G51FSE - Galactic War

Qingqi Shi qxs03u 4205003

Yinfeng Xie yxx13u 4205378

Design Portfolio

Galactic War will be a 2D arcade shooting game with a theme based on space and spaceship battles. It is going to be designed and implemented using python/pygame.

Game Concept

With a basic idea of game content in mind, we have looked at some popular 2D spaceship shooting games such as Space Shooter and Galaxy Gunner. We have decided that our game will have a similar theme as we will incorporate a few basic principles from these games outlined above whereas there will also be additional features that will make our game stand apart from them. The main objective of our game will be to obtain scores according to both the playtime and mainly the number of enemies killed。

Requirements Definition

Our task was to produce a 2D arcade game from scratch through design, prototyping, refinement, implementation, testing, and evaluation stages. The game that we are going to create should be innovative and addictive and should aim at people of all ages.

Requirements Specification

- Functional Requirements

1. The player ship shall only be controlled by mouse
    1.1 The ship will always face towards the cursor
    1.2 The ship will accelerate towards the cursor when the right mouse button is held down, and stops accelerating when it reaches a set velocity limit
    1.3 The ship will decelerate when the right mouse button is let go
    1.4 The ship will stop moving when its velocity reaches zero
    1.5 The ship will shoot bullets continuously when the left mouse button is held down, unless the auto-shoot option is selected
    1.6 The ship will stop shooting when the player lets go of the left button of the mouse

2. The bullets shall be released by both player's ship and enemies' ships
    2.1 The spaceships (player's ship or enemies' ships) will be destroyed when they collide with a bullet
    2.2 The bullets will be removed when they reach the edges of the window
    2.3 The player will be able to choose between click-and-shoot mode and auto-shoot mode

3. The enemies' spaceships shall appear at random positions on the screen
    3.1 There will be only types of enemy spaceship
    3.2 The ships will stay at the positions where they first appear
    3.3 The ships will appear at random times
    3.4 Both the player's ship and the enemy's ship will be destroyed when they collide with each other
    3.5 The ships will disappear and get removed when they are destroyed

4. Asteroids shall appear at random positions on the screen
    4.1 Asteroids will come in different sizes
    4.2 Asteroids may travel in set paths toward random directions
    4.3 Both the player's ship and asteroid will be destroyed when they collide with each other
    4.4 Asteroids will disappear and get removed when they are destroyed

5. The game will be over when the player's spaceship is destroyed
    5.1 The player's ship will be destroyed when it collides with any other elements including enemy spaceships, bullets fired by enemy spaceships and asteroids
    5.2 The player's ship will disappear and get removed when it is destroyed

6. There shall be a scoring system
    6.1 The score will increase continuously as the playtime increases
    6.2 The score will also increase every time an element in the game is destroyed by the player

7. The level of difficulty will increase when a certain playtime is reached
    7.1 The maximum number of enemy spaceships on the screen will increase as the level increases

8. Sounds shall be played in the game
    8.1 Sound effects will be played every time an element is destroyed
    8.2 Background music may be playing throughout the game
    8.3 Sound effects will be played when the spaceships shoot

9. There shall be a local leaderboard to show the player's high-score and the best score achieved

10. There may be an option in the game to allow connection to a social network

- **Non-Function Requirements**

1. The game must be compatible with Windows operating system, and it may be compatible with other major operating systems

2. The game must be able to run smoothly at an optimum frames per second

3. The player must be able to pause, restart or quit the game at any time

4. The player must be able to restart the game in under 1 second

5. The game must use less than 50 megabytes of hard disk space

Game Design

- Game Control

In order to stand apart from similar games and to ensure simplicity of control, we have decided that all of protagonist's actions shall be controlled entirely by the mouse. The player controlled spaceship will always track the position of the mouse cursor and face towards it, so that it changes direction when the player moves the mouse. When the right mouse button is held down, the ship will accelerate towards the cursor and stop when it reaches the cursor, also when the player lets go of the right mouse button the ship decelerates. Moreover, the ship will shoot when the player clicks the left mouse button and it will even shoot constantly when the left mouse button is kept held down.

