PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100 Feet Ring Road, BSK III Stage, Bengaluru-560 085

Department of Computer Science & Engineering

Title: Space Invaders

Team member details:

PES1UG21CS206 – GAUTHAM P ATREYAS

PES1UG21CS281 - KISHAN BALAJI S

PES1UG21CS485 – REHAN SAYED

ABSTRACT

Pygame is a cross-platform set of python modules designed for writing games. It includes computer graphics and sound libraries designed to be used with the Python Programming Language

The tkinter package is a python package which is used to make graphical user interfaces.

In this project Tkinter is used to make interface of the main menu of the game, “Space Invaders”. Clicking on the “PLAY” button directs the program to open the pygame window where the game begins. There are three levels including one boss level. Each level contains the mission of the level at the bottom of the screen.

TABLE OF CONTENTS

- Introduction
- Design/Implementation
- Testing
- Result and Analysis
- Conclusions & future enhancements
- References

INTRODUCTION

Space Invaders is a Japanese shooting video game developed by Tomohiro Nishikado and released in 1978 by Taito. 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 re-released in various forms.

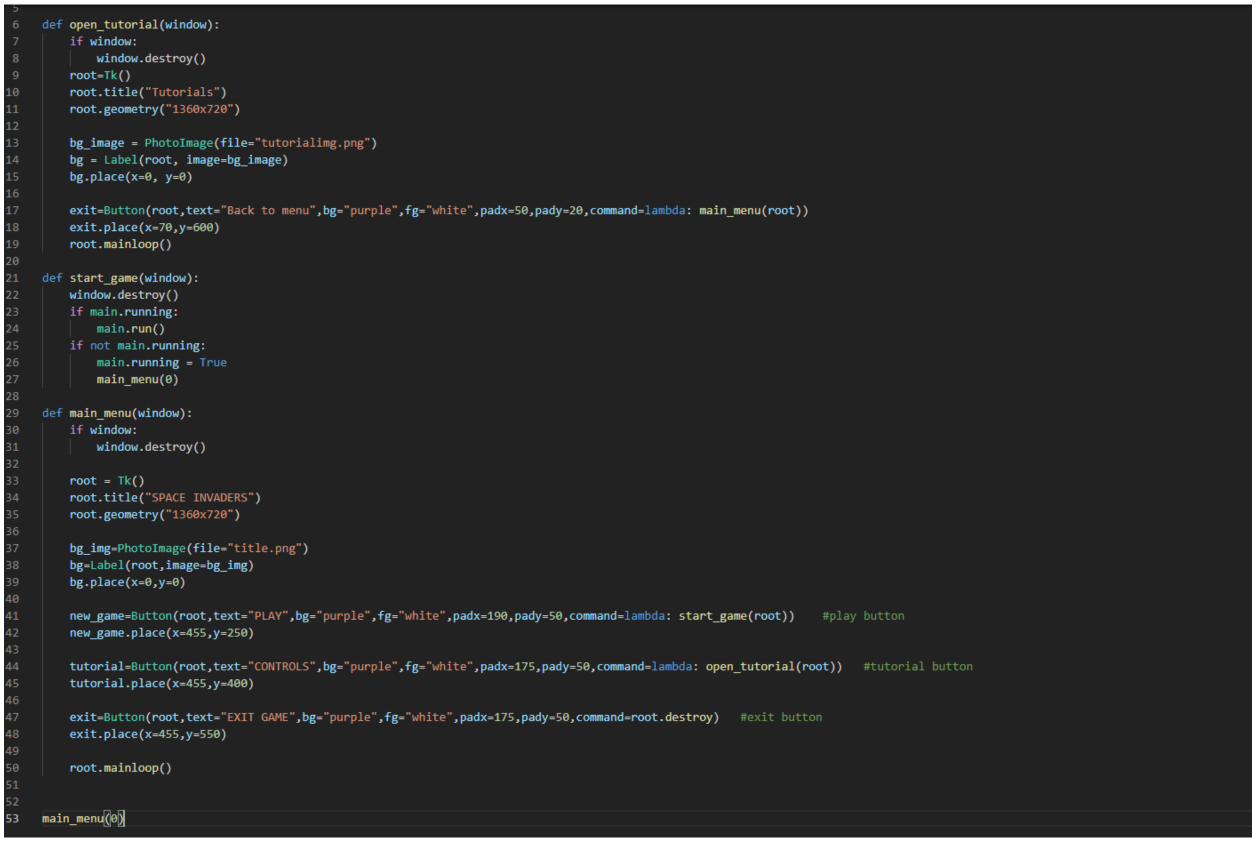
We have a created our game based off of the original, with many health system for the player; if the player takes too much damage, he will die and lose the game. There are three levels, each one increasing in difficulty. The mission of each level is shown at the bottom of the screen next to the player health bar.

