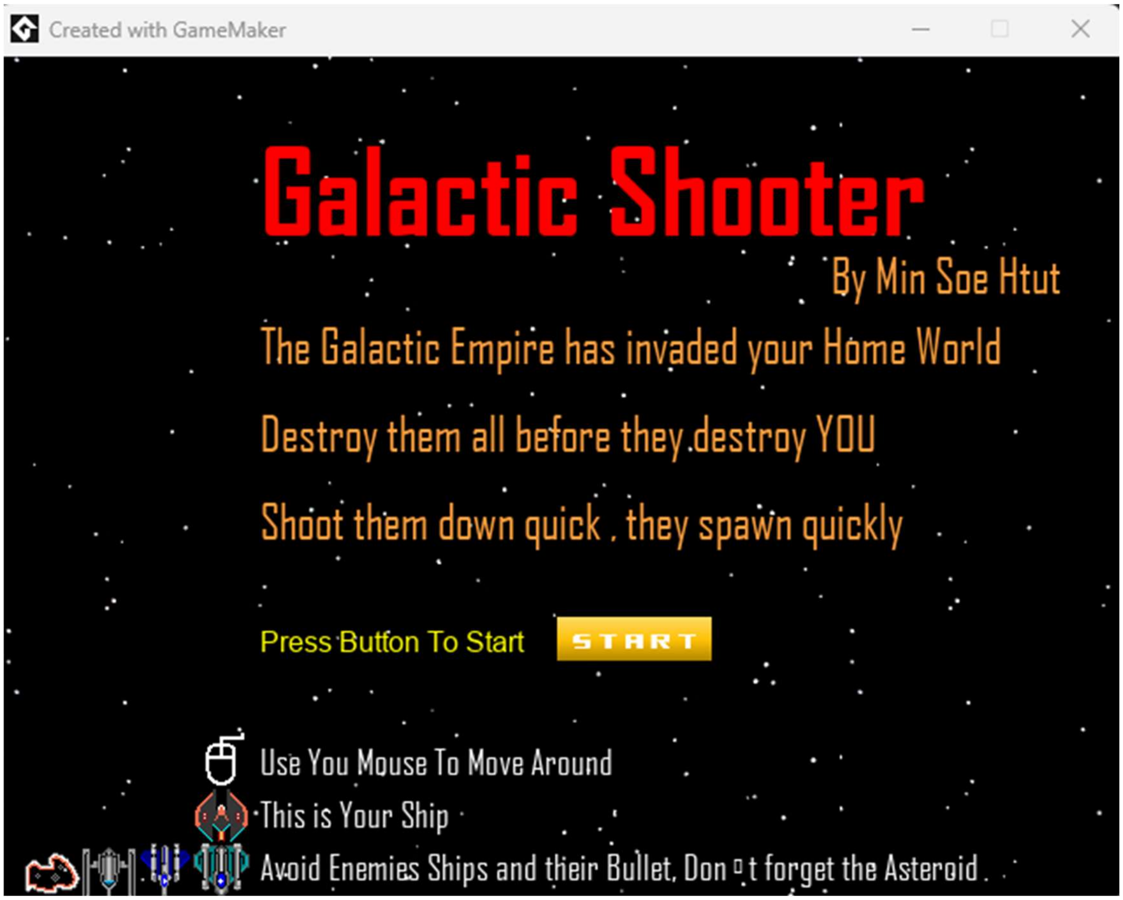
Galactic Shooter

1.Introduction - what did you set out to do (objectives)

I have planned to create a base on a space shooter game called "Galactic Shooter" The main objective of the game is for players to navigate through levels, destroy all enemy ships, avoid obstacles, and achieve the highest score.



Galactic Shooter

By Min Soe Htut

The Galactic Empire has invaded your Home World

Destroy them all before they destroy YOU

Shoot them down quick , they spawn quickly

Press Button To Start

Use You Mouse To Move Around

This is Your Ship

Avoid Enemies Ships and their Bullet, Don▫t forget the Asteroid

Figure 1: A screen of the intro screen

2.Design - how did you approach the problem (use your sketches to help explain)

The player ship will be firing three bullets and once I collect the weapon update it will shoot faster, enemy ships will move down from the top of the screen, and once it moves out of the screen it will jump back to the top. There will be multiple rooms for the player to enjoy, The Start Screen to introduce the player to what is the game about and basic guide on how to play the game.