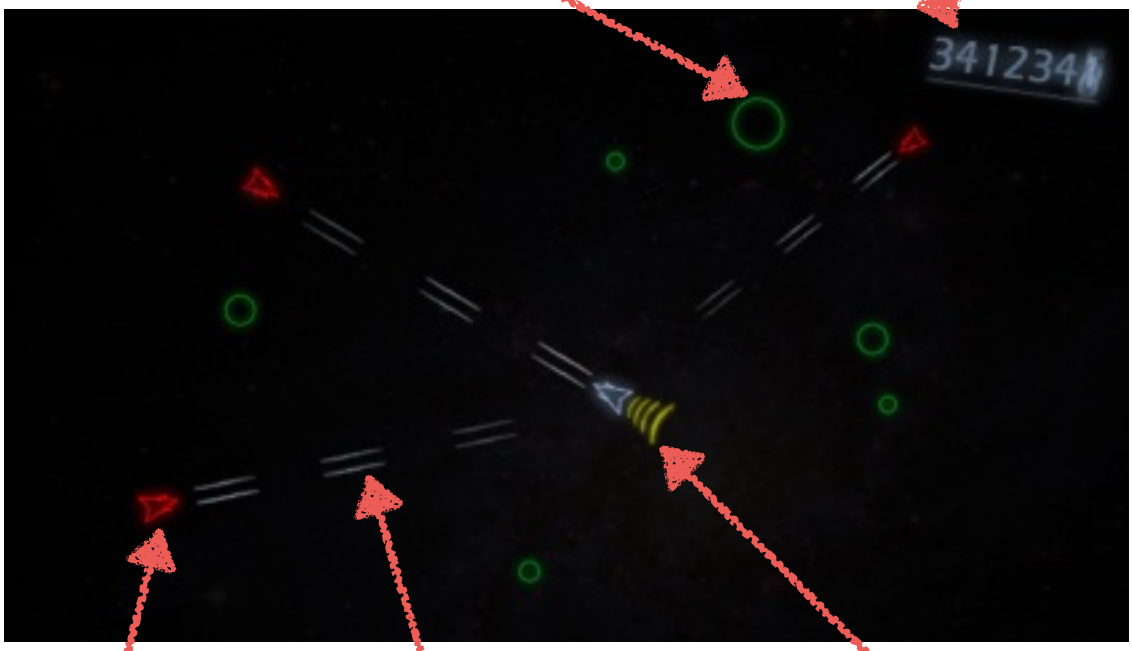
- Behaviour of elements

There will be enemy spaceships and asteroids appearing at random positions on the screen at random times. All enemy ships will stay fixed at the position where they first appeared and they will be targeting and firing at the protagonist continuously until they get destroyed . Additionally, in order to provide a richer experience for the player, asteroids will move towards random directions instead of staying fixed. However,they will be removed from the game once they reach the other edge of the screen.

- Scoring Mechanism

The player will gain scores according to both the length of play time and the number of enemy objects (enemy spaceships and asteroids) destroyed by the player. One thousand points will be awarded to the player each time an enemy spaceship is destroyed while the scores assigned to asteroids may vary depending on their sizes. Also, score will increase continuously by 1 point every millisecond as the play time increases. However, the score will be reset when the game is over.

- **Interaction Design (Expected outcome)**

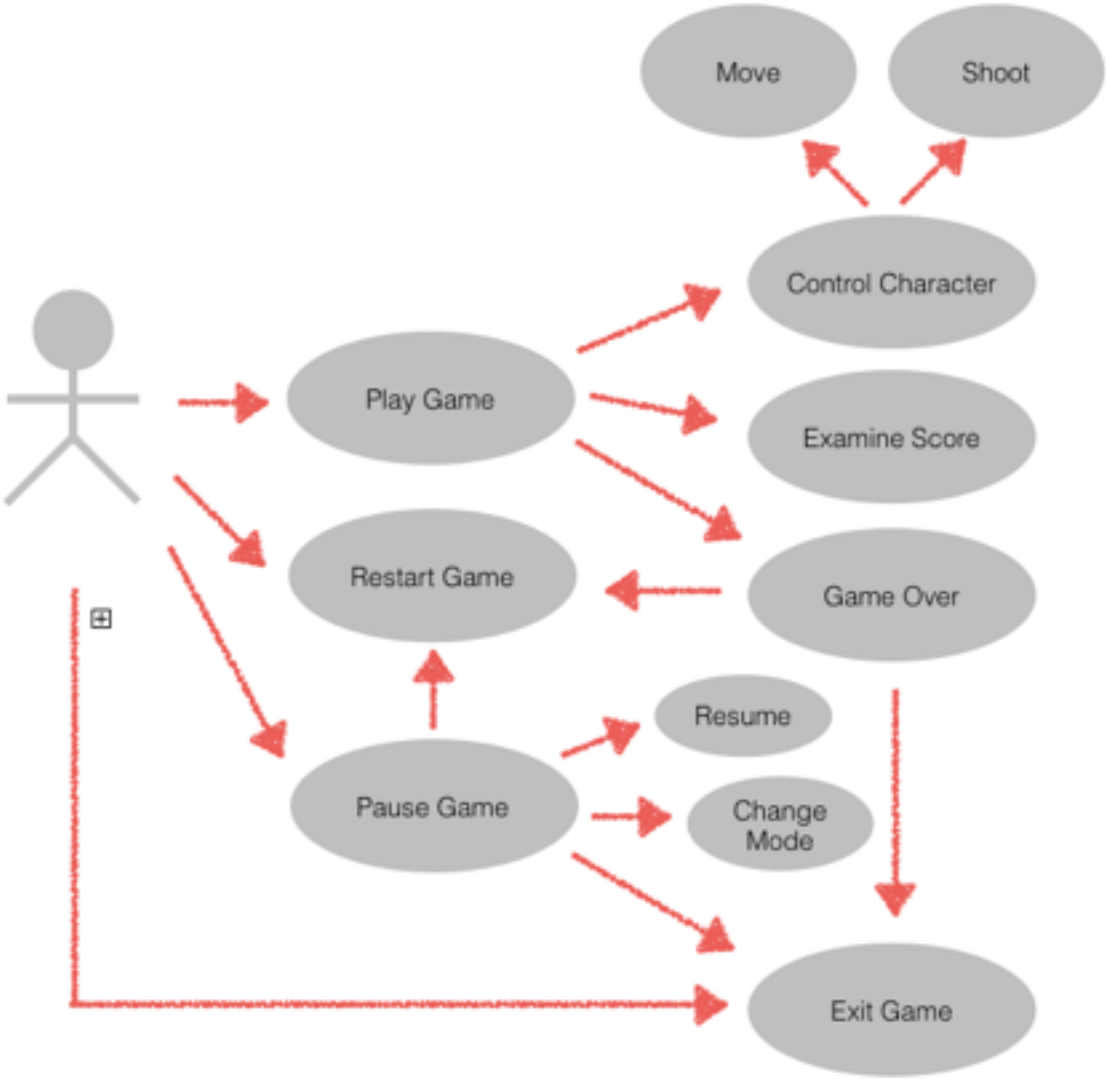


The player can easily press the ‘R’ key on the keyboard at any time to restart the game, or the ‘Esc’ key to pause the game and open the options menu. The restart option is also available in the options menu as well as other options such as ‘Quit’ and ‘Resume’.