Defeat the enemy boss and claim victory!

DESIGN/IMPLEMENTATION

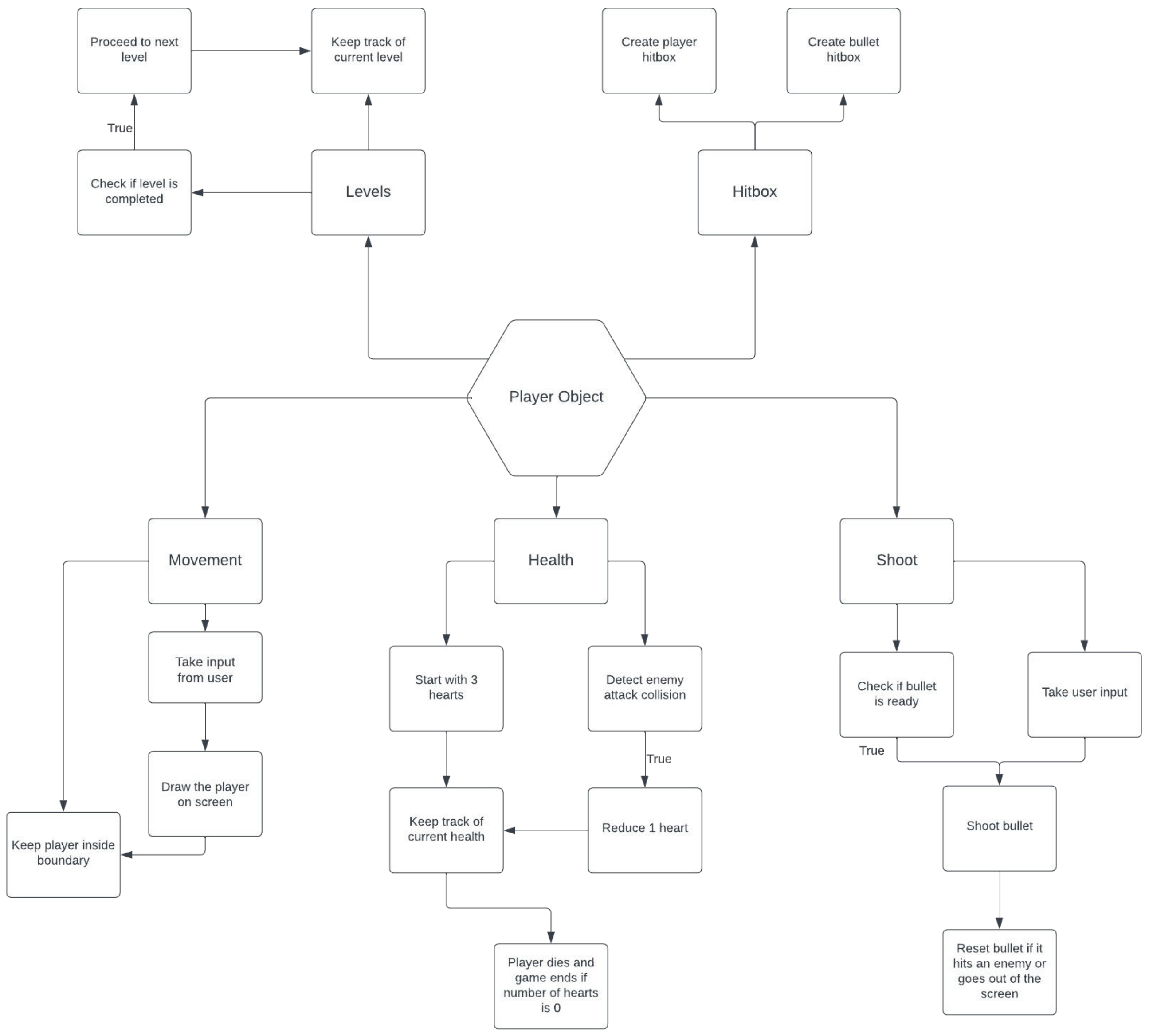
The entire project – code and images – can be downloaded from the following link.