2.1Conceptualization:

The design phase began with identifying key game objects and their functions. The player's ship, enemy ships, and asteroids were the primary elements considered.

2.3Player Interaction:

A pivotal design decision was enabling player movement via the mouse. This choice was intended to enhance user engagement and control precision.

2.3Enemy Mechanics:

Enemy ships were designed to spawn randomly at the top of the screen, creating a dynamic challenge. To maintain gameplay continuity, enemies reappear at the top after leaving the screen.

Obstacles:

Asteroids were introduced as an additional obstacle that required players to employ strategy and quick reflexes.

Level Design:

A multiple-room structure rather than a single-room endless wave of enemies was conceptualized to progressively increase the game's difficulty and complexity.

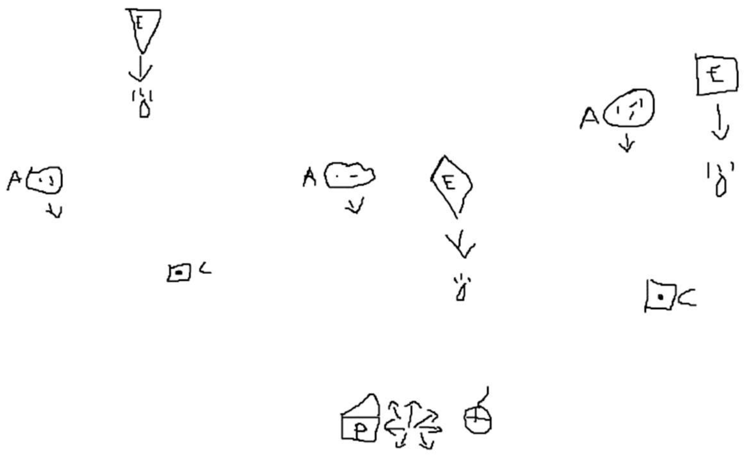


Figure 2: A sketch of the game design on paper.

P = Player E = Enime C = Collectible A = Asteroid

3.Build - how did you put it together (remembering to explain why you did it that way) - use screenshots to illustrate your process.

The game itself is fast-paced so the player needs to have quick precision control. So I first implemented the mouse mechanic to the player ship for control precision to shoot and avoid the fast-moving and random-firing enemy ships. The Player also needs to avoid the Asteroids & collect the collectable. By having a multiple room, I can introduce a new enemy per room rather than the player can only climb the highest score in an unending wave of enemies.

3.1Development Stages:

Enemy Spawning: Implemented random spawning of enemy ships at the screen's top.

Asteroid Mechanics: Integrated moving asteroids that players must avoid.

Level Progression: Developed a level structure with multiple rooms.

Scoring System: Established a point system based on asteroid avoidance.

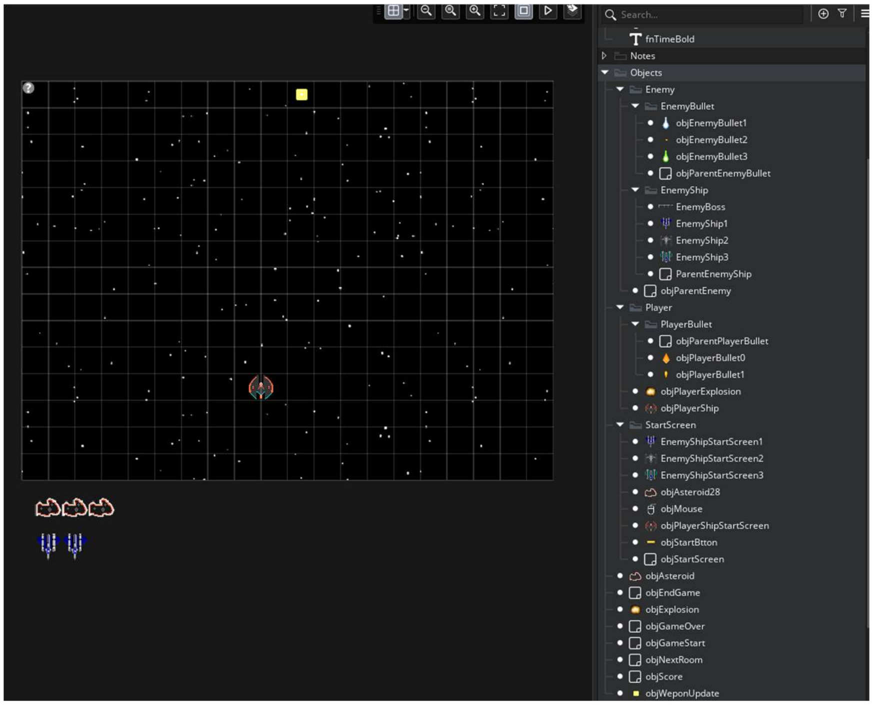


Figure 3: A screenshot of the game development

3.2Challenges Encountered:

Enemy spawning was problematic, with a lack of randomness and dependency on existing ships.

Difficulty in implementing random collectable spawns.

The enemy spawning mechanism lacked the desired sophistication.

Difficulty in mechanics of the guided missile.

When the End Condition of the room is met, it doesn't jump to the next room completely.

4. Result - what did you wind up with (and again use screenshots to show what you got)

4.1Player

The player ship uses the jump to point function to move around using the mouse by setting mouse_x, mouse_y. For the bullet Fire Rate, the count increases by +1 by every game step, once it reaches 60 it fires Three Bullet then sets the count to 0 and repeats the step. Once the All Enemy are destroyed move to the next room. Below is the code for firing three bullets at the same time.

```
// Create Top Bullet
var topBullet = instance_create_layer(x, y, "Instances", objPlayerBullet0);
with (topBullet) {direction = 90;}

// Create Top-Left Bullet
var topLeftBullet = instance_create_layer(x, y, "Instances",
objPlayerBullet1);
with (topLeftBullet) {direction = 135;}

// Create Top-Right Bullet
var topRightBullet = instance_create_layer(x, y, "Instances",
objPlayerBullet1);
with (topRightBullet) {direction = 45;}
```

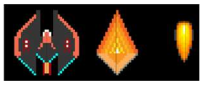
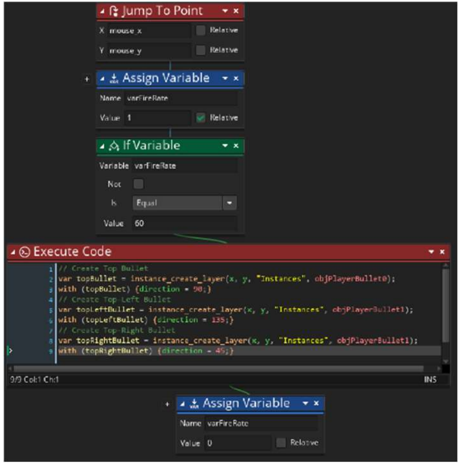


Figure 4.1.2 Screenshot of Player Ship & Bullets



Jump To Point
X mouse_x ☐ Relative
Y mouse_y ☐ Relative

Assign Variable
Name varFireRate
Value 1 ☑ Relative

If Variable
Variable varFireRate
Not ☐
Is Equal
Value 60

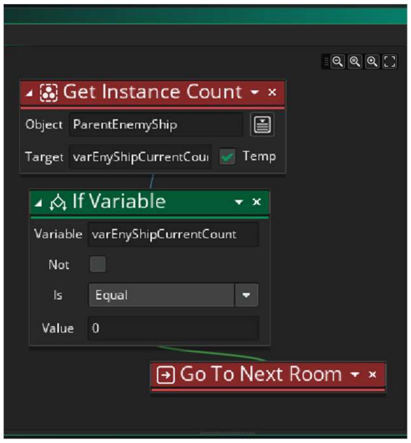
Execute Code

```
// Create Top Bullet
var topBullet = instance_create_layer(x, y, "Instances", objPlayerBullet0);
with (topBullet) {direction = 90;}
// Create Top-Left Bullet
var topLeftBullet = instance_create_layer(x, y, "Instances", objPlayerBullet1);
with (topLeftBullet) {direction = 135;}
// Create Top-Right Bullet
var topRightBullet = instance_create_layer(x, y, "Instances", objPlayerBullet1);
with (topRightBullet) {direction = 45;}
```

