Presidency University

Space infinity war

PREPARED BY:

Name	**Patient ID**
Md.Taufiqul Islam	173 039 038
Johora Islam Ema	173 054 038
Sohanur Rahman	172 017 038
Luna Akter	173 013 039
Sathy Akter	143 127 038

SUPERVISOR

Nadia Binte Asif

Lecturer (Full Time)

Presidency University

Department of Electrical and Computer Engineering

Submitted to the Department of Electrical and Computer Engineering of Presidency University, Dhaka, Bangladesh in partial fulfillment of the requirements for the degree of

BSc IN COMPUTER SCIENCE & ENGINEERING

DECLARATION

We would like to express our gratitude to our advisor **Nadia Binte Asif** for her guidance, support, and his continuous encouragement throughout the project. We are also very grateful and extend my sincere thanks to the faculty members of the Department of Electrical and Computer Engineering at Presidency University. Finally many thanks to friends, who have helped and given us suggestions throughout the project.

October 2021

…………………………………………………..

Nadia Binte Asif
Supervisor
Lecturer, Dept. of ECE
Presidency University

Approval

The Project titled „ Space infinity war‟ has been submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science and Engineering on October 2021 by the following patients and has been accepted as satisfactory.

i.	Md.Taufiqui islam	ID: 173-039-038
ii.	Johora islam ema	ID: 173-054-038
iii.	Luna akter	ID: 173-013-039
iv.	Sohanur Rahman	ID: 172-017-038
v.	Sathy akter	ID: 143-127-038

……………………………………………..

Nadia Binte Asif
Supervisor
Lecturer,, Dept. of ECE
Presidency University

ABSTRACT

A game is a structured form of play, usually undertaken for entertainment or fun, and sometimes used as an educational tool. Often, part of the entertainment for children playing a game is deciding who part of their audience is and who is a player. Key components of games are goals, rules, challenge, and interaction.

"Space infinity war" is one of them. This game comes with a single player mode. The graphics of the gameplay system is good and smooth to control for the users. This "Space infinity war" Game in python is an easy game for all. The strategy of this game is competitive and fun to play. You have to stay focused on it while playing. This game is controlled through a keyboard (Left, Right, and Fire), kill those randomly moving enemies, Get points and get rid of obstacle that can make your gameplay hard. And after each level it will get harder to play.

The programming language we have used is Python and used its library "Pygame". Pygame is an open-source Python library for making multimedia applications like games built on top of the excellent SDL library. This library is a combination of C, Python, Native and OpenGL. This allows you to create fully featured games and multimedia programs in the python language. Pygame is highly portable and runs on nearly every platform and operating system.

Table of Contents

LIST OF FIGURE

Chapter 1

Introduction

1.1 Introduction

Space Invaders is considered one of the most influential video games of all time. It helped expand the video game industry from a novelty to a global industry, and ushered in the golden age of arcade video games. It was the inspiration for numerous video games and game designers across different genres, and has been ported and re-released in various forms. The 1980 Atari VCS version quadrupled sales of the VCS, thereby becoming the first killer app for video game consoles. More broadly, the pixelated enemy alien has become a pop culture icon, often representing video games as a whole.

So we take the concept of "Space invader" and create our game "Space infinity war" using Python "Pygame" library. This game is controlled through a keyboard (Left, Right, and Fire), kill those randomly moving enemies, Get points and get rid of obstacle that can make your gameplay hard. And after each level it will get harder to play. By killing each enemy you will get 100 points and after reach 1000 points you will go to the next level

1.2 Digital Games for Children's

The game as a key component of children's development and their learning has gained a prominent place in modern forms of education, learning, and teaching. In this modern digital age, digital types of games have been developed which are now increasingly responding to the world of children and young people. Games contribute to spontaneous learning, interactivity, and dynamics through learning, as well as to the functional adoption of new learning contents. In chapter 2 we will discuss how "Space infinity war" plays a vital role on Digital game for children.

1.3 Design

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term "design" is defined as "the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization". It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient Detail to permit its physical realization. Before the implementation of our project, we decided on the

- Pygame
- Workflow diagram.
- Menu design
- Main game loop design
- Graphics design
- Playing strategy design
- Each level difficulties and achievements

Details can be found in Chapter 3.

1.4 Implementation

In chapter 4, we will Structure our Design into physical form by using the Python pygame library. And go through these steps:

- Pygmae initialization
- Title, logo, Background
- Main menu
- Main game loop
- Movement mechanics in pygame
- Input control and keys
- Pause, Restart, Quit
- Creating the enemies
- Shooting bullets
- Collision Detection
- Adding meteors
- Score and High score
- Background music and Sound\
- Game over

1.5 Conclusion

In this chapter, we briefly discussed the different aspects of the implementation of our project. In the next chapter, we are going to discuss the Design to implement our Space infinity war.

Chapter 2

Digital games for children's

2.1 Introduction

The use of games in education has a long history. The latest type of games, that is, digital games, have gained a significant place in today's society as a result of the growing influence the internet has had in the lives of people. Looking at the facts that individuals, who are accustomed to playing games, play games for approximately 10 thousand hours before they reach the age of 21 and that this period of time partly coincides with the time spent in education from primary to elementary school, provides significant insight about the place of digital games in individuals' lives. Digital games for learning are widely used (Martín-SanJosé, Juan, Gil-Gómez & Rando, 2014). O'Neil, Wainess and Baker (2005) described the learning potential through computer games as "striking". The relevant literature has revealed that children use computers every day to play games (Mumtaz, 2001), that learning through games is motivating (Virvou, Katsionis & Manos, 2005) and supports collaborative learning (Hoda, Henderson, Lee, Beh & Greenwood, 2014), that computer game playing improves mental rotation abilities in children between the ages of 8 and 9 (Lisi & Wolford, 2002), that playing games improves their thinking skills (Furió, González-Gancedo, Juan, Seguí & Costa, 2013), and that games can stimulate children's attention and memory as well as support their language development (Garaigordobil, 2005). The literature generally shows that games have a positive influence on learning. Lee, Wong and Fung (2010) pointed out a gap in the literature on how, precisely, computer games facilitate learning

Space infinity war game can be a good game for children who are newly interacting with the computer and other devices. Space infinity war game is easy to play and easy to understand the strategy of the game.

2.2 Game for education

Games designed specifically for the purpose of educating children can motivate self-learning and problem-solving skills to a great extent. Game-Based Learning plays important role in teaching by making students to collaborate, communicate, interact and work in teams. Strategic games improve the functioning of brain. Strategy-based games enhance the functioning of the brain, inspire children to learn new things, develop their skills and build an emotional connect to learn the subject matter. The feature of receiving feedback immediately after playing a game gives insight on how to improve the performance in a positive way.

2.3 Digital Games as a Context for Children's

The strong appeal of digital game play via dedicated video game consoles, computers, and, increasingly, mobile devices, remains indisputable. For example, in their recent survey of the computer and video game industry, the Entertainment Software Association 2018 reported that 64% of the 4,000 US households they sampled contained at least one individual who played video games 3 or more hours per week. A little under 30% of these players were aged 18 and under. Digital game play is often a family affair, with about two-thirds of the parents surveyed indicating that they played video games with their children on a weekly basis and perceived the games as beneficial to their children (Entertainment Software Association, 2018. Despite their ubiquitous use, there is very little research on the beneficial or harmful aspects of video games for school-age children's cognition or learning. This situation has ramifications for policy recommendations concerning their use that are also based on sparse research

2.4 Conclusion

The game contributes to more healthy childhood, intellectual development of children and at the same time speaking abilities stimulation. Through games children show their interests in what surrounds them, they are stimulated to keep researching and investigating so as to find their own solutions in a particular situations. That way, they get to know the environment better as well as human relations, they build their own behavior and world and life attitude, they get answers to many questions and feed their curiosity, enrich imagination, develop precision and reflexes etc.

Chapter 3

Design of Space Infinity War

3.1 Introduction

In this chapter, we are going to describe the design of our “Space infinity war” game. Before the design process, we formulated the requirements for Space infinity war system; e.g. Workflow diagram, Pygame features, Graphics design, Menu design, playing strategy, each level difficulties. Based on these requirements we first came up with the Workflow diagrams, and then the Pygame features. Based on these specifications we developed our application. Thus, in Section 3.2, we list the system and language requirements for running our application. 3.3 We will discuss about pygame. In Section 3.4 we discuss the about workflow diagram of our project. In Section 3.5, we discuss the Main menu design. In Section 3.6 we will discuss about our Main game loop design. In section In section 3.7 we will shows our game Graphics that we will use for the project, and in 3.8, 3.9 we will discuss about our playing strategy and each level difficulties.

3.2.1 Hardware Requirement

- 64-bit Windows 7 or later or OS X 10.11 or later.
- Processor: 1.5GHz or faster.
- Memory: 4GB (4,096MB) RAM
- keyboard, mouse, and headset

3.2.2 Language used

- Python

3.2.3 Library’s

- Pygame
- Random
- Math

3.3 Pygame:

Pygame is a set of Python modules designed for writing video games. Pygame adds functionality on top of the excellent SDL library. This allows you to create fully featured games and multimedia programs in the python language. Pygame is highly portable and runs on nearly every platform and operating system. Python is an excellent choice for rapid prototyping of games. But it has limits with performance. Therefore for more resource-intensive games, you should consider the industry standard which is C# with Unity or C++ with Unreal. Some popular games like EVE Online and Pirates of the Caribbean were created using Python.

3.4 Workflow Diagram:

The workflow diagrams show all sections of our project. We can see 2 main process sections Main menu and the main game loop. These 2 main section will play an important role in our project.

When we start our game the main menu window appears and it will take input from the user. Press play will take him to the main game loop, else he can quit the game click on the exit button and the program will stop and quit the game. in the main game loop the game start, we can see there are 5 sections in this game loop.

1. Events check.
2. Score
3. Game level
4. Spacecraft
5. Enemies

In the main game loop the game start, and we can play until the game is over, before that you can pause the game by pressing the "ESC" key, and you can also restart the game by pressing the "R" key. By pressing the "ESC" key it will take you to the Pause function and until you press "C" to continue the game won't continue. And by pressing "R" the game will restart and it will take you to the main menu. By pressing the 'spacebar' user can fire and if the bullet collides with an enemy user get 100 points in every collision. And reach every 1000 point user get a new level and a new Spacecraft. And Enemies moving speed also increase. And if the user can’t defeat the enemies, slowly the enemies come down to player position. And the game will be over.

When the game is over you can see a window that shows you lost, and also show your score. After that, you can play the game again and again to beat the High score or you can quit the game.