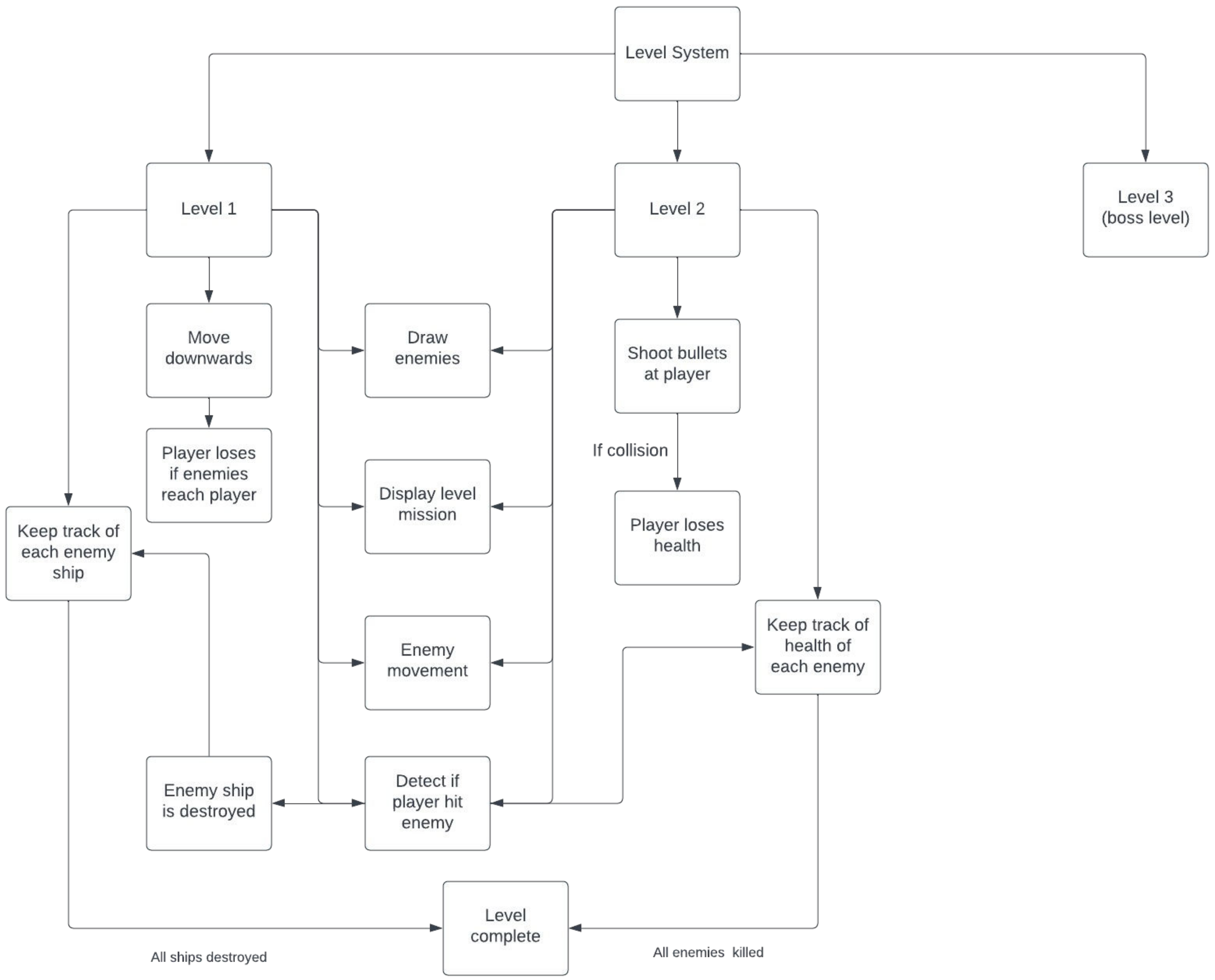
Menu file -

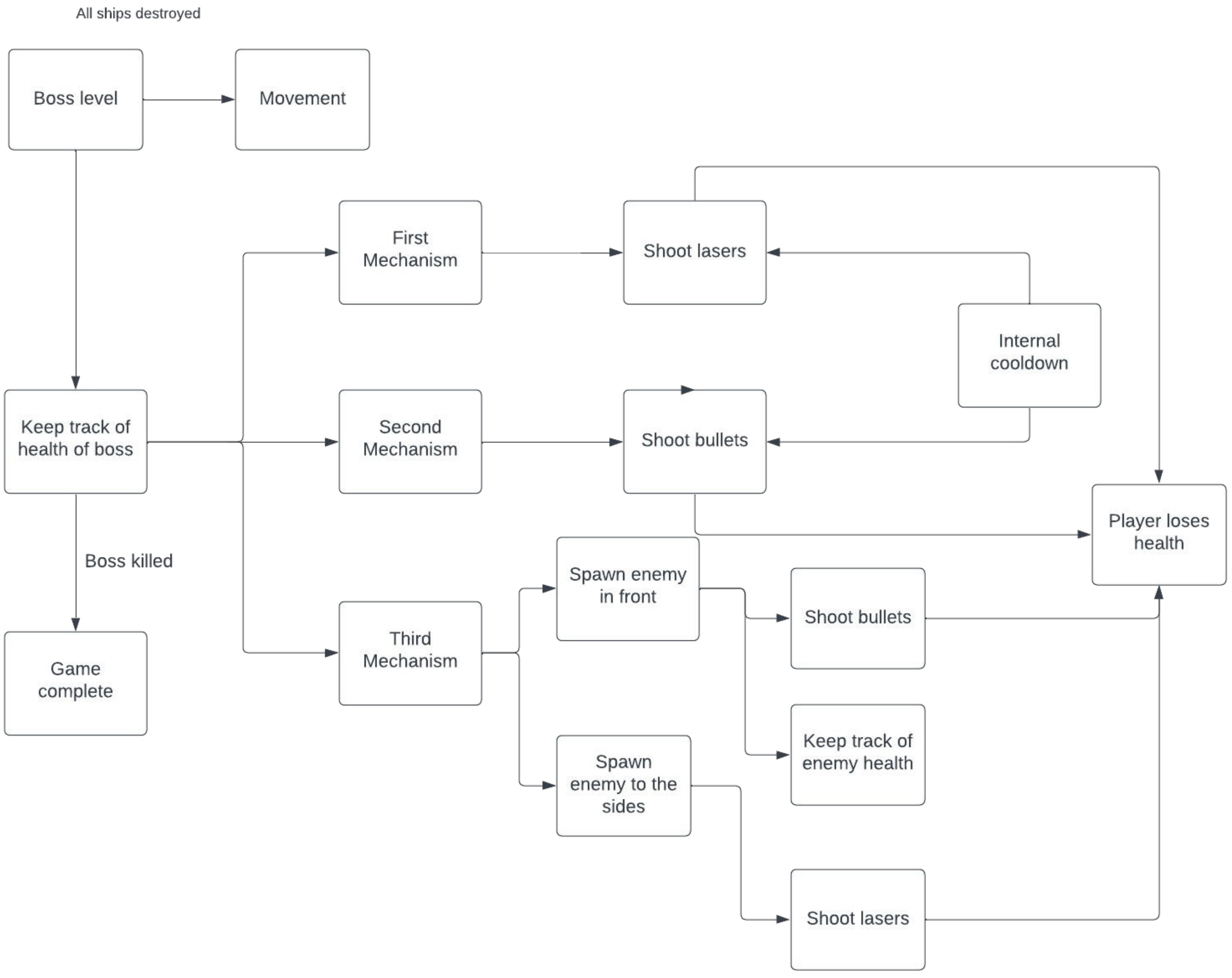


```python
def open_tutorial(window):
    if window:
        window.destroy()
    root=Tk()
    root.title("Tutorials")
    root.geometry("1360x720")

    bg_image = PhotoImage(file="tutorialimg.png")
    bg = Label(root, image=bg_image)
    bg.place(x=0, y=0)

    exit=Button(root,text="Back to menu",bg="purple",fg="white",padx=50,pady=20,command=lambda: main_menu(root))
    exit.place(x=70,y=600)
    root.mainloop()

def start_game(window):
    window.destroy()
    if main.running:
        main.run()
    if not main.running:
        main.running = True
        main_menu(0)

def main_menu(window):
    if window:
        window.destroy()

    root = Tk()
    root.title("SPACE INVADERS")
    root.geometry("1360x720")

    bg_img=PhotoImage(file="title.png")
    bg=Label(root,image=bg_img)
    bg.place(x=0,y=0)

    new_game=Button(root,text="PLAY",bg="purple",fg="white",padx=190,pady=50,command=lambda: start_game(root))    #play button