- **Game Overview**

Galactic War is going to be a single-player game viewed from an eagle-eye perspective . The protagonist will be a spaceship controlled with mouse by the player who is able to move freely within the game window. Initially, the ship is going to be placed at the centre of the screen. When the game starts,enemy spaceships and asteroids will appear randomly and continuously onto the screen. The enemy spaceships will always target and shoot at the protagonist constantly while asteroids will move in random paths. In order to survive, the protagonist has to destroy enemy ships and asteroids by shooting at them while dodging them and their attacks. The game is over when the player controlled spaceship collides with any other elements in the game.

- **Use Case Diagram**



- **Use Case Diagram**

We have decided to break down the tasks into smaller ones so that problems can be dealt with one at a time. Different classes will handle the implementation of different components of the game.

Menu:object
displayMenu()

Weapon
weaponNum:int coolDown:float bulletSpeed:float bullets:[Bullet]
fire() clearBullets()

EnemyShip:Spaceship
playership
update()

Spaceship:Sprite
image:Surface originImage:Surface rect:Rect velocity:[float] acceleration:float speed:float maxSpeed:int direction:int weapon:Weapon center[int]
update() updateDirection() accelerate() decelerate() changeWeapon(int) calculateCenter() fire()

Bullet:Sprite
image:Surface originImage:Surface rect:Rect direction:int position:(int, int) bulletSpeed:float weaponNum:int
update()

PlayerShip:Spaceship
update()

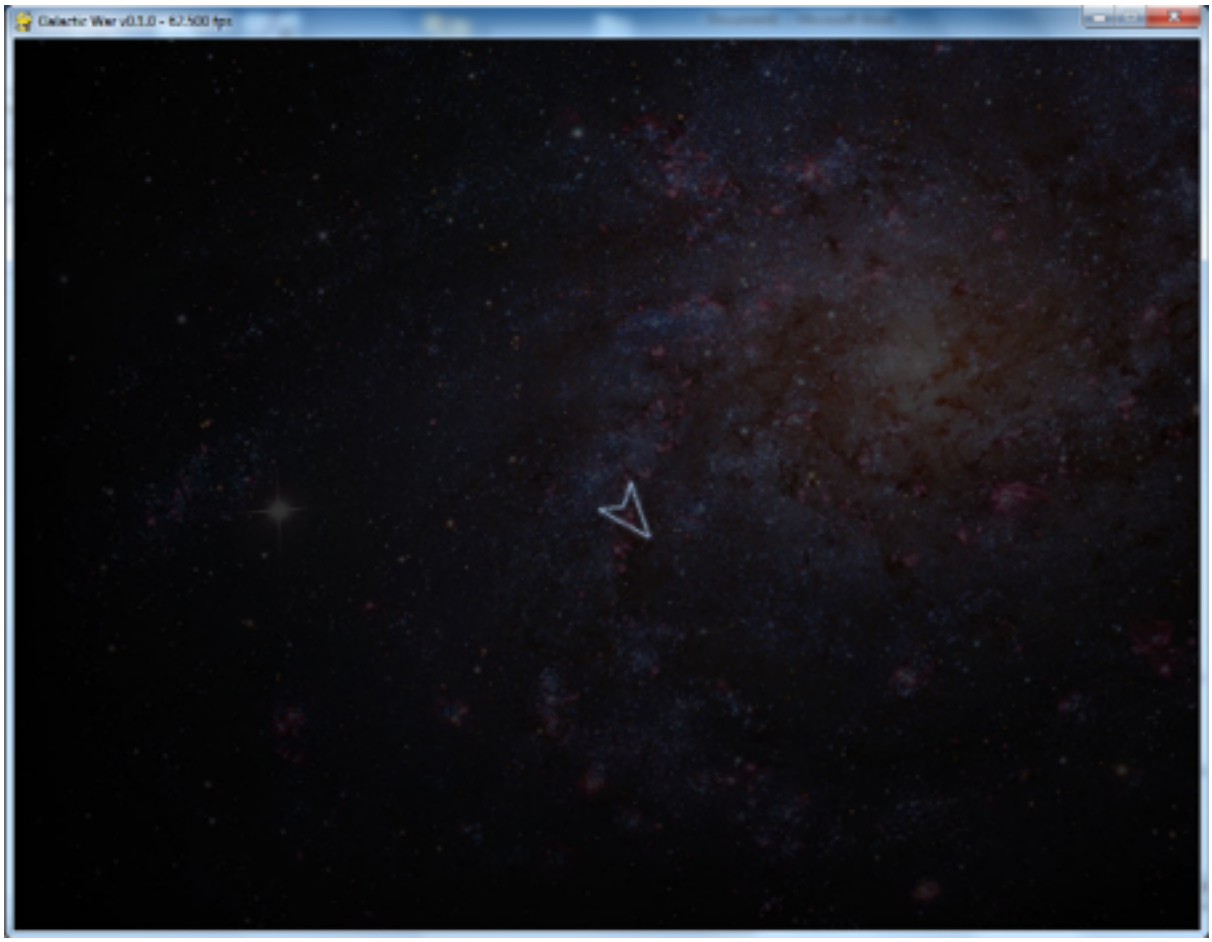
HUD:object
updateScore()

Prototype Evaluation & Refinement

The first prototype of Galactic War implemented majority of the most basic functionalities of the game.