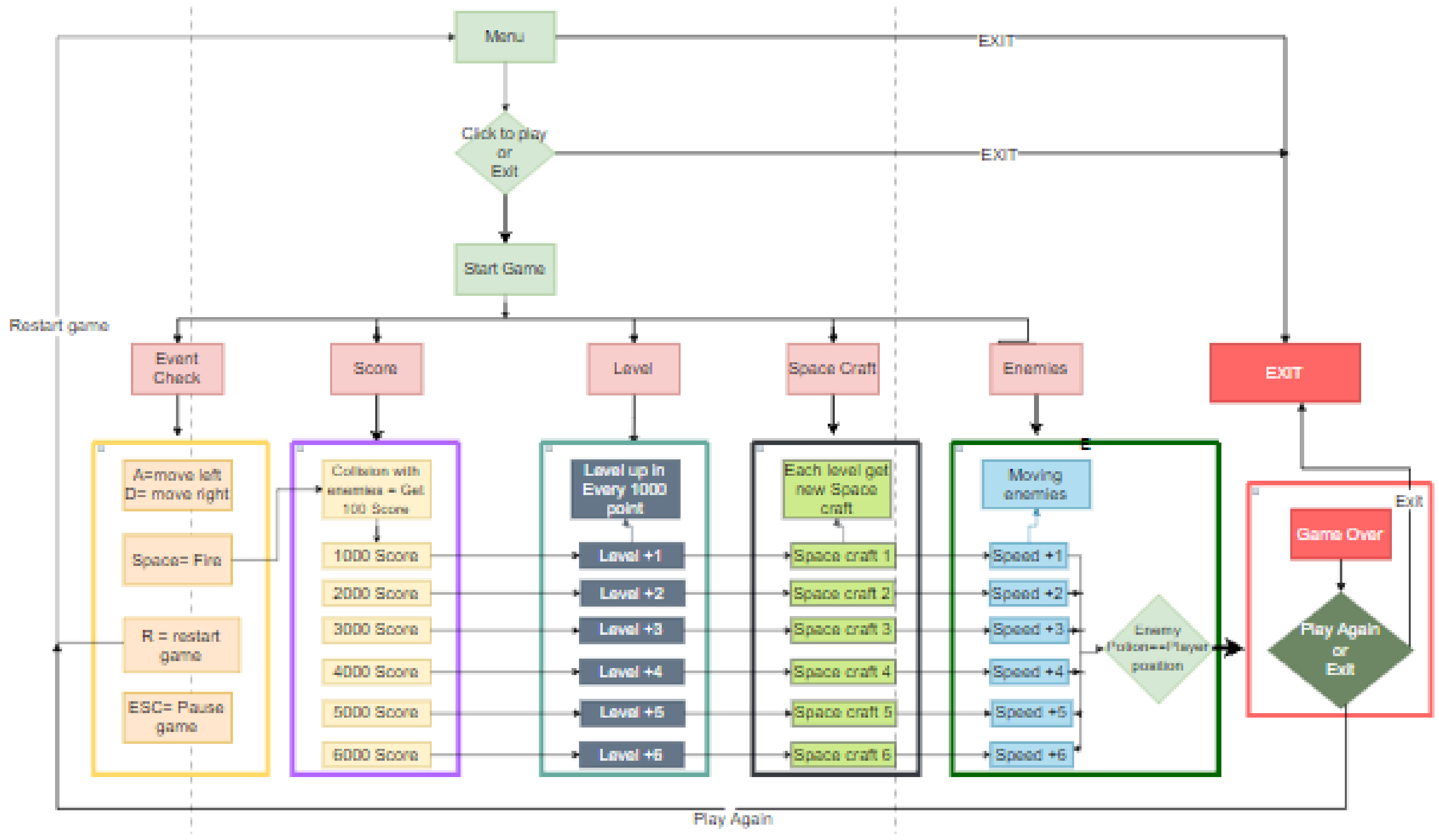


Figure 3.1: Workflow Diagram

3.5 Main menu design:

A menu is a very common part of video games. it tells us about the game, the game will more understandable to the user, it describes how to use this game. A beautiful and user-friendly menu can attract users to play the game. No one wants a complex menu where users can get confused and lose their motivation to play the game. That's why in our Space infinity war game we are also using a very simple menu. Where on the top the game title will appear. In the middle part, there will be a "start playing" instruction. And at the bottom, there will be some control text about the game.

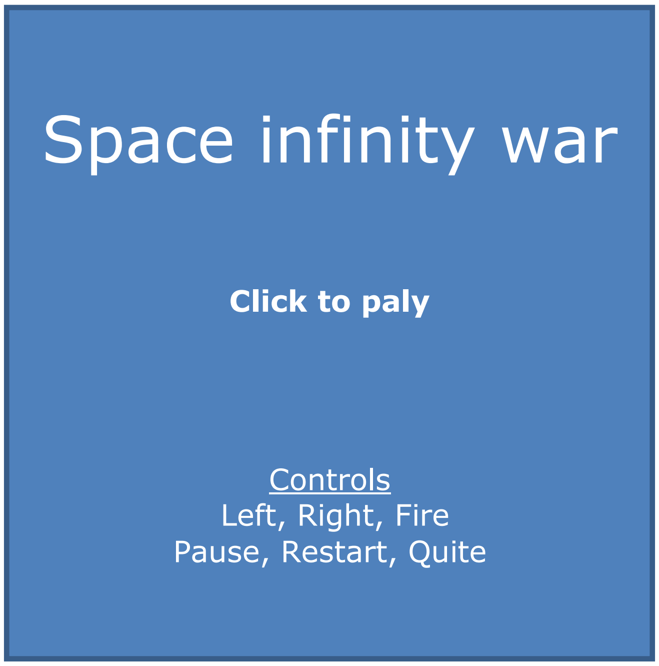


Figure 3.2: Main menu design.

3.6 Main game loop design:

A game loop (also called a main loop) is a loop where the code does three things:

- Handles events.
- Updates the game state.
- Draws the game state to the screen.

The game state is simply a way of referring to a set of values for all the variables in a game program. In many games, the game state includes the values in the variables that tracks the player’s health and position, the health and position of any enemies, which marks have been made on a board, the score, or whose turn it is. Whenever something happens like the player taking damage (which lowers their health value), or an enemy moves somewhere, or something happens in the game world we say that the game state has changed.
If you’ve ever played a game that let you save, the "save state" is the game state at the point that you’ve saved it. In most games, pausing the game will prevent the game state from changing. Since the game state is usually updated in response to events (such as mouse clicks or keyboard presses) or the passage of time, the game loop is constantly checking and re-checking many times a second for any new events that have happened. Inside the main loop is code that looks at which events have been created (with Pygame, this is done by calling the pygame.event.get () function). The main loop also has code that updates the game state based on which events have been created. This is usually called event handling.

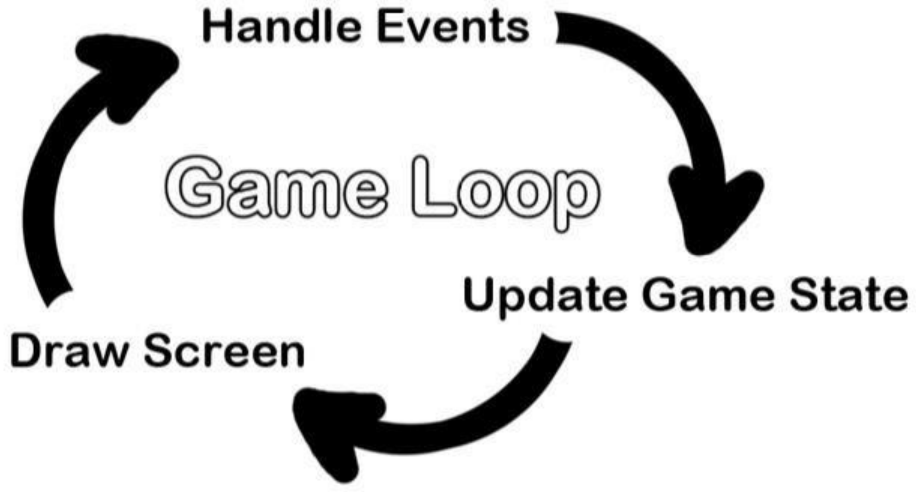


Figure 3.3: Main game loop.

3.7 Graphics design:

The ideal of beauty also concerns game developers: Graphics are an important measure of a game's worth because of the power of images and art to influence perception, emotion, and, ultimately, immersion in a game world--it's not just about good looks. Good graphic design makes games more trusted and more pleasurable to play and when done right, fades into the background so players can focus on the game. So we chose 2D png images for our game. The images are:

- Background image
- Title image
- Player Spacecraft images
- Bullet images
- Enemy Spaceship images
- Meteor images

Background image:

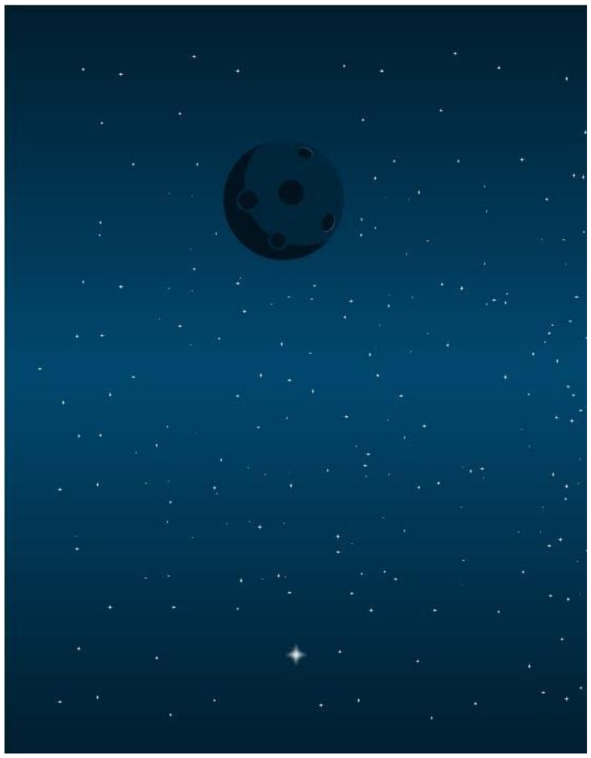


Figure 3.4: Background image

Title images:

Figure 3.5: Title image

Player Spacecraft images:

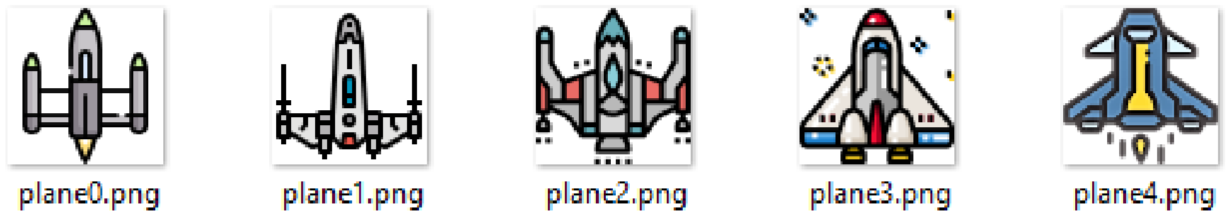


Figure 3.6: Player Spacecraft image

Bullet images:

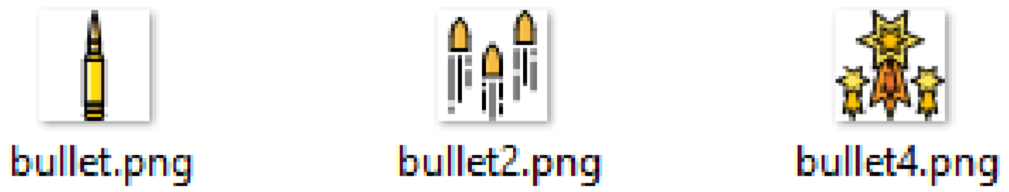


Figure 3.7: Bullet image

Enemy Spaceship images:

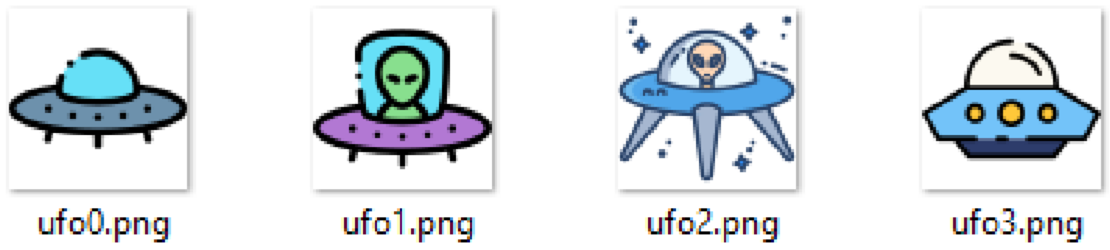


Figure 3.8: Enemy Spaceship image

Meteor images:

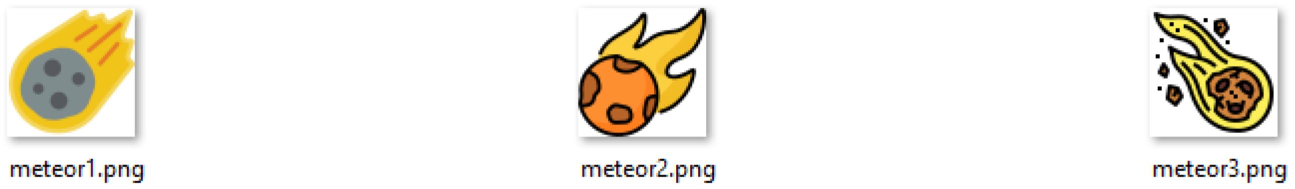


Figure 3.9: Meteor image

Investigator has a one-to-one relationship with Investigation, one-to-many relationship with Doctor, and many-to-many relationship with Patients.