    new_game.place(x=455,y=250)

    tutorial=Button(root,text="CONTROLS",bg="purple",fg="white",padx=175,pady=50,command=lambda: open_tutorial(root))   #tutorial button
    tutorial.place(x=455,y=400)

    exit=Button(root,text="EXIT GAME",bg="purple",fg="white",padx=175,pady=50,command=root.destroy)   #exit button
    exit.place(x=455,y=550)

    root.mainloop()


main_menu(0)
```

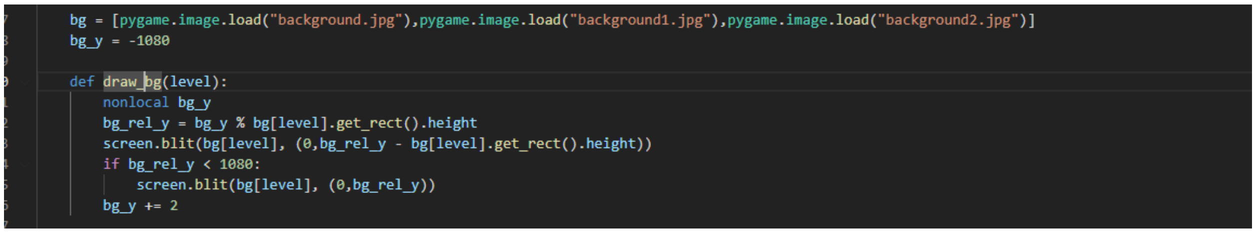
Game -


Proceed to next level
Keep track of current level
Create player hitbox
Create bullet hitbox
True
Check if level is completed
Levels
Hitbox
Player Object
Movement
Health
Shoot
Take input from user
Start with 3 hearts
Detect enemy attack collision
Check if bullet is ready
Take user input
True
True
Draw the player on screen
Keep player inside boundary
Keep track of current health
Reduce 1 heart
Shoot bullet
Player dies and game ends if number of hearts is 0
Reset bullet if it hits an enemy or goes out of the screen