9/9 Col:1 Ch:1 INS

Assign Variable
Name varFireRate
Value 0 ☐ Relative

Figure 4.1.2 Screenshot of step even in obj player



Get Instance Count
Object ParentEnemyShip
Target varEnyShipCurrentCou ☑ Temp

If Variable
Variable varEnyShipCurrentCount
Not ☐
Is Equal
Value 0

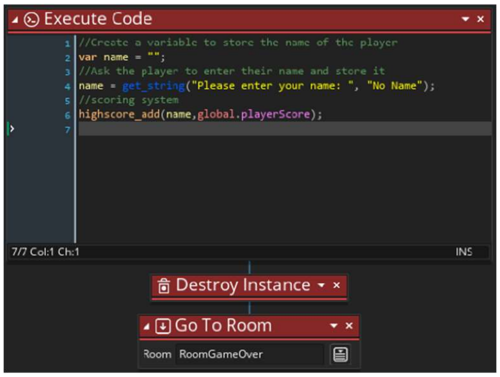
Go To Next Room

Figure 4.1.3 Screenshot of End step even in obj player

4.1Player

Once the Player hits with enemy ship, enemy bullet or asteroid the instance changes to objPlayerExplosion. In the objPlayerExplosion event ask the player to Enter Their Name to show it on the scoreboard. Then destroy the objPlayerExplosion and move to another room. Below is the code asking Name of the player.

```
//Create a variable to store the name of the player
var name = "";
//Ask the player to enter their name and store it
name = get_string("Please enter your name: ", "No Name");
//scoring system
highscore_add(name,global.playerScore);
```



Execute Code

```
//Create a variable to store the name of the player
var name = "";
//Ask the player to enter their name and store it
name = get_string("Please enter your name: ", "No Name");
//scoring system
highscore_add(name,global.playerScore);
```

7/7 Col:1 Ch:1 INS

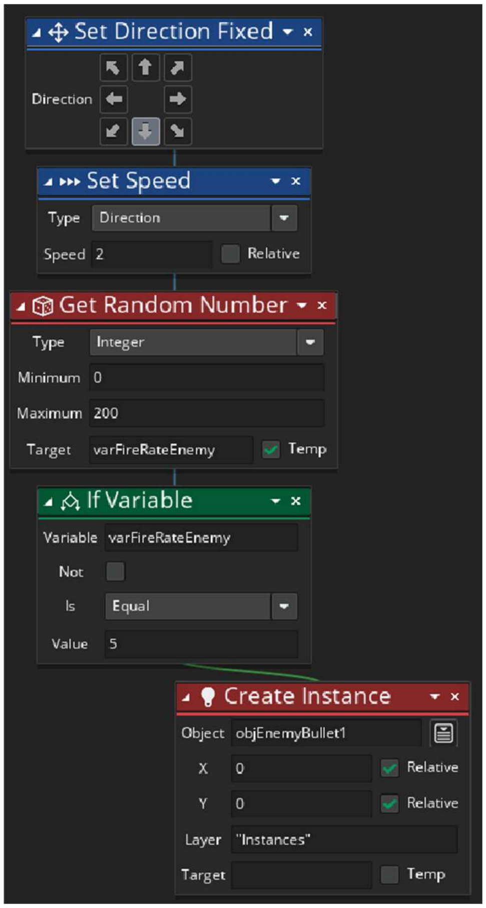
Destroy Instance

Go To Room
Room RoomGameOver

Figure 4.1.4 Screenshot of even in objPlayerExplosion

4.2 Enemy

Enemy ships will spawn randomly at the top of the screen and move down to the bottom. If the same type of ship is less than 5 it will create a new enemy at the top. The enemy will reappear at the top after leaving the screen. They will fire randomly in a straight direction.



Set Direction Fixed
Direction: down

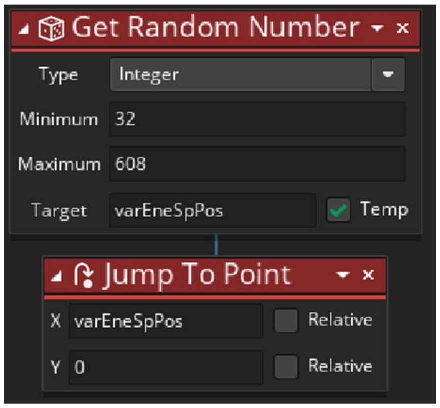
Set Speed
Type: Direction
Speed: 2 □ Relative

Get Random Number
Type: Integer
Minimum: 0
Maximum: 200
Target: varFireRateEnemy ☑ Temp