3.8 Playing strategy:

The playing strategy of “Space infinity war” is very simple and straightforward. The Player will have their Spacecraft (Figure 3.6) on the bottom of the screen and on top of the game screen, the Enemy Spaceship (Figure 3.8) will appear. The Enemy spaceship will be randomly moving horizontally on the screen and vertically come down slowly.

The player can move their Spacecraft by pressing the left or right key and fire bullet (Figure 3.7) by pressing the Space bar. When the bullet meets the enemy point the collision happened and the player gets 100 points after that a new enemy comes to the random position. Get a high score is the main theme of this game. After every 1000 points, our game level will increase by one and the game will become harder at each level. After reaching level 3 the meteor wave (Figure 3.9) start and player get hit by any meteor, enemies will come down with fast speed. If an enemy spaceship vertically comes to our point and if the player fails to kill, the game will over.

3.9 Each level difficulties and achievements:

A game without difficulties and achievements is Valued less. After overcoming the obstacle of a game, a reward is a must. And it makes a player feel proud, and more interested to face new difficulties and get new achievements. Space infinity war has also difficulties and achievements:

Achievements:

- After each 1000 points Level will increase by +1
- Each level player get new Spacecraft
- Each 2 level player get new bullets

Difficulties:

- After level 1 the enemy moving speed will increase
- After level 3 the first meteor wave(Figure 3.9) will come
- After level 5 the Second meteor wave(Figure 3.9) will come

3.10 Conclusion:

In chapter 3, we have fully discussed the design of our project. In the next chapter, we will discuss our project implementation par

Chapter 4

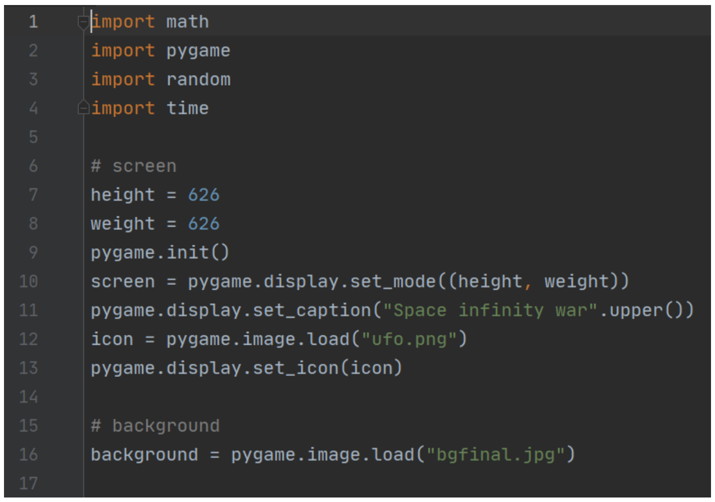
Implementation

4.1 Introduction

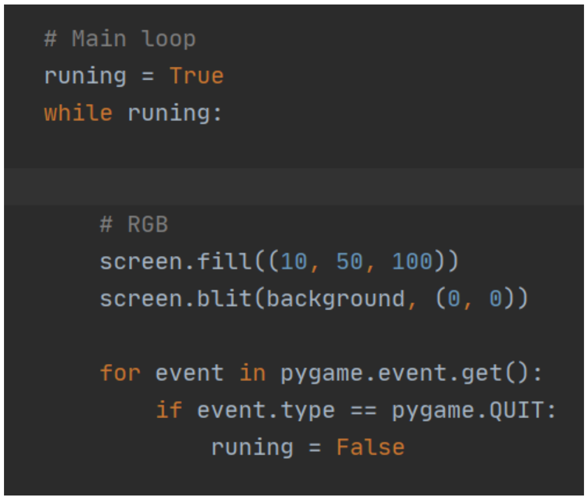
In this chapter, we are going to discuss the functionality of Space infinity war with various screenshots. Here, we discuss the functionality and working process of all panels. Thus in Section 4.2, we have discussed the Pygame initialization. In Section 4.3, we discussed the Main menu. In Section 4.4, we have discussed the input control and key. In Section 4.5, we have discussed Pause, Restart functions. In Section 4.6, we have discussed Player. In section 4.7, we have discussed the Enemies. In section 4.8, we have discussed the Collision. In section 4.9, we have discussed the Meteors. In section 4.10, we have discussed the Score and High Score. In section 4.11, we have discussed the Background music and sound. And the last we will discuss about Game over in section 4.12.

4.2 Pygame Initialization

In pygame we have to import pygame into our environment. And initialize the game window to start our project. And create the main game loop where the window will show on-screen continually. Until the user quits the window.



```python
import math
import pygame
import random
import time

# screen
height = 626
weight = 626
pygame.init()
screen = pygame.display.set_mode((height, weight))
pygame.display.set_caption("Space infinity war".upper())
icon = pygame.image.load("ufo.png")
pygame.display.set_icon(icon)

# background
background = pygame.image.load("bgfinal.jpg")

```

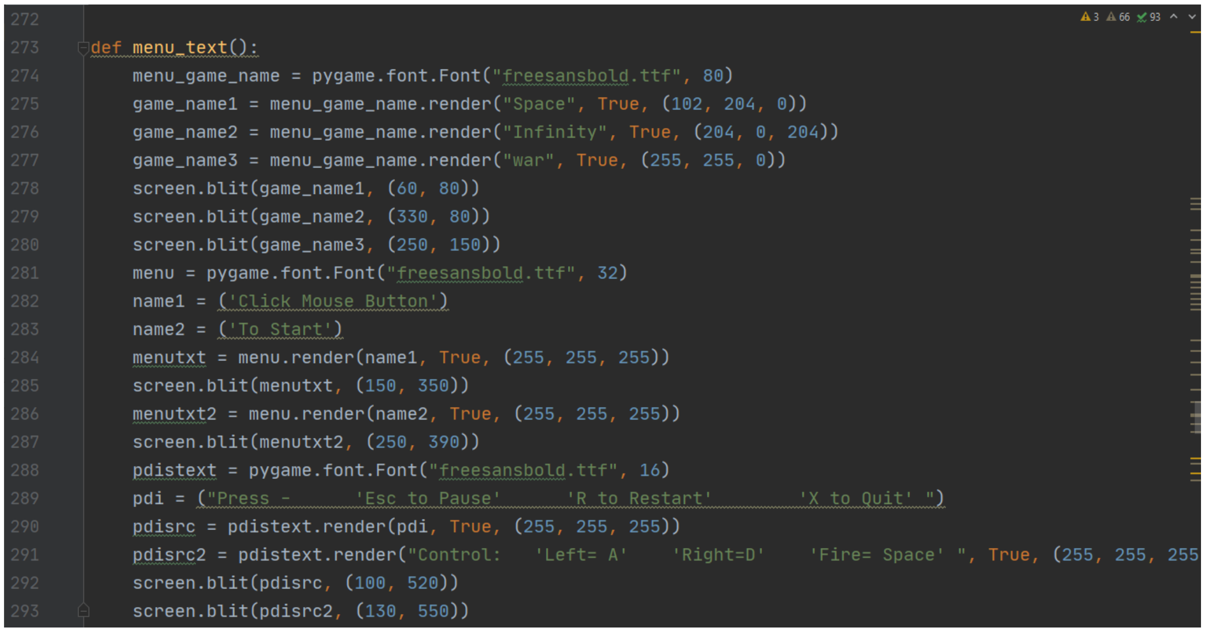


```python
# Main loop
runing = True
while runing:


    # RGB
    screen.fill((10, 50, 100))
    screen.blit(background, (0, 0))

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            runing = False
```

Figure 4.1: Pygame initialization.

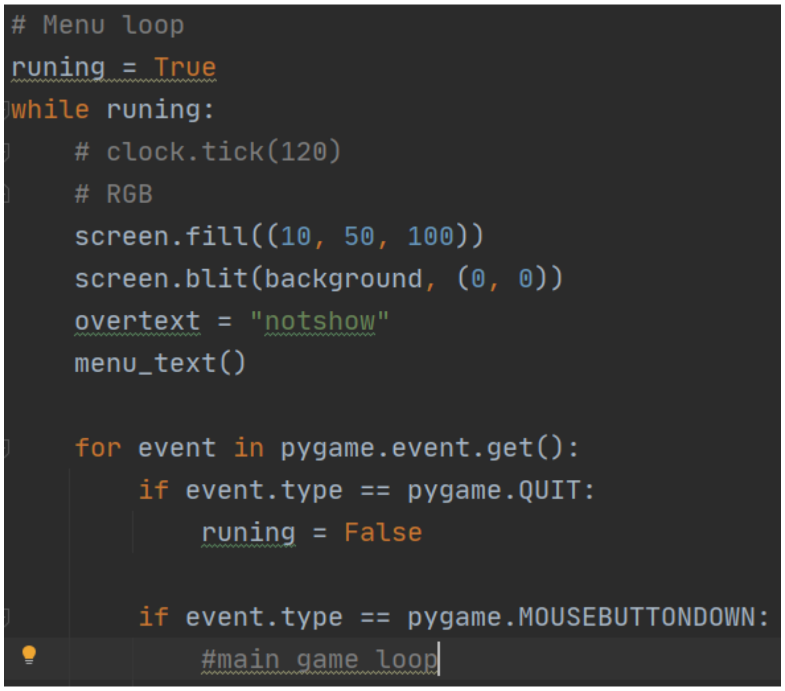
4.3 Main menu

In chapter 3.5 we have discussed our main menu design. We write a function for the text area of our menu. And use the main menu loop where the loop runs until the user clicks to play and the loop will jump into the nested main game loop.



```python
def menu_text():
    menu_game_name = pygame.font.Font("freesansbold.ttf", 80)
    game_name1 = menu_game_name.render("Space", True, (102, 204, 0))
    game_name2 = menu_game_name.render("Infinity", True, (204, 0, 204))
    game_name3 = menu_game_name.render("war", True, (255, 255, 0))
    screen.blit(game_name1, (60, 80))
    screen.blit(game_name2, (330, 80))
    screen.blit(game_name3, (250, 150))
    menu = pygame.font.Font("freesansbold.ttf", 32)
    name1 = ('Click Mouse Button')
    name2 = ('To Start')
    menutxt = menu.render(name1, True, (255, 255, 255))
    screen.blit(menutxt, (150, 350))
    menutxt2 = menu.render(name2, True, (255, 255, 255))
    screen.blit(menutxt2, (250, 390))
    pdistext = pygame.font.Font("freesansbold.ttf", 16)
    pdi = ("Press -       'Esc to Pause'      'R to Restart'        'X to Quit' ")
    pdisrc = pdistext.render(pdi, True, (255, 255, 255))
    pdisrc2 = pdistext.render("Control:   'Left= A'    'Right=D'    'Fire= Space' ", True, (255, 255, 255
    screen.blit(pdisrc, (100, 520))
    screen.blit(pdisrc2, (130, 550))
```

Figure 4.2: Menu text function.



```python
# Menu loop
runing = True
while runing:
    # clock.tick(120)
    # RGB
    screen.fill((10, 50, 100))
    screen.blit(background, (0, 0))
    overtext = "notshow"
    menu_text()

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            runing = False

        if event.type == pygame.MOUSEBUTTONDOWN:
            #main game loop
```

Figure 4.3: Menu Loop.