Level System
Level 1
Level 2
Level 3 (boss level)
Move downwards
Player loses if enemies reach player
Draw enemies
Display level mission
Enemy movement
Detect if player hit enemy
Enemy ship is destroyed
Keep track of each enemy ship
Shoot bullets at player
If collision
Player loses health
Keep track of health of each enemy
Level complete
All ships destroyed
All enemies killed

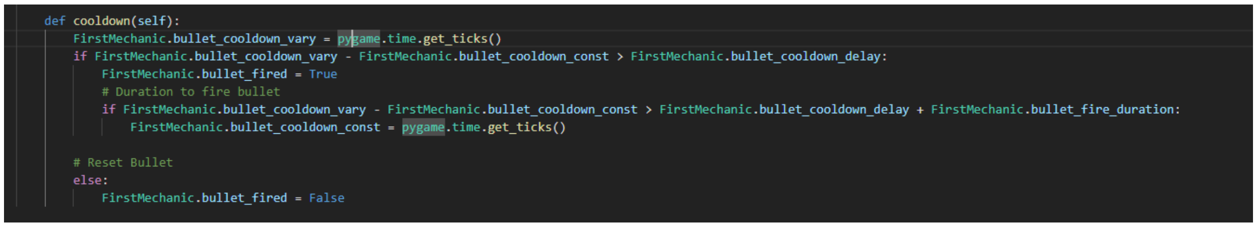

All ships destroyed
Boss level
Movement
First Mechanism
Shoot lasers
Internal cooldown
Keep track of health of boss
Second Mechanism
Shoot bullets
Player loses health
Boss killed
Spawn enemy in front
Shoot bullets
Third Mechanism
Game complete
Keep track of enemy health
Spawn enemy to the sides
Shoot lasers

Scrolling background which changes with level -



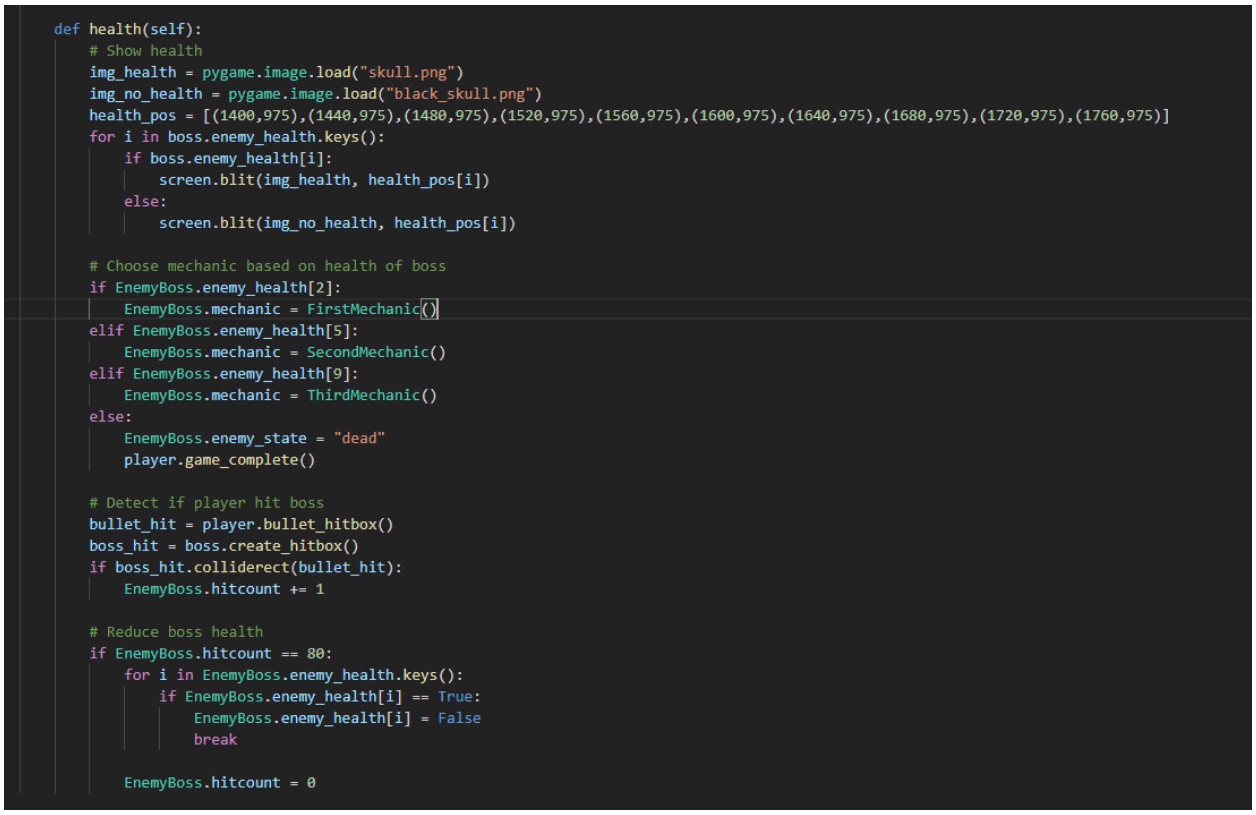
```
bg = [pygame.image.load("background.jpg"),pygame.image.load("background1.jpg"),pygame.image.load("background2.jpg")]
bg_y = -1080

def draw_bg(level):
    nonlocal bg_y
    bg_rel_y = bg_y % bg[level].get_rect().height
    screen.blit(bg[level], (0,bg_rel_y - bg[level].get_rect().height))
    if bg_rel_y < 1080:
        screen.blit(bg[level], (0,bg_rel_y))
    bg_y += 2
```