- **Game Screen**

The prototype had resolution of 1024 x 768 pixels with an image of space as background. Initially, it was designed to run at 40 frames per second which we found unsatisfactory. In order to ensure better user experiences, the rate was then changed to 60 frame per second so that all elements were able to move smoothly. The player controlled spaceship was placed at the centre of the screen when the game started.



- **Movements**

As we have planned at the beginning of the project, the protagonist's actions were able to be controlled entirely by mouse in the prototype. This was one of the main functionalities that we felt that could make our game stand out.

- The ship always faces towards the cursor wherever the cursor is
- The ship shoots a bullet when the player clicks the left mouse button
- The ship shoots bullets continuously when the player holds down the left mouse button
- The ship starts to accelerate until it reaches its maximum speed when the player holds down the right mouse button
- The ship starts to decelerate before it stops when the player lets go of the right mouse button
- The ship shoots and moves at the same time if the player holds down both mouse buttons

- **Enemy Spaceships**

In a random amount of time after the game started, the enemy spaceships started to appear one by one at random points on the screen. Rather than having spaceships and various bullets filling up the screen, we have decided that enemy spaceships should share the same weapon class with the player's spaceship in order to ensure a simpler and clearer

game interface. This means that all spaceships in the game fire the same kind of bullet with same fire rate whereas the only difference is that enemy spaceships shoot at the protagonist while the protagonist shoot at the direction of the cursor. However, after a few test-runs with the prototype, we realised that the game becomes too difficult as the number of enemy spaceships increases, because they all fire at a rapid rate.

Our solution was to reduce the rate of bullet firing of all spaceships to allow more time for the protagonist to dodge the bullets.
Code added:

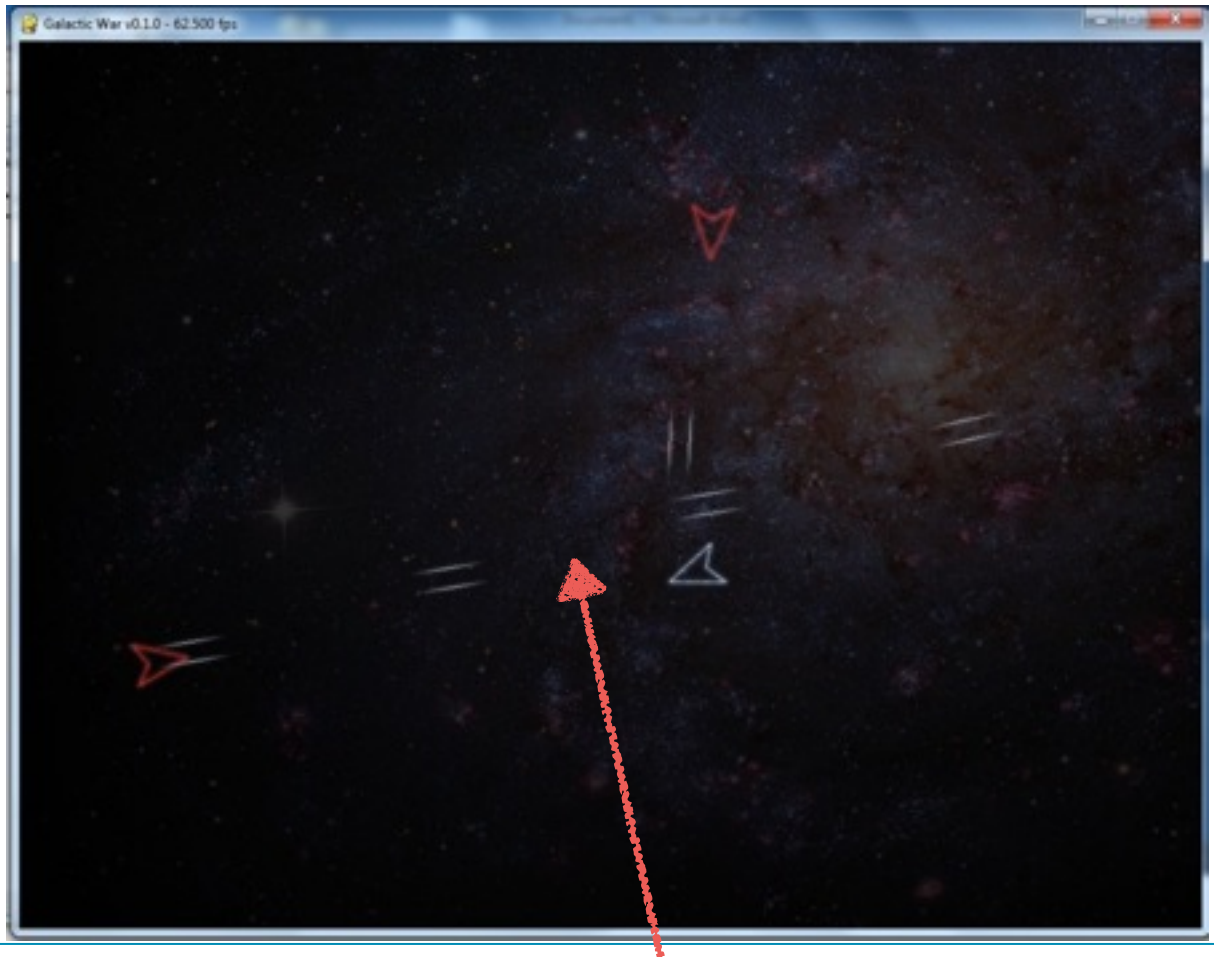
```
self.coolDown = 0.5
```

Cool down time was initialised to be 0.5 second.

```
def fire(self, position, direction):
    if time.time() - self.lastFired > self.coolDown:
        newBullet = Bullet(self.groups, self.bulletSpeed, self.weaponNum, position, direction)
        self.bullets.append(newBullet)
        self.lastFired = time.time()
```

This function tells the system to only fire a bullet if the cool down time from the time that last bullet was fired is greater than the cool down set for the weapon (0.5s in this case).