Output:

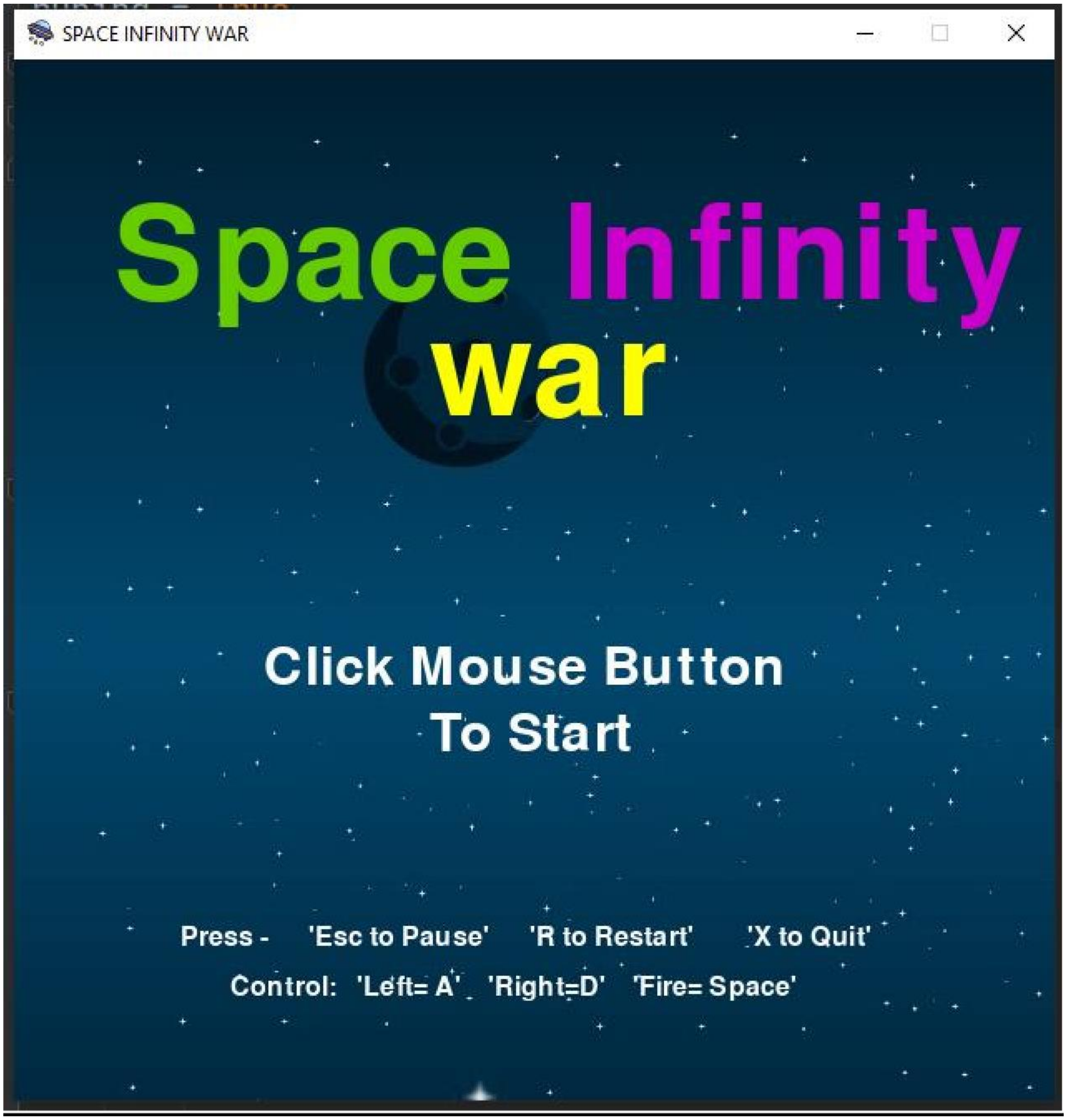
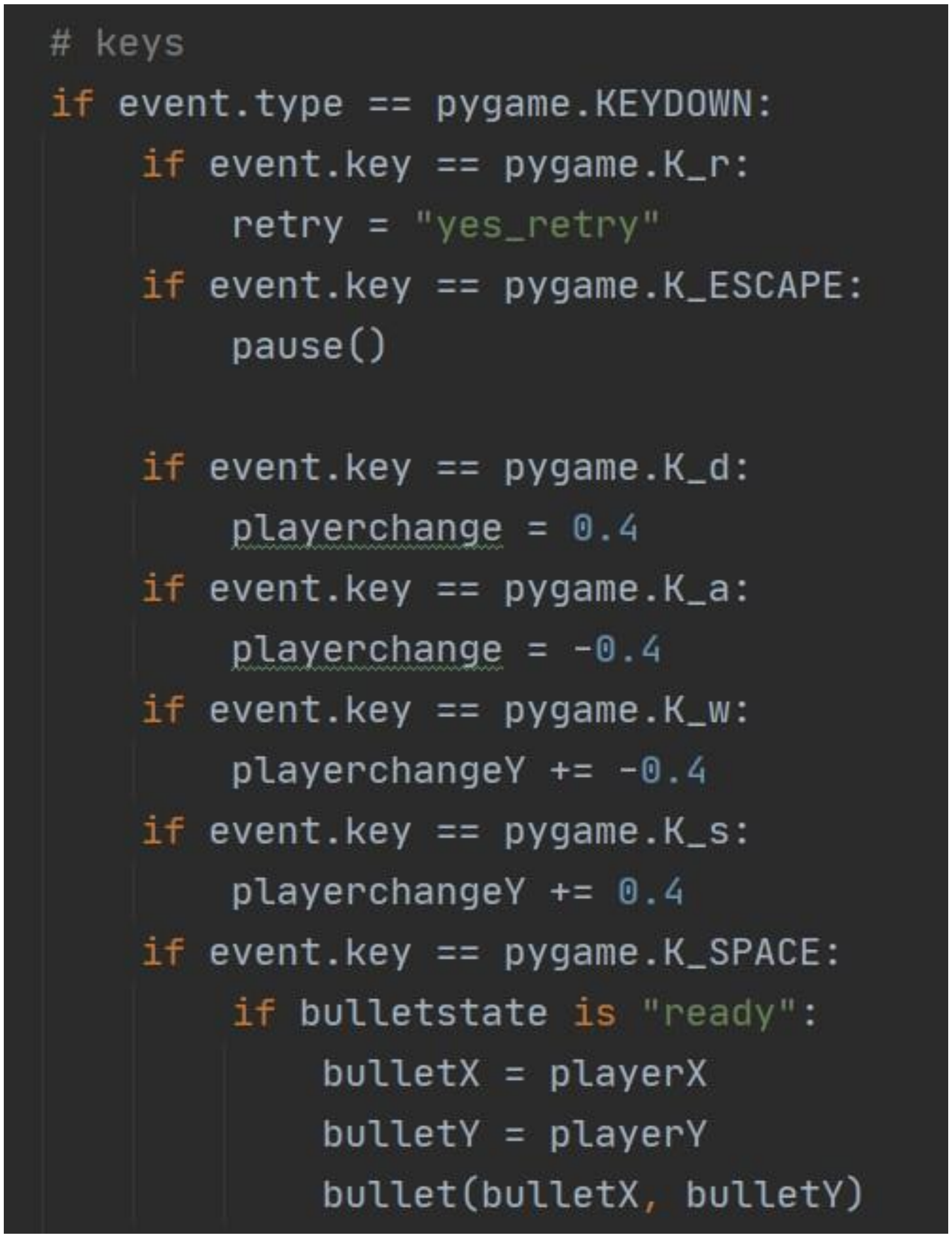

SPACE INFINITY WAR
Space Infinity
war
Click Mouse Button
To Start
Press - 'Esc to Pause' 'R to Restart' 'X to Quit'
Control: 'Left= A' 'Right=D' 'Fire= Space'

Figure 4.4: Main Menu.

4.4 Input controls and key

Here   we are going to setup our gaming controls and keys. Using our keyboard and mouse we can play our game.



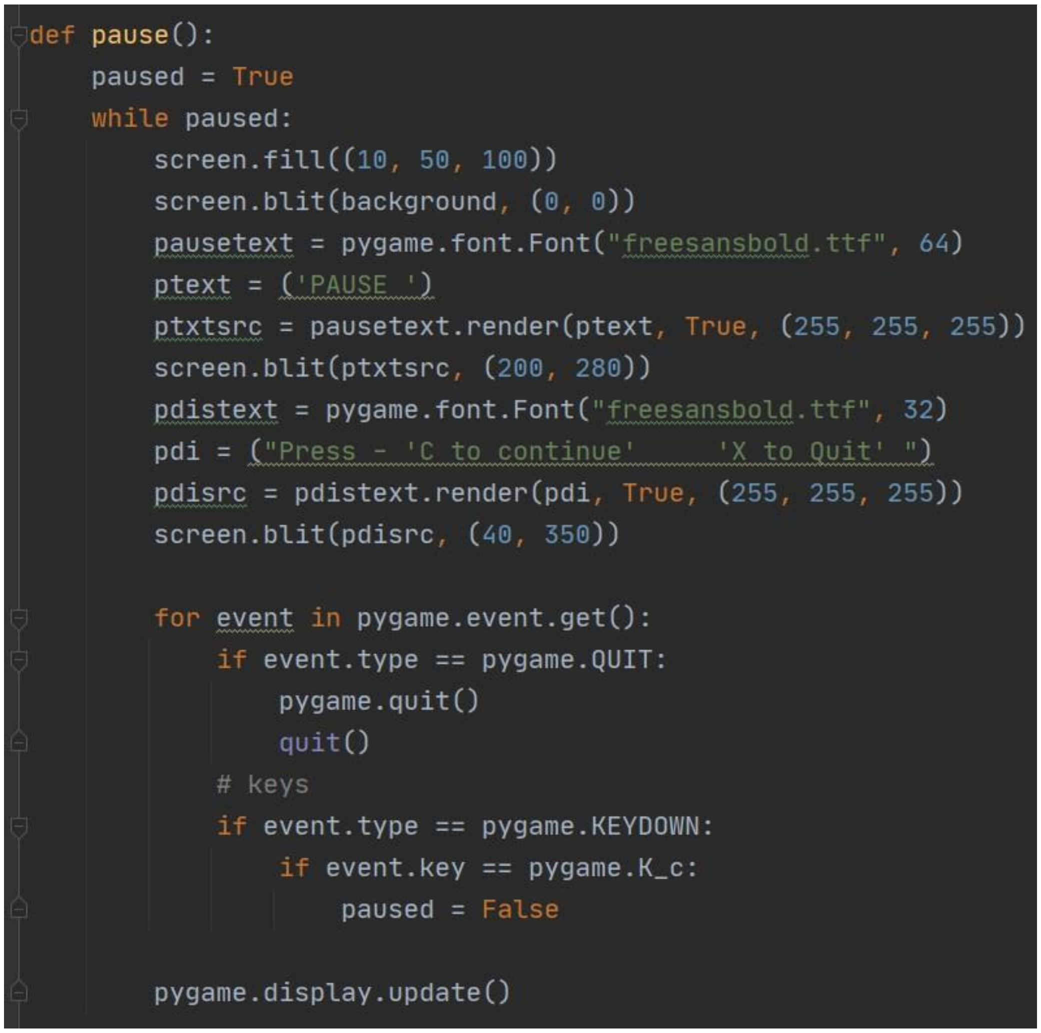
```
# keys
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_r:
        retry = "yes_retry"
    if event.key == pygame.K_ESCAPE:
        pause()

    if event.key == pygame.K_d:
        playerchange = 0.4
    if event.key == pygame.K_a:
        playerchange = -0.4
    if event.key == pygame.K_w:
        playerchangeY += -0.4
    if event.key == pygame.K_s:
        playerchangeY += 0.4
    if event.key == pygame.K_SPACE:
        if bulletstate is "ready":
            bulletX = playerX
            bulletY = playerY
            bullet(bulletX, bulletY)
```