Internal cooldown system template -



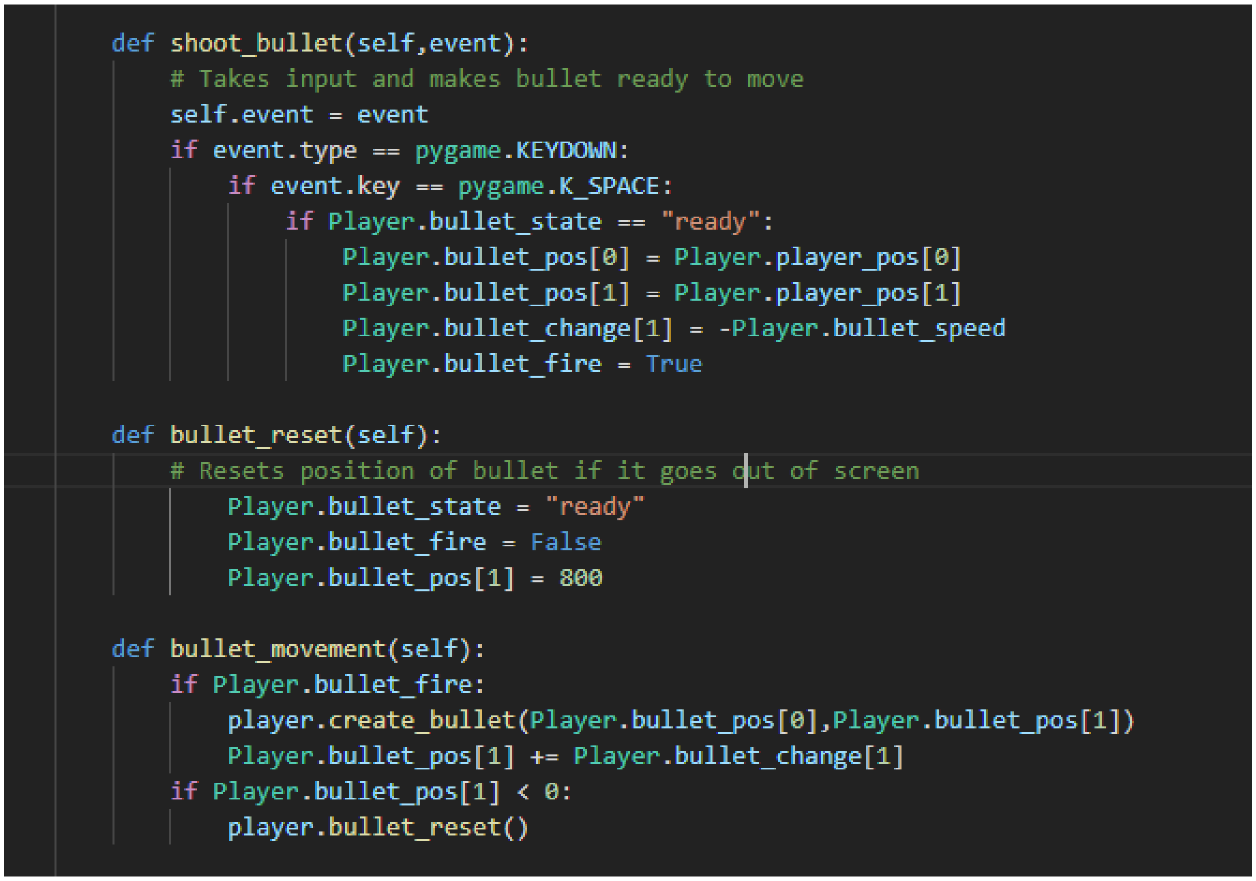
```
def cooldown(self):
    FirstMechanic.bullet_cooldown_vary = pygame.time.get_ticks()
    if FirstMechanic.bullet_cooldown_vary - FirstMechanic.bullet_cooldown_const > FirstMechanic.bullet_cooldown_delay:
        FirstMechanic.bullet_fired = True
        # Duration to fire bullet
        if FirstMechanic.bullet_cooldown_vary - FirstMechanic.bullet_cooldown_const > FirstMechanic.bullet_cooldown_delay + FirstMechanic.bullet_fire_duration:
            FirstMechanic.bullet_cooldown_const = pygame.time.get_ticks()

    # Reset Bullet
    else:
        FirstMechanic.bullet_fired = False
```