Refined version of the bullets:



- **Further Refinements**

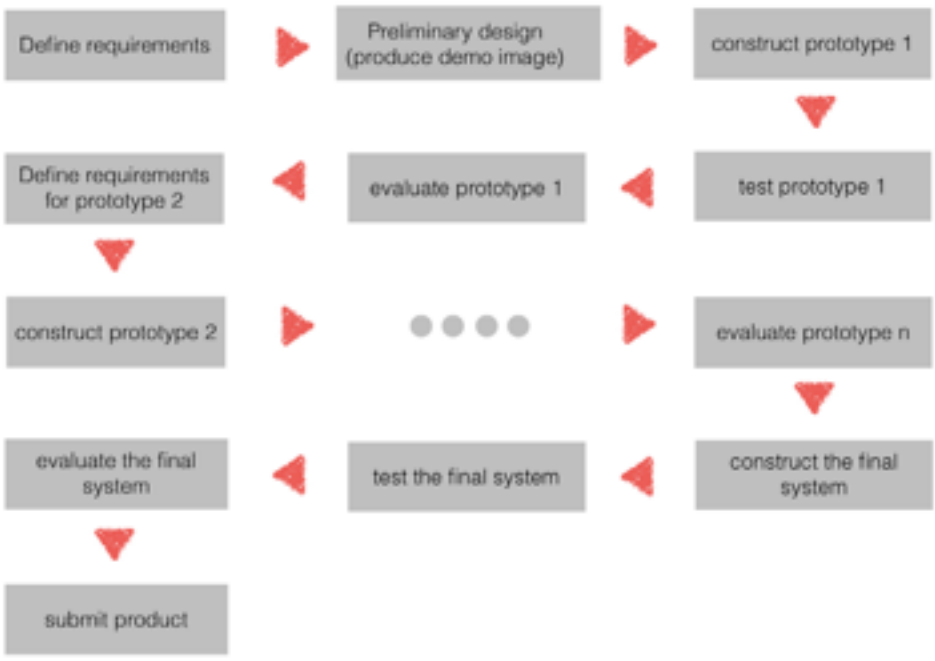
There are still a few features that need to be added since this is so far a prototype that only contains the main game mechanisms.

- A scoring system needed to keep track of the player's score.
- Keyboard shortcuts should be added to allow instant replay or exit.
- A message needed to notify the player when the game is over.
- Options to restart or quit the game should appear when the game is over.
- Asteroids should be added to provide richer game content.
- Auto-shoot mode should be introduced to offer simpler control for the player.
- The level of difficulty should increase when a certain playtime is reached.
- The maximum number of enemy ships on the screen should be limited according to the level. For example, only one enemy ship may be allowed on the screen at the start of the game and the maximum number of enemy ships on the screen may increase by one every 30 seconds.
- Sound effects should be added to the game.
- Best score achieved and current game score may be displayed when the game is over.
- Connection to social networks may be added.

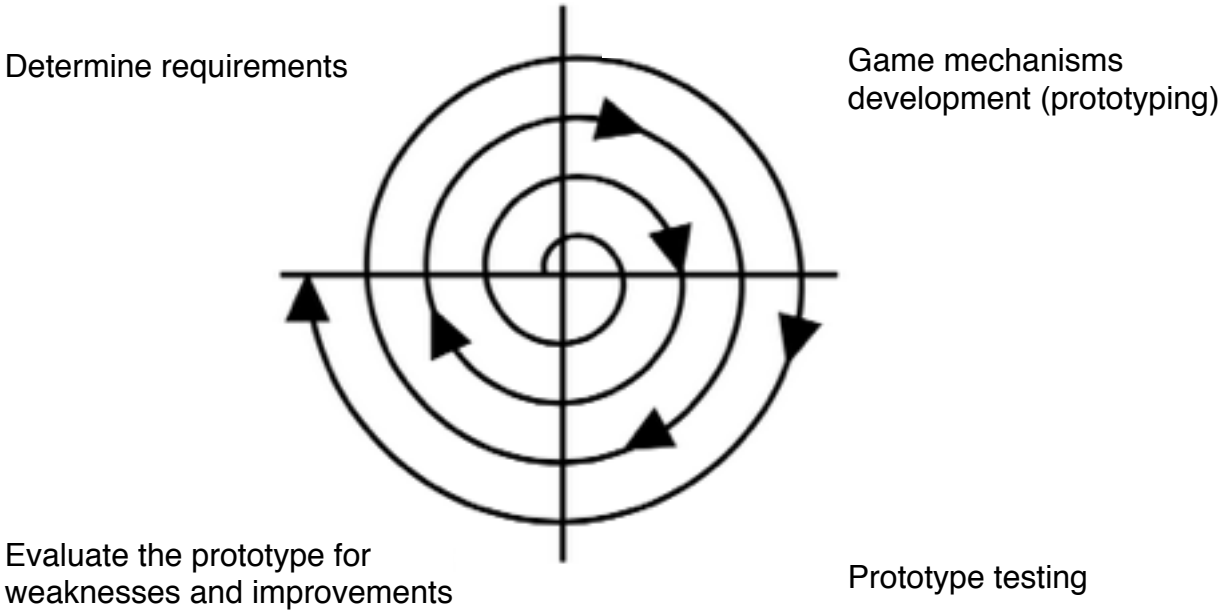
Development Methodology

The Spiral Model would be the most appropriate model to describe the process of the development of our game, as our idea was to combine design and prototyping together in each stage through the development of the game. We started by defining important features of the game, trying them out, discovering achievable improvements while testing the features, and them moving on to the next stage of the development. These steps were repeated until we have the final product that satisfied the requirements.