If Variable
Variable: varFireRateEnemy
Not: □
Is: Equal
Value: 5

Create Instance
Object: objEnemyBullet1
X: 0 ☑ Relative
Y: 0 ☑ Relative
Layer: "Instances"
Target: □ Temp

Figure 4.2.3 Screenshot of Begin Step Event in Enemy Ship



Get Random Number
Type: Integer
Minimum: 32
Maximum: 608
Target: varEneSpPos ☑ Temp

Jump To Point
X: varEneSpPos □ Relative
Y: 0 □ Relative

Figure 4.2.4 Screenshot of Outside Room the event in Enemy Ship

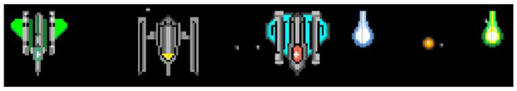
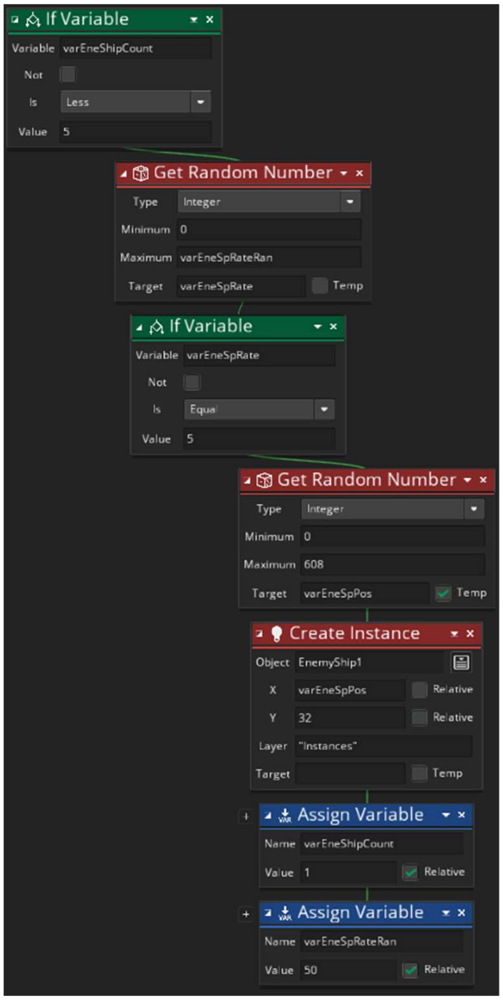


Figure 4.2.1 Screenshot of Enemy Ships & Bullets



If Variable
Variable: varEneShipCount
Not: □
Is: Less
Value: 5

Get Random Number
Type: Integer
Minimum: 0
Maximum: varEneSpRateRan
Target: varEneSpRate □ Temp

If Variable
Variable: varEneSpRate
Not: □
Is: Equal
Value: 5

Get Random Number
Type: Integer
Minimum: 0
Maximum: 608
Target: varEneSpPos ☑ Temp

Create Instance
Object: EnemyShip1
X: varEneSpPos □ Relative
Y: 32 □ Relative
Layer: "Instances"
Target: □ Temp

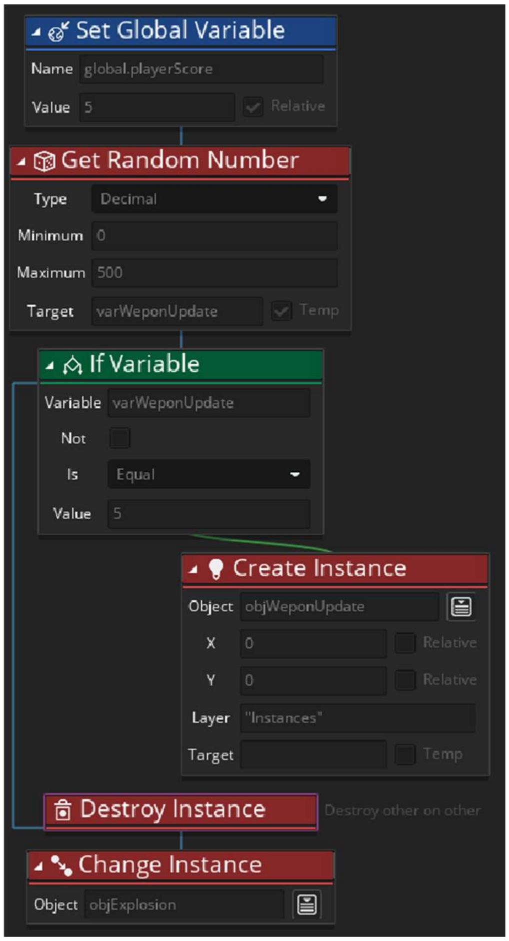
Assign Variable
Name: varEneShipCount
Value: 1 ☑ Relative

