement of the enemies on the movements of the player by checking if the mobs were in a specific zone surrounding the player. I also used the random module to determine each enemy’s firing rate back at the player. The scrolling backgrounds were implemented in such a way that there was seamless transition when tiling the images to create the illusion of an infinite scrolling background.

In order to create levels, the game function had to be modified so some of the parameters could be generalized, and that any inputs the function can take would be plugged into different objects and their behaviour. Enemy and asteroid movement was generalized so they could take any range of speeds before spawning on the game board. The amount of space above the game screen was also generalized to accommodate for the set amount of mobs that would be spawning, keeping the screen from becoming too dense and intensive to play. I planned to make at least 90 levels, so on a separate py file, I created a dictionary so for every key (the level) had a set list of values that can be called by the main program into the game function. Similarly, we used another file to store all the images we used pygame.image for, in order to be called by the main function.

Once the fundamentals were complete, we started paying attention to details. We refined some behaviours of the game, such as the timing of the explosion vs. the mob disappearing, displaying the number of points added to the score once a mob was killed, and how the health and power up timer bars were drawn onto the screen.

**Testing**

The final step included tenaciously testing the game in order to debug it and streamline any unnecessary details. If there was an opportunity to complete a missing detail, it was taken. An example was implementing cheat codes to skip to the final two levels of the game. We found that it would take a substantially long time to get to the final levels so this method was targeted towards those who wanted to experience them directly the hardest challenge in the game.

A notable bug was an issue regarding the scrolling background. The magic behind the infinite scrolling background illusion was that two the same images were stacked on top of each other, and moved towards the bottom at the same speed. Once the top of the bottom-most image reached beyond the bottom of the screen, it was moved on top the second image above it. Through careful analysis, it was soon discovered that the problem resided in the y-speed of the scrolling background. The y-speed was not a factor of the y-dimension of the screen, thus when the image moved out of sight and back to the top, there was a small delay between the first and second images, resulting in a white space (due to a remainder left over at the end of the background image). To fix this, a y-speed to a factor of the screen dimension was calculated and this solved the issue flawlessly.

This testing stage was ongoing both in the game’s development and even more after the game was completed. Using pencil and paper to fix problems such as above helped in assessing logical problems. Overall, this section was very enjoyable because it required a clear mind, endurance, and tenacity to do, proving to be a very good challenge.

**Reflection/Overall Remarks**

The program we have created was a big success, as we had managed to reach even further than we had initially planned. These extra features included the leaderboard, confirmation dialogues, customizable ships, and the cheat codes. Although it was a success from this viewpoint, I felt in a mechanical and efficiency sense, it was far from perfect. Firstly, the way the levels were calculated needed more attention, as the difficulty between each level was sometimes inconsistent and not challenging enough. If more time was allotted, more detail could have been put in levels, such as mini boss enemies. Secondly, while testing, there were quite a few points where the game had a subtle slowdown, but it did not prevent the game from being played and enjoyed.

Self Evaluation:

Overall, I felt happy with the progress I made towards developing the game. With the amount of time I spent on it, 2-3 hours per day over the course of 2 weeks, I am satisfied with the pace at which I worked at. A few criticisms I would give myself include how I contributed to the project and how ambitious my goals were.

Although both of us worked on the code, I felt I displayed greater control over the code, since I mainly had all of the scripts and other files. I think a good solution to equal contribution is a coding version of Google Docs, where multiple people can work on the same file at the same time. An example of this is Github and its commit system.

Secondly, realistically in the time frame allocated, I kept on adding more features, details, and goals that complicated our project. For this, I can only say I was having too much fun coding the game. I do plan on revisiting my game over the summer, but for the assignment/summative, there had to be a stopping point, which was hard for me to conform to.

Peer Evaluation:

Throughout the project, my partner was agreeable and very easy to get along with. Even off school when we did not see each other in person, communicating with my partner was fairly easy. He was an enormous help to the project and always contributed towards the goals assigned on time. As a partner, he would even remind me to stop if some of my goals and features I wanted to add next were too ambitious.

If I had to criticize, in contrast to what I had stated before in my self evaluation, it would be that he could have played a larger role towards the game mechanics and coding, which would result in a more efficient work method.

Overall, I felt our synergy was very high, since we had accomplished so much within a limited time frame, despite a bit of tightness in time management. With better methods of working on the code together, the time frame would not be a concern at all.