- **Development Process**

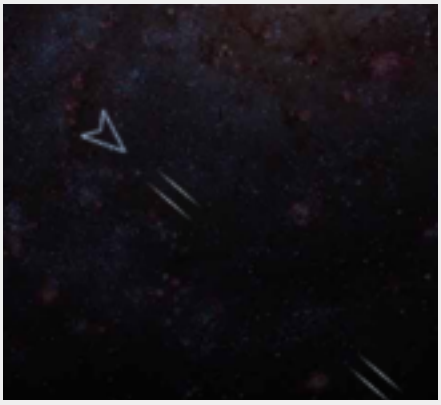
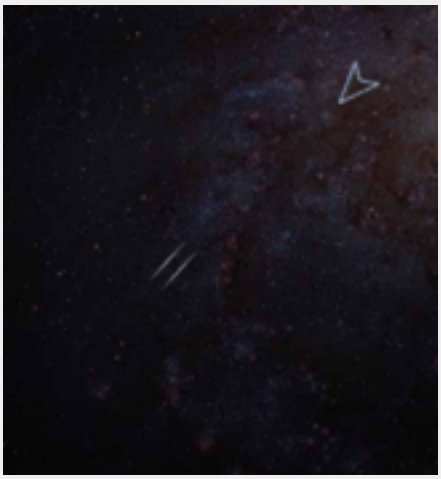


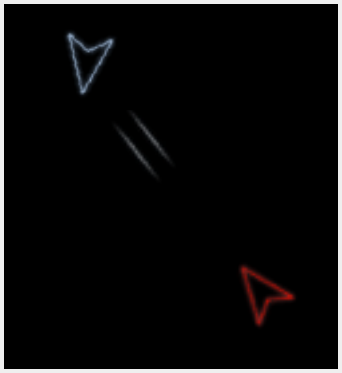
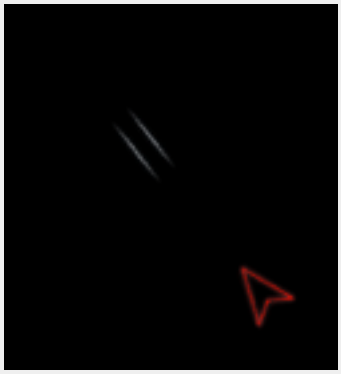
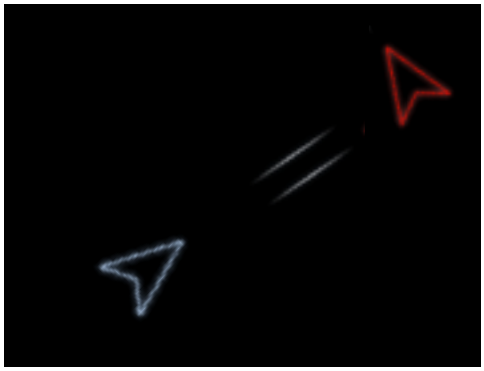
Our development stages can be generalised as follow:

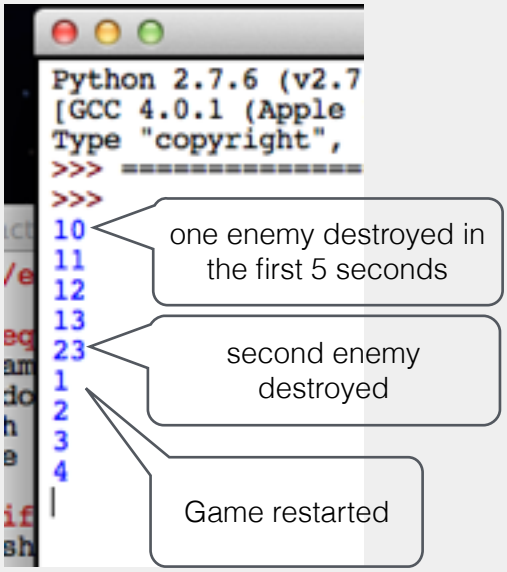


- **Prototype Functionality Tests**

Test No.	Test Name	Test Description	Evidence
1	Fire bullets	Use left mouse button to test if bullets can be fired and get rendered correctly	