Assign Variable
Name: varEneSpRateRan
Value: 50 ☑ Relative

Figure 4.2.2 Screenshot of Step Event in Enemy Ship

4.2 Enemy

When the enemy ship is destroyed by the player, add the Score count by 5 and spawn the collectable randomly at the area where the ship has been destroyed.
Destroy the player bullet and change the instance to the ObjExplostion.



Set Global Variable
Name global.playerScore
Value 5 ☑ Relative

Get Random Number
Type Decimal
Minimum 0
Maximum 500
Target varWeponUpdate ☑ Temp

If Variable
Variable varWeponUpdate
Not □
Is Equal
Value 5

Create Instance
Object objWeponUpdate
X 0 □ Relative
Y 0 □ Relative
Layer "Instances"
Target □ Temp

Destroy Instance
Destroy other on other

Change Instance
Object objExplosion

Figure 4.2.5 Screenshot of Collection vent with Player Bullet in Enemy Ship

4.3 Asteroid

Asteroids are an additional obstacle that players need to avoid. Once it exits from the screen player gets 5 Points for a Score and it will reappear at a random location at the top of the screen.

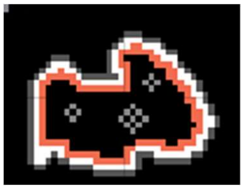


Figure 4.3.1 Screenshot of Asteroid

4.4 Collectable

Collectable will move down from the top for a player to collect. Once it exits from the screen it will reappear at a random location at the top of the screen. When the player collects the collectable they will provide the player with 10 points for Score.

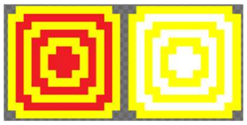


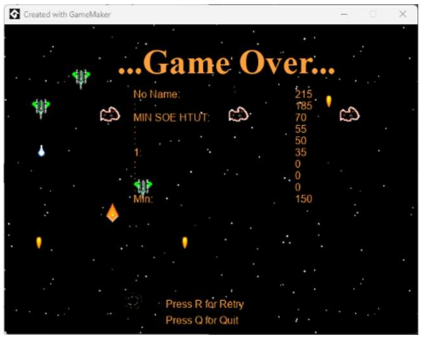
Figure 4.4.1 Screenshot of Collectable

4.5 Scoring System & High Score Table

The scoring system is where the player can keep track of the score for destroying enemy ships, collecting collectables, and avoiding asteroids. At the end of the Game when a player dies or wins the game player will be asked to Fill in their Name and their height score will be displayed at the table for this run.

Score: 165

Figure 4.5.1 Screenshot of Live Score



...Game Over...

```
No Name:              215
                      185
MIN SOE HTUT:         70
                      55
                      50
1:                    35
                      0
                      0
                      0
Min:                  150
```

Press R for Retry
Press Q for Quit

Figure 4.5.1 Screenshot of High Score Table

4.6 Room And Sound

There is a Start Screen where players are introduced to what is the game about and a basic guide on how to play the game.

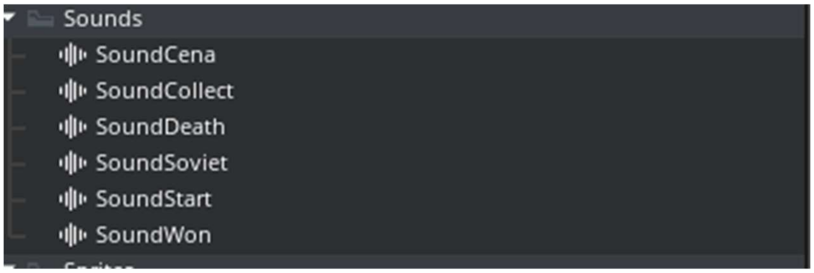
There is a Test Room that the developer can use to try out the object.

There are 4 total playable rooms where the player can face the enemy.

After each room, there were where players could decide to go to the next room or quit a room.