Figure 4.5: Input control and key.

4.5 Pause and Restart functions

Pause and restart functions are very common in game development. The game is incomplete without a pause and restarts function. So we create a function to pause the game when the user press the “ESC” key the main game loop will be terminated to pause the function. And to restart the game we use a logical operation. When the user press “R” to restart the game and the main game loop will break and the main menu will run again.

Pause:



```
def pause():
    paused = True
    while paused:
        screen.fill((10, 50, 100))
        screen.blit(background, (0, 0))
        pausetext = pygame.font.Font("freesansbold.ttf", 64)
        ptext = ('PAUSE ')
        ptxtsrc = pausetext.render(ptext, True, (255, 255, 255))
        screen.blit(ptxtsrc, (200, 280))
        pdistext = pygame.font.Font("freesansbold.ttf", 32)
        pdi = ("Press - 'C to continue'     'X to Quit' ")
        pdisrc = pdistext.render(pdi, True, (255, 255, 255))
        screen.blit(pdisrc, (40, 350))

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
            # keys
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_c:
                    paused = False

        pygame.display.update()
```

Figure 4.6: Pause function.

Output:

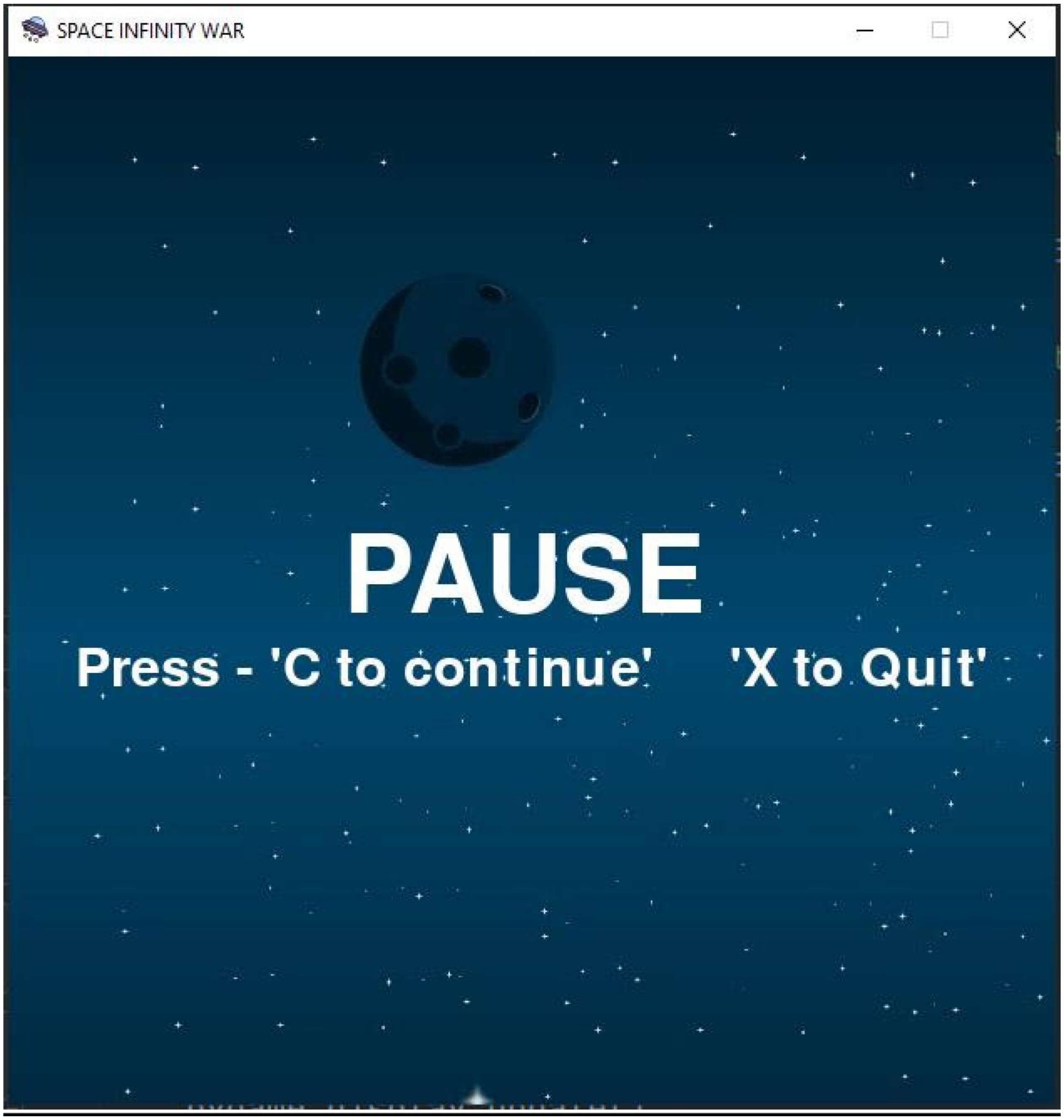
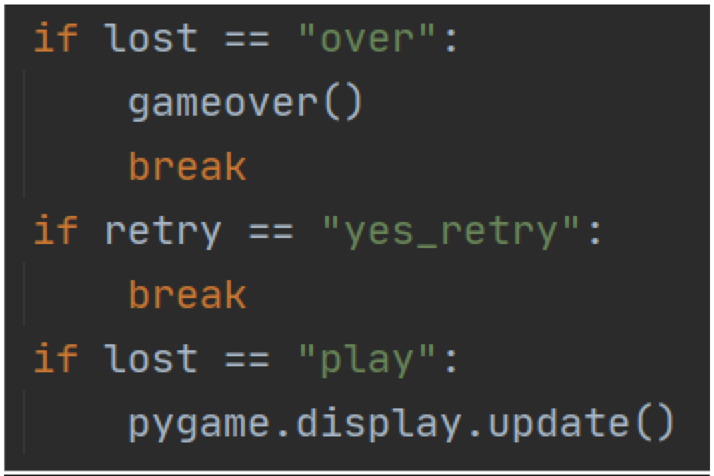


Figure 4.7: Pause game.

Restart:



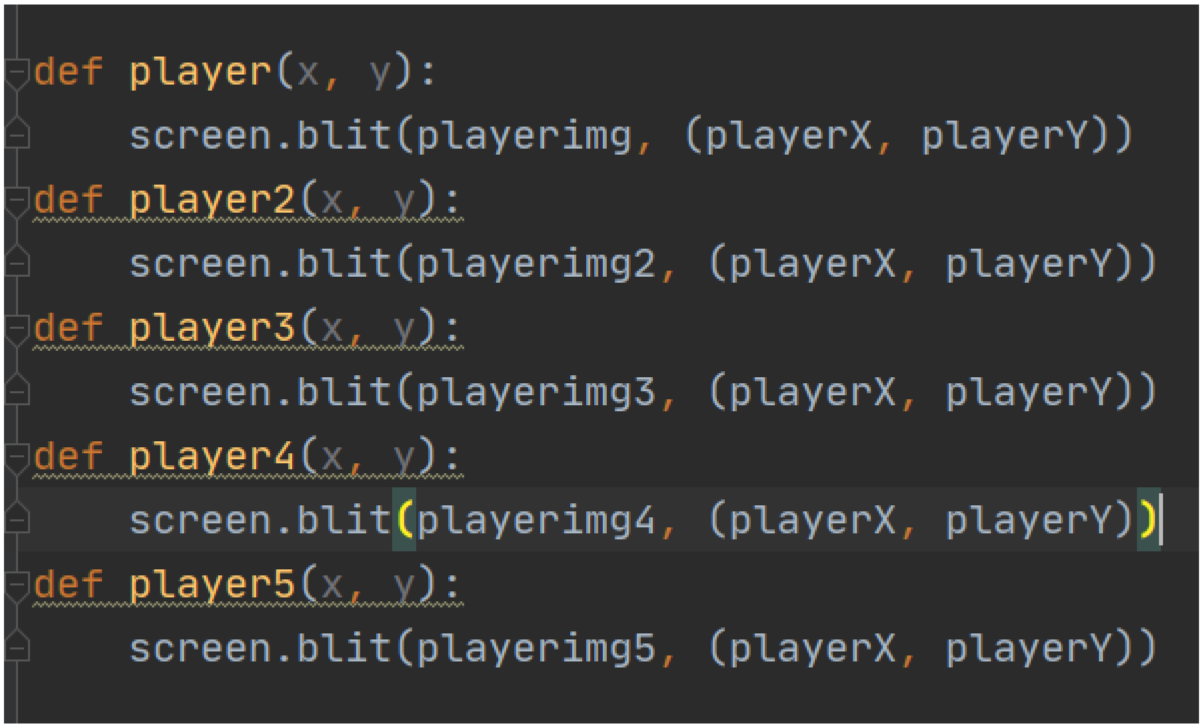
```
if lost == "over":
    gameover()
    break
if retry == "yes_retry":
    break
if lost == "play":
    pygame.display.update()
```

Figure 4.8: Restart game.

The output of restart will be same as main menu figure 4.4.

4.6 Player functions:

In chapter 3 we have watched our player images in figure 3.6. And we create function for each spacecraft. Those functions will call in main game loop.



```
def player(x, y):
    screen.blit(playerimg, (playerX, playerY))
def player2(x, y):
    screen.blit(playerimg2, (playerX, playerY))
def player3(x, y):
    screen.blit(playerimg3, (playerX, playerY))
def player4(x, y):
    screen.blit(playerimg4, (playerX, playerY))
def player5(x, y):
    screen.blit(playerimg5, (playerX, playerY))
```

Figure 4.9: Player functions.