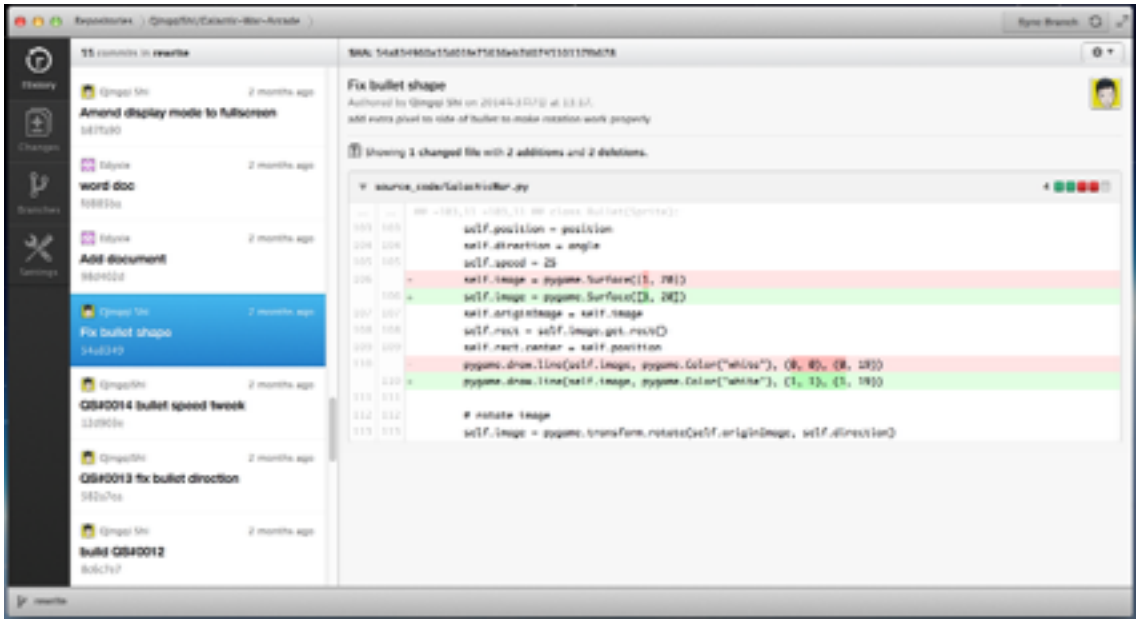
Test No.	Test Name	Test Description	Evidence
2	Bullet movement	Fire bullets at different directions to test for the calculations of direction and movement of bullets	 
3	Bullet removal	Print out the number of bullets currently being stored to test if bullets outside the screen gets deleted	

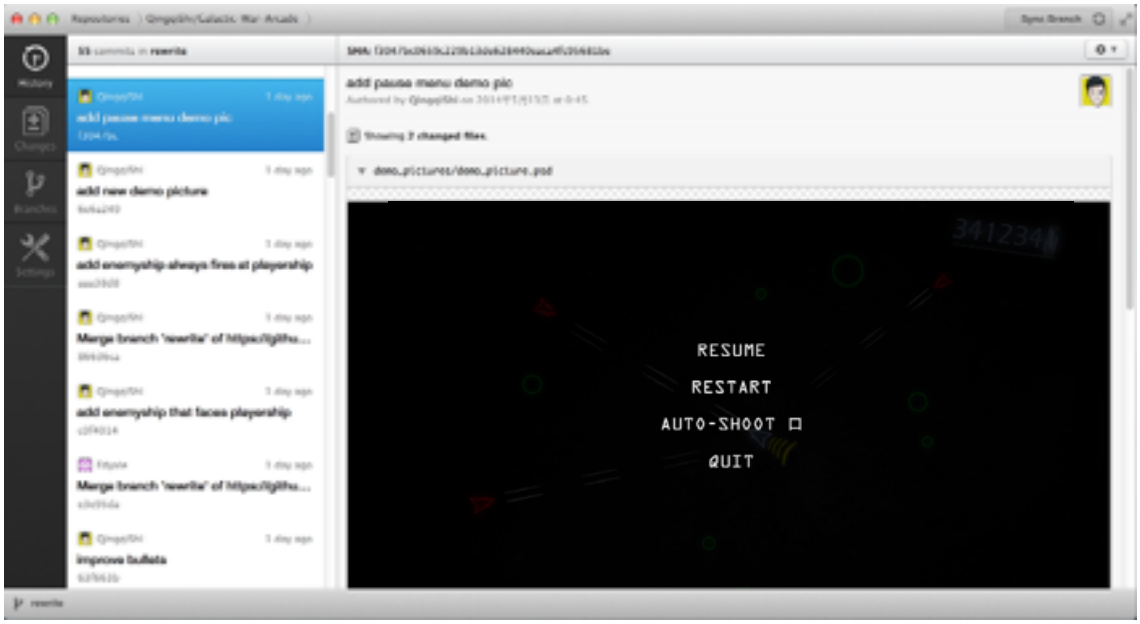
Test No.	Test Name	Test Description	Evidence
4	Player spaceship removal	Test if the player's spaceship disappears when its hit by enemy's bullets	 
5	Enemy spaceship removal	Test if the enemy's spaceship disappears when its hit by player's bullets	 