Health system template -



```
def health(self):
    # Show health
    img_health = pygame.image.load("skull.png")
    img_no_health = pygame.image.load("black_skull.png")
    health_pos = [(1400,975),(1440,975),(1480,975),(1520,975),(1560,975),(1600,975),(1640,975),(1680,975),(1720,975),(1760,975)]
    for i in boss.enemy_health.keys():
        if boss.enemy_health[i]:
            screen.blit(img_health, health_pos[i])
        else:
            screen.blit(img_no_health, health_pos[i])

    # Choose mechanic based on health of boss
    if EnemyBoss.enemy_health[2]:
        EnemyBoss.mechanic = FirstMechanic()
    elif EnemyBoss.enemy_health[5]:
        EnemyBoss.mechanic = SecondMechanic()
    elif EnemyBoss.enemy_health[9]:
        EnemyBoss.mechanic = ThirdMechanic()
    else:
        EnemyBoss.enemy_state = "dead"
        player.game_complete()

    # Detect if player hit boss
    bullet_hit = player.bullet_hitbox()
    boss_hit = boss.create_hitbox()
    if boss_hit.colliderect(bullet_hit):
        EnemyBoss.hitcount += 1

    # Reduce boss health
    if EnemyBoss.hitcount == 80:
        for i in EnemyBoss.enemy_health.keys():
            if EnemyBoss.enemy_health[i] == True:
                EnemyBoss.enemy_health[i] = False
                break

        EnemyBoss.hitcount = 0
```

TESTING

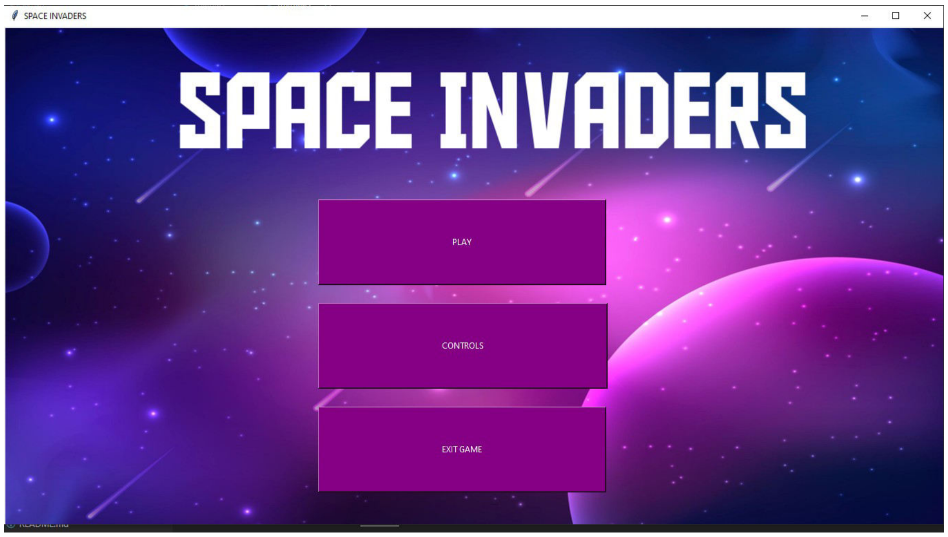


```
def shoot_bullet(self,event):
    # Takes input and makes bullet ready to move
    self.event = event
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_SPACE:
            if Player.bullet_state == "ready":
                Player.bullet_pos[0] = Player.player_pos[0]
                Player.bullet_pos[1] = Player.player_pos[1]
                Player.bullet_change[1] = -Player.bullet_speed
                Player.bullet_fire = True

def bullet_reset(self):
    # Resets position of bullet if it goes out of screen
        Player.bullet_state = "ready"
        Player.bullet_fire = False
        Player.bullet_pos[1] = 800

def bullet_movement(self):
    if Player.bullet_fire:
        player.create_bullet(Player.bullet_pos[0],Player.bullet_pos[1])
        Player.bullet_pos[1] += Player.bullet_change[1]
    if Player.bullet_pos[1] < 0:
        player.bullet_reset()
```