Output:

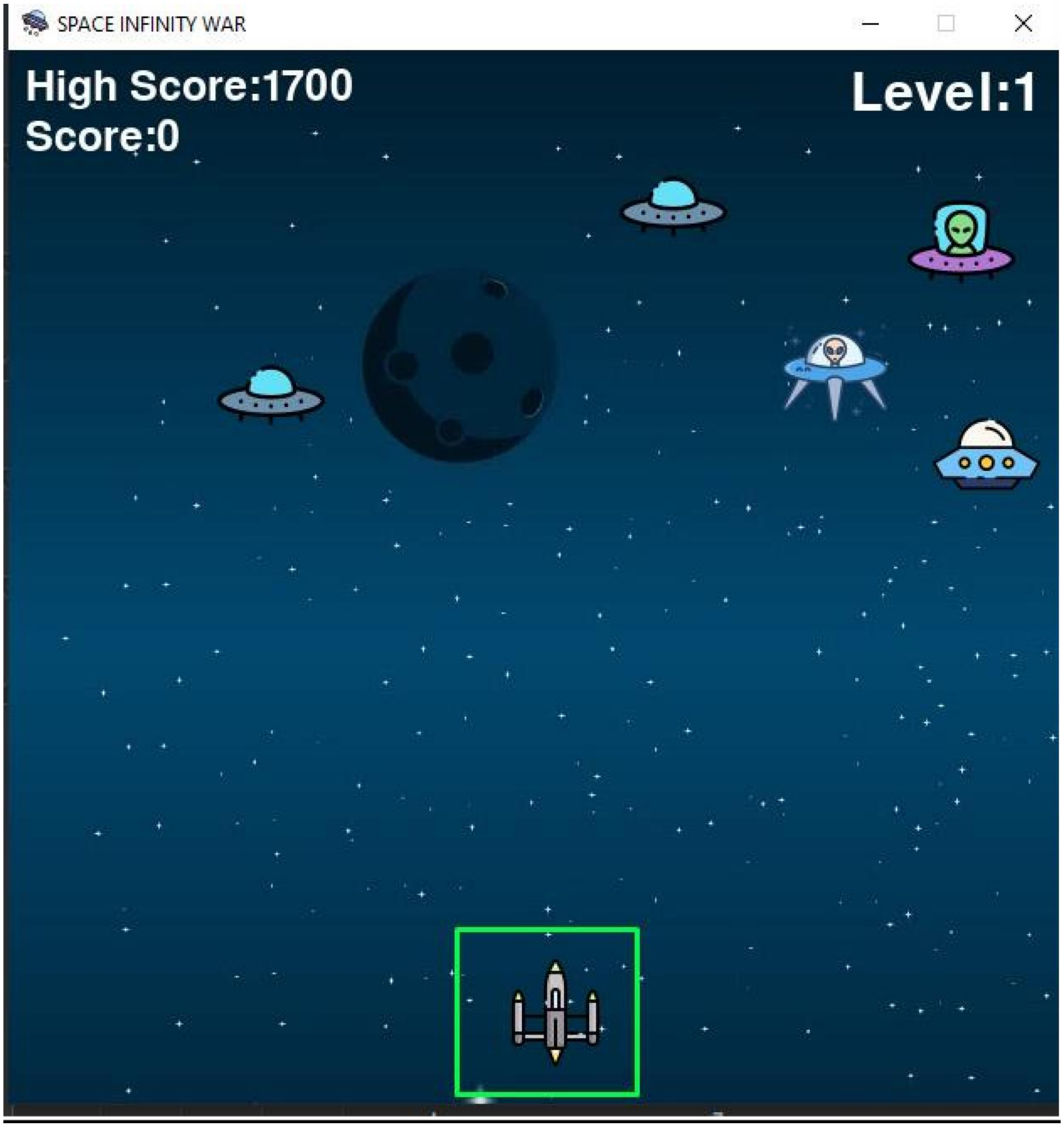
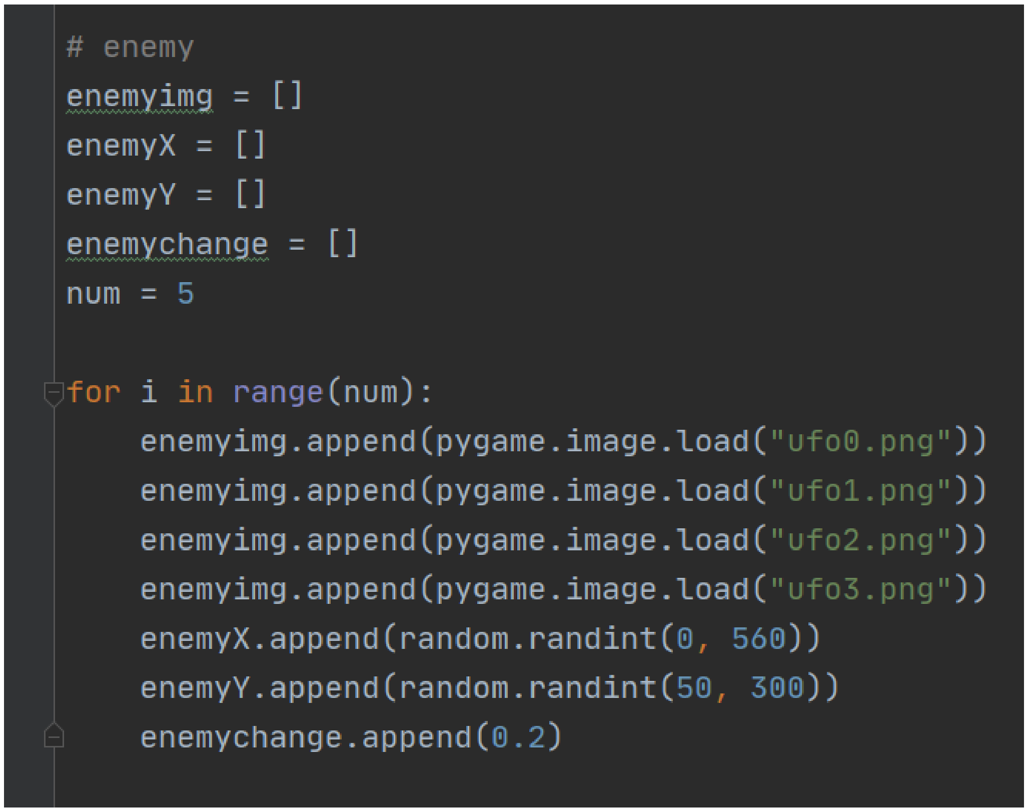


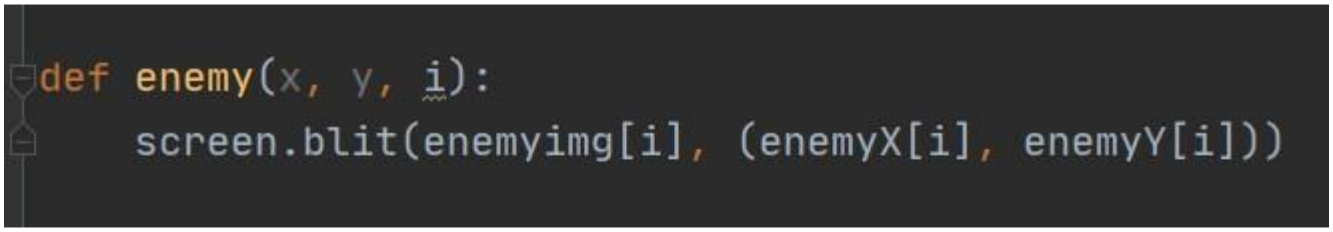
Figure 4.9: Player.

4.7 Enemies:

We create multiple enemies and use for loop store the values of enemy into enemy arrays. And create the enemy function where each element will shows up on the game window.



```
# enemy
enemyimg = []
enemyX = []
enemyY = []
enemychange = []
num = 5

for i in range(num):
    enemyimg.append(pygame.image.load("ufo0.png"))
    enemyimg.append(pygame.image.load("ufo1.png"))
    enemyimg.append(pygame.image.load("ufo2.png"))
    enemyimg.append(pygame.image.load("ufo3.png"))
    enemyX.append(random.randint(0, 560))
    enemyY.append(random.randint(50, 300))
    enemychange.append(0.2)
```



```
def enemy(x, y, i):
    screen.blit(enemyimg[i], (enemyX[i], enemyY[i]))
```

Figure 4.10: Enemies function.

Output:

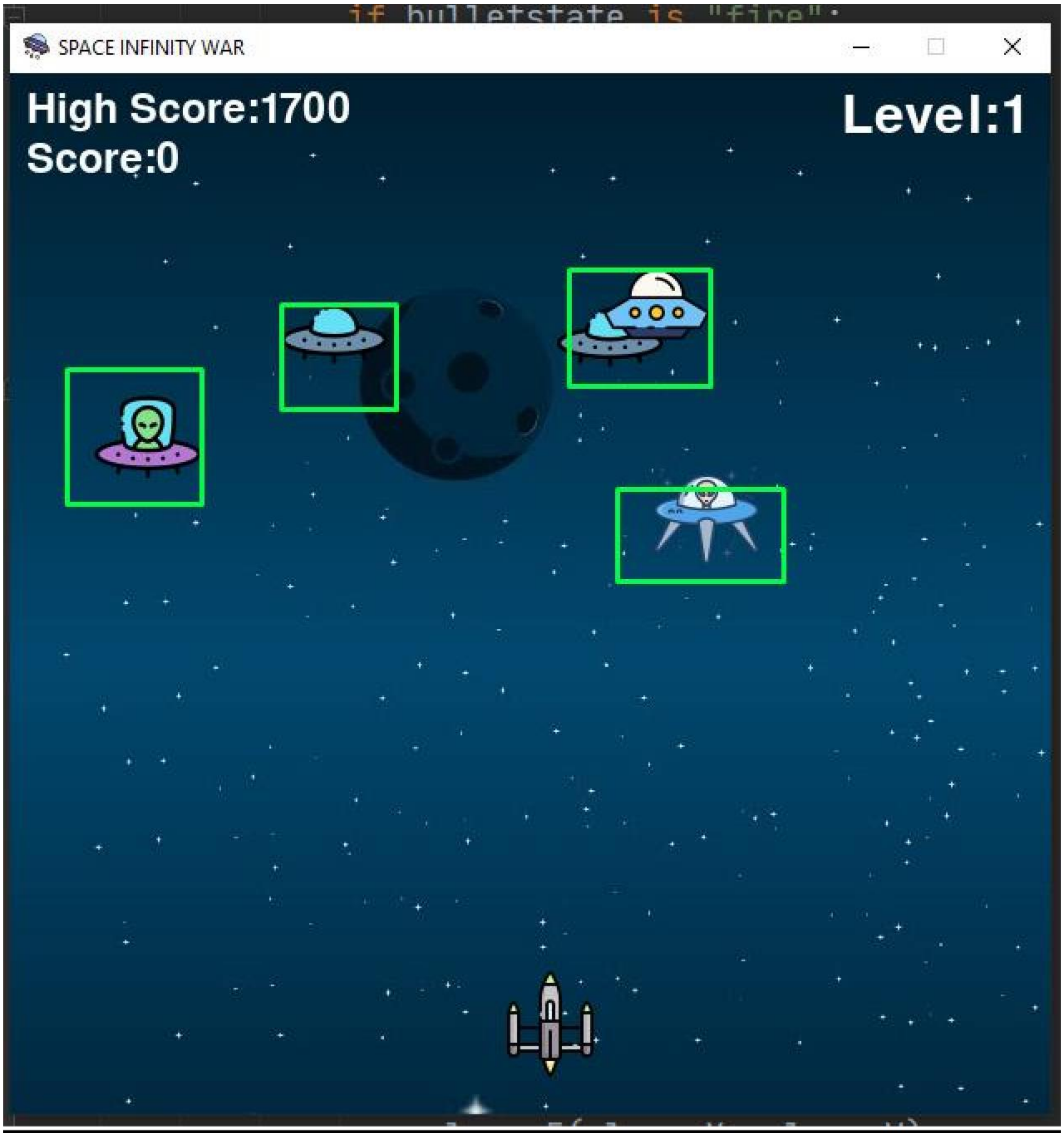
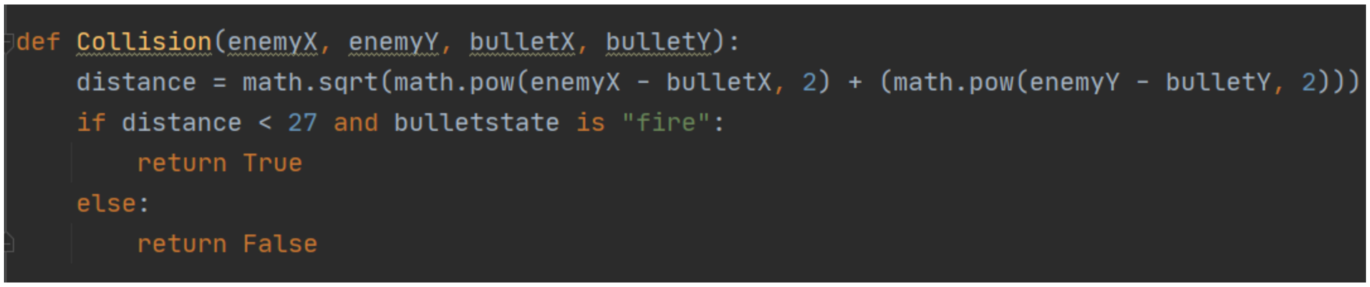