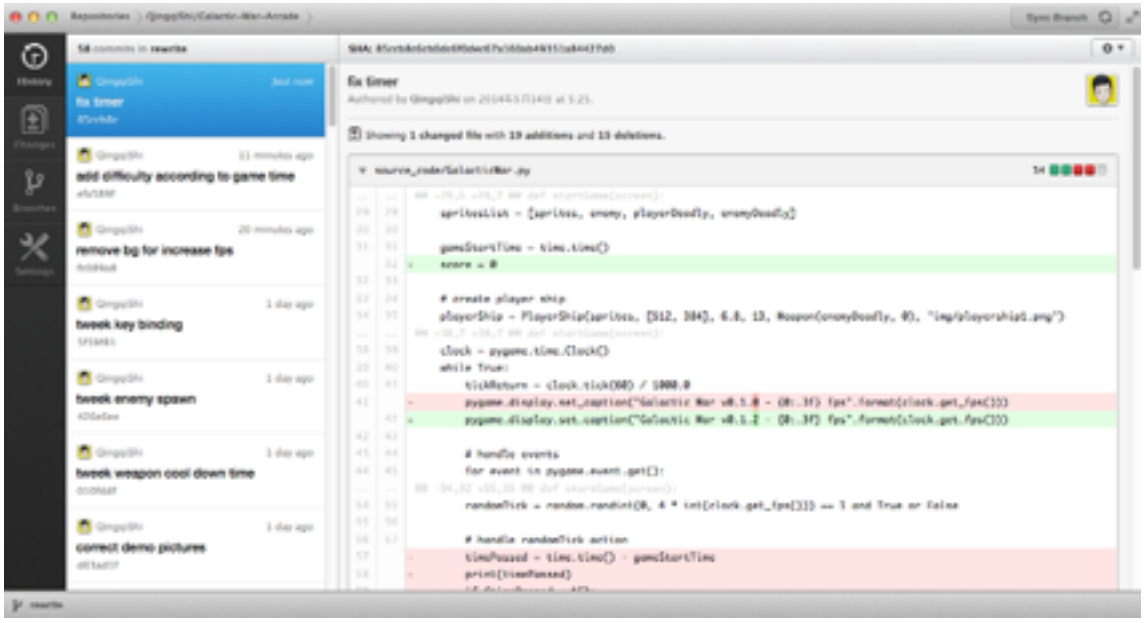
Test No.	Test Name	Test Description	Evidence
6	Scoring system	Test if the score increases by one every 5 seconds Test if the score increases by 10 when an enemy spaceship is destroyed Test if the score will be reset when the game is restarted	
7	Game Over menu	Test if a system message appears when the game is over Test if the current game score and the best score achieved are both displayed when the game is over	 

Version Control System

In order to effectively manage the different parts of the game development between the two of us, a version control system was used to avoid the problems that we may encounter when sharing documents and codes between us. GitHub was our preferred system, as it allowed us to access the latest files in the repositories instantly and to work on the same code without having to send the files back and forth. Also, one of the most beneficial advantage that we found of using a version control system was that we can instantly browse previous ‘commits’ of work and restore to earlier versions when something bad happens.







Final Functionality Test

Test No.	Purpose	Pass/Fail	Comments
1	Game runs at approximately 60 fps	Pass	displayed correctly at the top of the window
2	Correct screen size (resolution)	Pass	None
3	Player's spaceship rendered correctly	Pass	Appeared at the centre of the screen
4	Player's spaceship always faces towards the cursor	Pass	None
5	Player's spaceship's weapon system implements correctly	Pass	The ship always fires toward the position of the cursor
6	Player's spaceship moves correctly	Pass	Worked correctly as expected
7	Player's spaceship accelerates correctly	Pass	The acceleration of the ship was calculated correctly
8	Spaceships get removed from the screen when destroyed	Pass	Both the player's ship and the enemies' ships will be removed when destroyed
9	Spaceships get destroyed when collide with opponent's bullets	Pass	Enemy ships can only be destroyed by player's bullets.

Test No.	Purpose	Pass/Fail	Comments
10	Bullets get removed from the game when they leave the screen	Pass	None
11	Game score displays correctly	Pass	None
12	Scoring system implements correctly	Pass	The score was altered when enemy objects destroyed and when a certain play time is reached
13	Keyboard shortcuts works correctly	Pass	None
14	System messages display correctly when the game is over	Pass	None
15	Options to restart or quit the game should appear when the game is over	Fail	This requirement was altered. Instead of providing the options, there will be a message that notifies the player to use keyboard shortcuts to restart or quit
16	Auto-shoot mode implements correctly when enabled	Pass	Auto-shoot mode can be enabled by keyboard shortcut
17	The level of difficulty increases as play time increases	Pass	The level of difficulty is measure by the amount of enemy spaceship on the screen at the same time
18	Sound effects implement correctly	Pass	None

Test No.	Purpose	Pass/Fail	Comments
19	Display best score achieved and current game score when the game is over	Pass	None
20	Connection to social networks	Fail	We were unable to achieve this requirement with Pygame
21	Introduce asteroids to the game	Fail	This idea was abandoned after a few tests with the game prototype, as we felt that the involvement of asteroids would make the game too difficult to play

Evaluation & Refinement

- The player controlled spaceship was coded to accelerate towards the cursor when the player holds down the right mouse button. We believe that we have calculated the acceleration correctly and it was able to provide a rather realistic play experience for the player.

- The level of the game is presented by the maximum amount of enemy spaceships on the screen at the same time, and the level will increase by one every 15 seconds until the player reaches level four. This means that the higher the level, the more enemy spaceships that could appear on the screen. Initially, there will only be one enemy spaceship appearing, the number increases as play time increases. However, the reason we have set the maximum amount of enemy ships appearing same time to be four is that we felt that extra enemy spaceships will increase the difficulty significantly, therefore reduce the length of play time of each game.

- In the latest version of the game, the player controlled spaceship and the enemy spaceships share the same weapon system with same fire rate. According to the feedback from one of our test users, the disadvantage of the current weapon system is that the bullets released from player spaceship and enemy spaceships are the same, which made it harder to distinguish if the bullets were fired by enemies. One of the suggested from test users for future improvement was to introduce a new weapon system for the player controlled spaceship which may contain multiple bullets to enrich user experiences.

- Another test user suggested that it might be a good idea to add a power-up management system to the game. For example, the protagonist may be able to obtain shields in the game to help to survive longer under enemy attacks.

- There is a scoring system that keeps track of the points earned by the player. The player is awarded one point every five-seconds and two points on destruction of an enemy spaceship.
- The game can be easily paused and resumed by pressing the ‘Esc’ key. However, a notable error was found and then resolved during the final testing of the game regarding the pause function. The problem was that the play time counter does not stop when the game is paused, which effected the level of the game when the game is resumed, as the level mechanism is depending on the length of the play time.