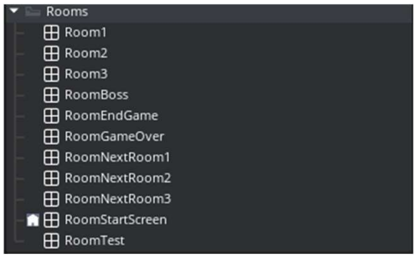
There is a Room for when players win the game or lose the game where the player can see their high score.

There is a Sound for each room and a sound for collectables.



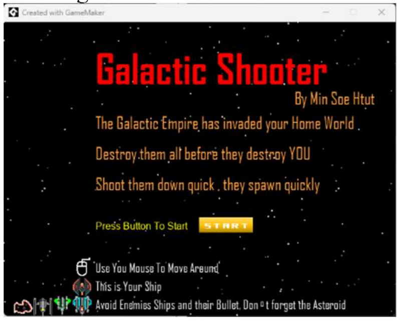
- Sounds
  - SoundCena
  - SoundCollect
  - SoundDeath
  - SoundSoviet
  - SoundStart
  - SoundWon

Figure 4.6.4 Screenshot of Sounds



- Rooms
  - Room1
  - Room2
  - Room3
  - RoomBoss
  - RoomEndGame
  - RoomGameOver
  - RoomNextRoom1
  - RoomNextRoom2
  - RoomNextRoom3
  - RoomStartScreen
  - RoomTest

Figure 4.6.1 Screenshot Rooms



Galactic Shooter

By Min Soe Htut

The Galactic Empire has invaded your Home World

Destroy them all before they destroy YOU

Shoot them down quick , they spawn quickly

Press Button To Start START

Use You Mouse To Move Around
This is Your Ship
Avoid Enemies Ships and their Bullet. Don▫t forget the Asteroid

Figure 4.6.2 Screenshot of Start Screen

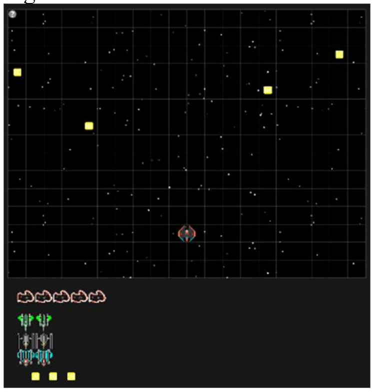


Figure 4.6.3 Screenshot of playable Room

5. Reflection - is that what you meant to do, how did it go, are you happy with that

5.1 Time Investment:

Approximately 42-45 hours were dedicated, primarily focusing on creating the game, debugging, and refining mechanics.

5.2 Achievements and Shortcomings:

While the primary objectives were mostly met, additional features and bug fixes remain. The Remaining functions that need debugging and rework mechanics are guided missile, collectable, Final Boss, and Room jumping.

5.3 Learning Experience:

Understanding of game mechanics and how they interact to create a cohesive and engaging player experience. The process of designing and implementing features like enemy behaviors, obstacle dynamics, and player controls was critical in understanding the delicate balance required in game design. A substantial portion of the development process was dedicated to troubleshooting and resolving unexpected issues. This involved debugging code, refining game mechanics, and ensuring smooth gameplay. The challenges encountered necessitated a methodical approach to problem-solving and creative and innovative solutions. Creating a game required adaptability and a willingness to learn new techniques and technologies. The development process highlighted the importance of patience and persistence. Creating a game, especially one that is engaging and bug-free, is a time-consuming process that often involves revisiting and revising many aspects of the game multiple times. The skills and knowledge gained through this process are invaluable and will undoubtedly be beneficial in future projects.

6. Cost/time analysis - did it go to plan/was it cost-effective

Development Efficiency:

The time invested versus the project's current stage reflects moderate development efficiency. The learning curve and debugging phases were particularly time-consuming.

6.1 Expectation Alignment:

The time commitment broadly aligns with initial expectations, though some aspects require more time than anticipated.

6.2 Conclusion - did you meet your objectives (a summary of the previous few sections)

"Galactic Shooter" has made commendable progress towards achieving its initial objectives. However, further development is necessary to resolve current issues and add new features. The project has been both challenging and rewarding, closely aligning with the envisioned game concept. Despite the need for further improvements, the journey of developing "Galactic Shooter" has been a combination of a series of challenges and achievements.