Figure 4.11: Enemies.

4.8 Collision:

In Collision part we use math library and create a function where we implement a collision equation of two object.

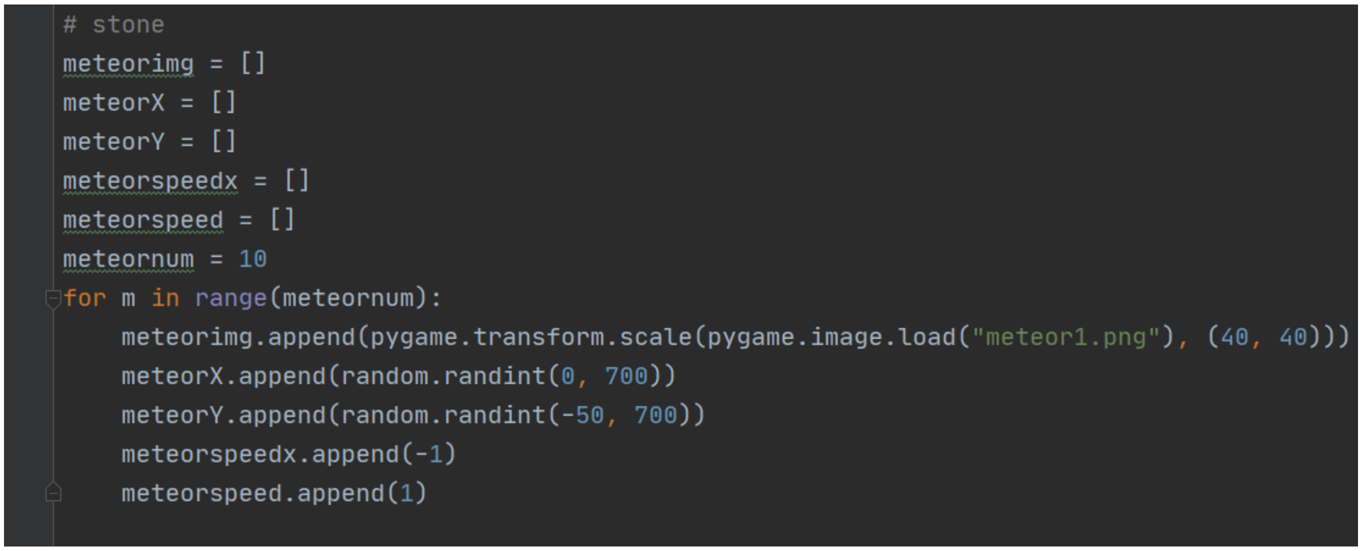


```python
def Collision(enemyX, enemyY, bulletX, bulletY):
    distance = math.sqrt(math.pow(enemyX - bulletX, 2) + (math.pow(enemyY - bulletY, 2)))
    if distance < 27 and bulletstate is "fire":
        return True
    else:
        return False
```

Figure 4.12: Collision.

4.9 Meteors:

In space infinity war meteor are show on chapter 3, Figure 3.9. The meteor wave will active after reach the level 3. And it move left to right. Hit by a meteor will make you trouble



```python
# stone
meteorimg = []
meteorX = []
meteorY = []
meteorspeedx = []
meteorspeed = []
meteornum = 10
for m in range(meteornum):
    meteorimg.append(pygame.transform.scale(pygame.image.load("meteor1.png"), (40, 40)))
    meteorX.append(random.randint(0, 700))
    meteorY.append(random.randint(-50, 700))
    meteorspeedx.append(-1)
    meteorspeed.append(1)
```

Figure 4.13: Meteors function.

<u>Output:</u>

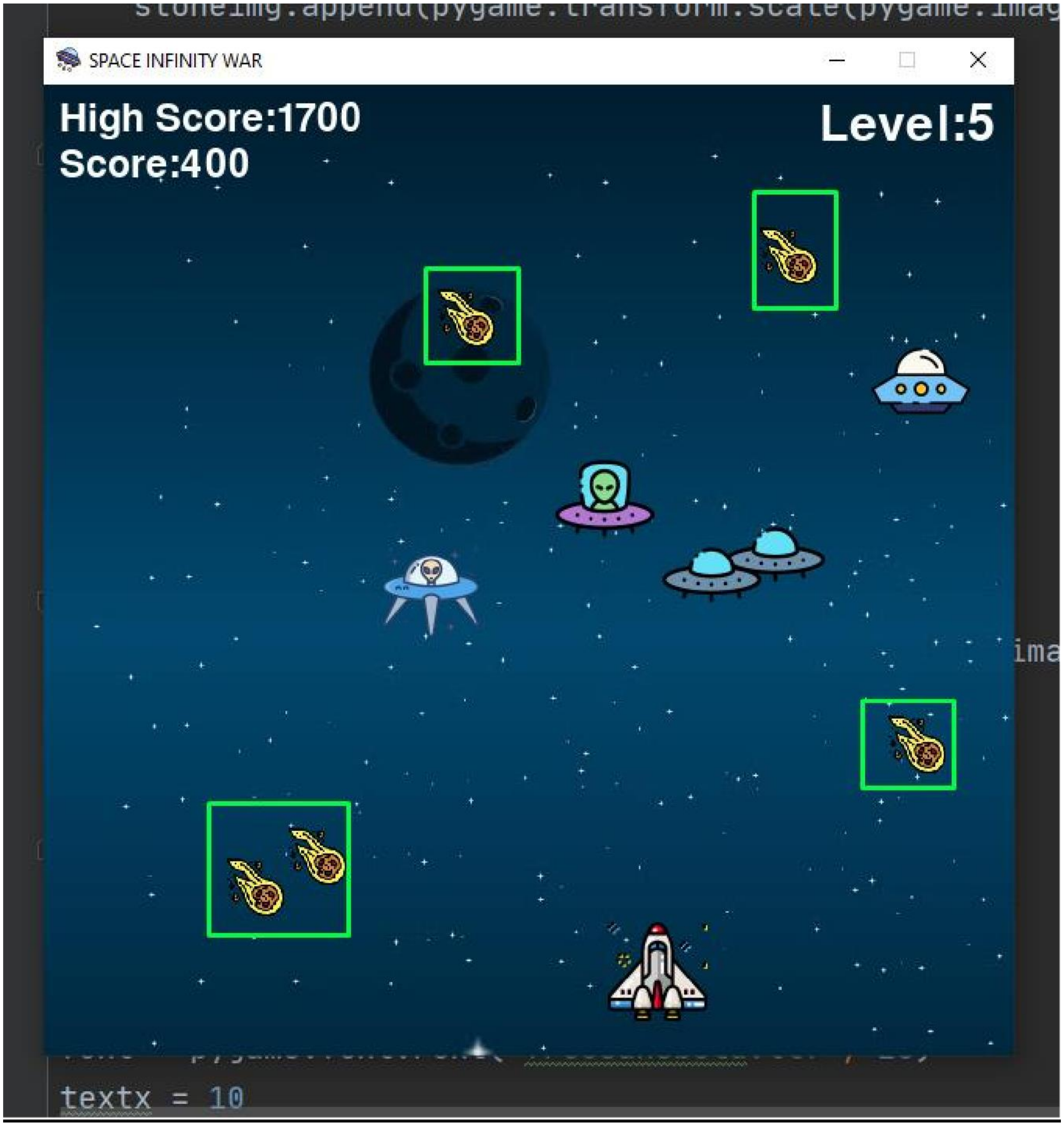
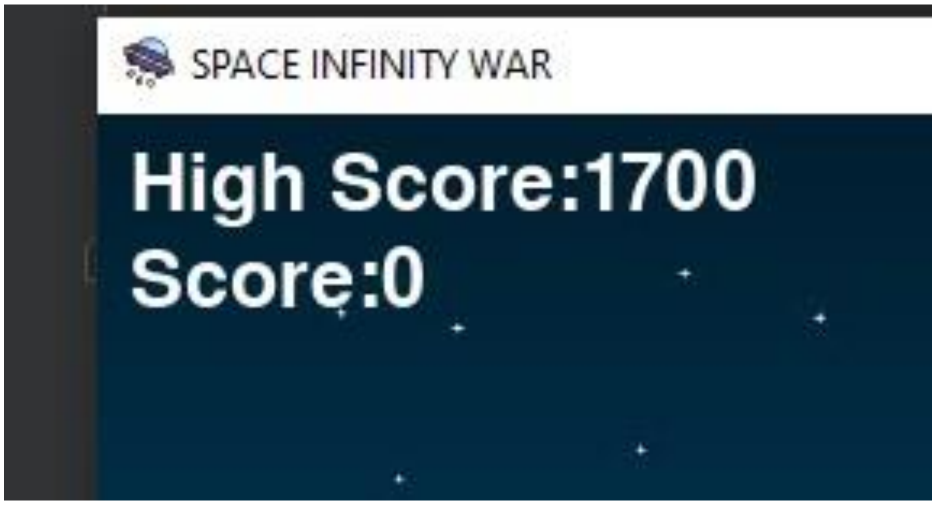


Figure 4.14: Meteors.

4.10 Score and High score:

Player will get 100 score for killing each enemies. And in every game paly will create a score record the highest point will mark as a high score.



SPACE INFINITY WAR

High Score:1700
Score:0

Figure 4.15: Score and high score.

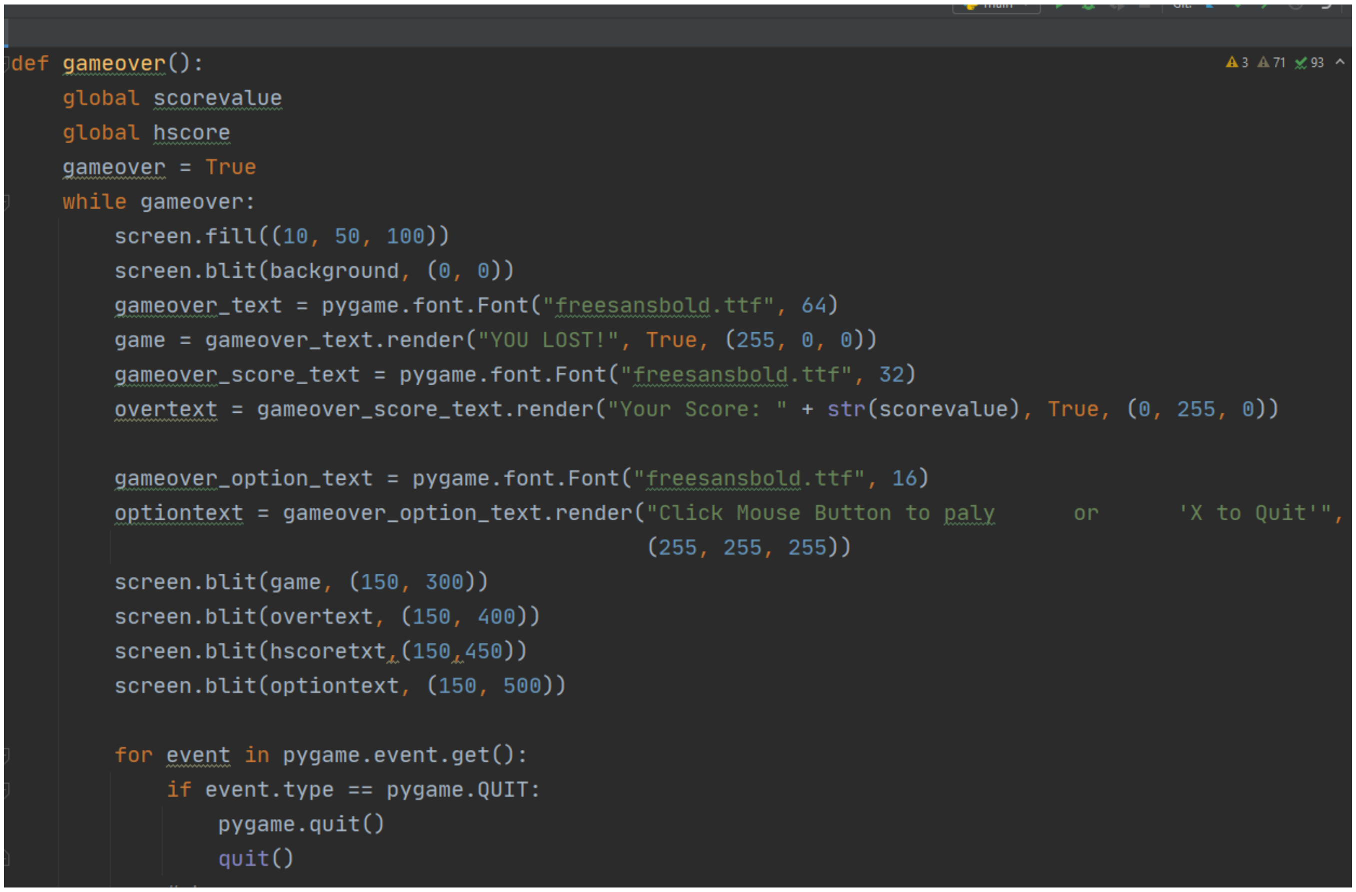
4.11 Background music and sound:

In a video game Background music and sound is must. While it's hard to say how background music affects players' performance, it clearly helps them remember a game better. A remarkable soundtrack leaves a deep trace in the players' minds, helping them recall the game's plot and events even if they haven't played it for years. So we use pygame Mixer method to add music and sound to our space infinity war game.

1. Starting the mixer mixer.init()
2. Loading the song. mixer.music.load("song.mp3")
3. Setting the volume. mixer.music.set_volume(0.7)
4. Start playing the song. mixer.music.play()

4.12 Game over:

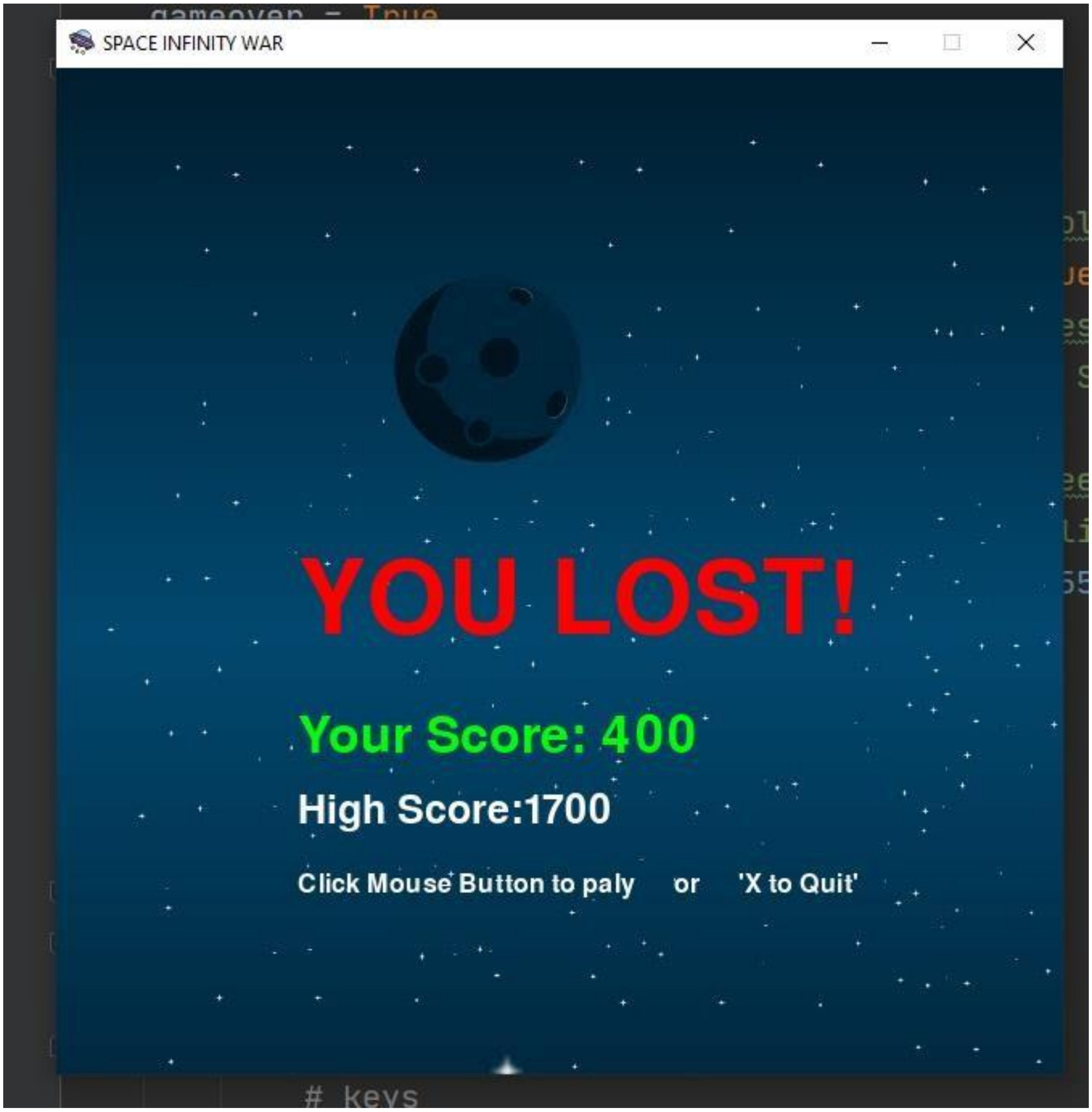
When a player lose to defeat the enemies and when the enemies come to the player point the game over function will run.



```
def gameover():
    global scorevalue
    global hscore
    gameover = True
    while gameover:
        screen.fill((10, 50, 100))
        screen.blit(background, (0, 0))
        gameover_text = pygame.font.Font("freesansbold.ttf", 64)
        game = gameover_text.render("YOU LOST!", True, (255, 0, 0))
        gameover_score_text = pygame.font.Font("freesansbold.ttf", 32)
        overtext = gameover_score_text.render("Your Score: " + str(scorevalue), True, (0, 255, 0))

        gameover_option_text = pygame.font.Font("freesansbold.ttf", 16)
        optiontext = gameover_option_text.render("Click Mouse Button to paly      or      'X to Quit'",
                                                 (255, 255, 255))
        screen.blit(game, (150, 300))
        screen.blit(overtext, (150, 400))
        screen.blit(hscoretxt,(150,450))
        screen.blit(optiontext, (150, 500))

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
```

Figure 4.16: Game over function.

Output:



YOU LOST!

Your Score: 400

High Score:1700

Click Mouse Button to paly or 'X to Quit'

Figure 4.17: Game.

4.1 Conclusion

In chapter 4, we have discussed the implementation of our project with screenshots. In the next chapter, we will discuss the future improvements that will our project more user-friendly and efficient.

Chapter 5

Future Works and Conclusion

5.1 Introduction

Many different adaptations, tests, and experiments have been left for the future due to lack of time (e.g. the experiments with real data are usually very time consuming, requiring even days to finish a single run). Future work concerns deeper analysis of particular mechanisms, new proposals to try different methods, or simply curiosity. There are some ideas that we will have liked to try, which will be discussed in this chapter.

5.2 Future Work

The following ideas could be tested:

1. We could develop it as a Browser game.
2. We could develop a Boss fight level.
3. We could add more graphics detail.
4. We could develop an option to see other player's high scores.

5.3 Conclusion

In this project, we developed a Space invader game for children. We want to create a game that has a very interesting mix of inputs. In order to ensure that the game is fun, we want the difficulty of the game to scale linearly with the length of time/ skill of the player. We also want to ensure that we have as many enemieson screen as possible by the end. We hope that once the dust settles, we won’t have just a barebones game, but a fun and challenging experience that lives up to the name of the classic Space Invaders.

References

[1] Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming 2nd Edition[Eric matthes]

[2] Game Development Using Python by [James R. Parker]

[3]Beginning Game Development with Python and Pygame: From Novice to Professional (Expert's Voice) by [will Mcgugan]

[4] https://www.pygame.org/news

[5] https://files.eric.ed.gov/fulltext/EJ1176181.pdf

[6]https://srcd.onlinelibrary.wiley.com/doi/full/10.1002/sop2.3?fbclid=IwAR0iwmyuTqy49ZyUiIgHeuKZDUE-zn1zLM1mpYV2ysckvWBxabsF5XMyYUI

[7] https://realpython.com/pygame-a-primer/https://www.w3schools.com/css/css_boxmodel.asp

[8] https://www.pygame.org/docs/ref/mixer.htmlhttps://www.typescriptlang.org/docs

[9] https://www.researchgate.net/publication/315684797_Advertising_in_digital_games_targeted_to_children

[10]https://www.researchgate.net/publication/260815121_Role_of_the_Game_in_the_Development_of_Preschool_Child

[11]https://srcd.onlinelibrary.wiley.com/doi/full/10.1002/sop2.3

[12]https://files.eric.ed.gov/fulltext/EJ1176181.pdf