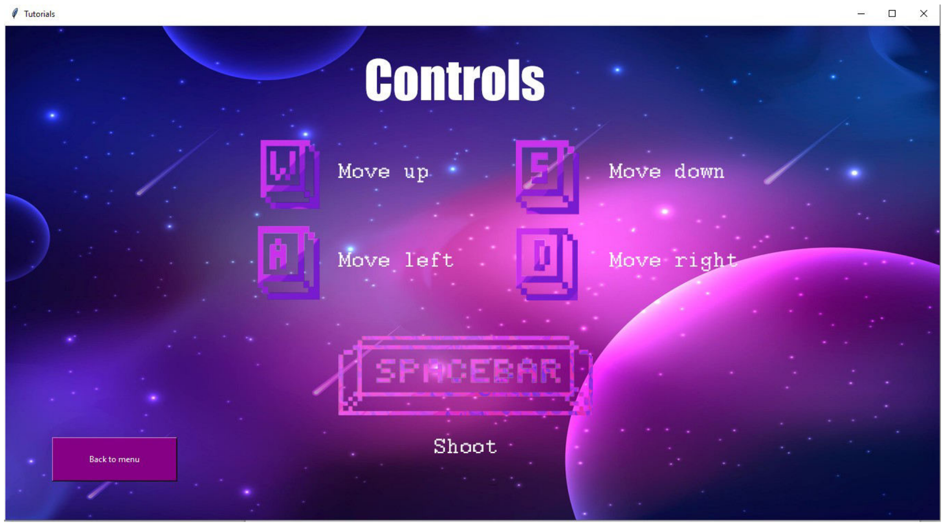
In the player shooting system, we need two variables – “bullet_state” to check if the bullet is ready or not, and “bullet_fire” to check if the bullet is currently moving across the screen. Thanks to the bullet_fire variable, we can prevent the player from spamming the shoot button, which would have kept resetting the bullet position. If it was so, the bullet would never be able to travel to the enemy.

RESULT AND ANALYSIS

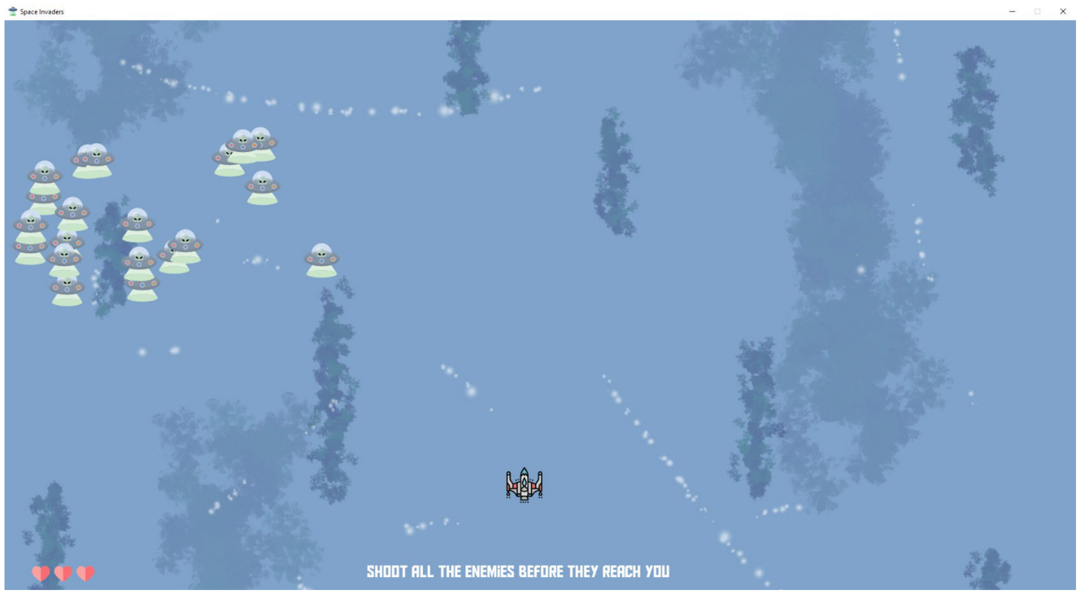
When you run the program, you are greeted by the main menu



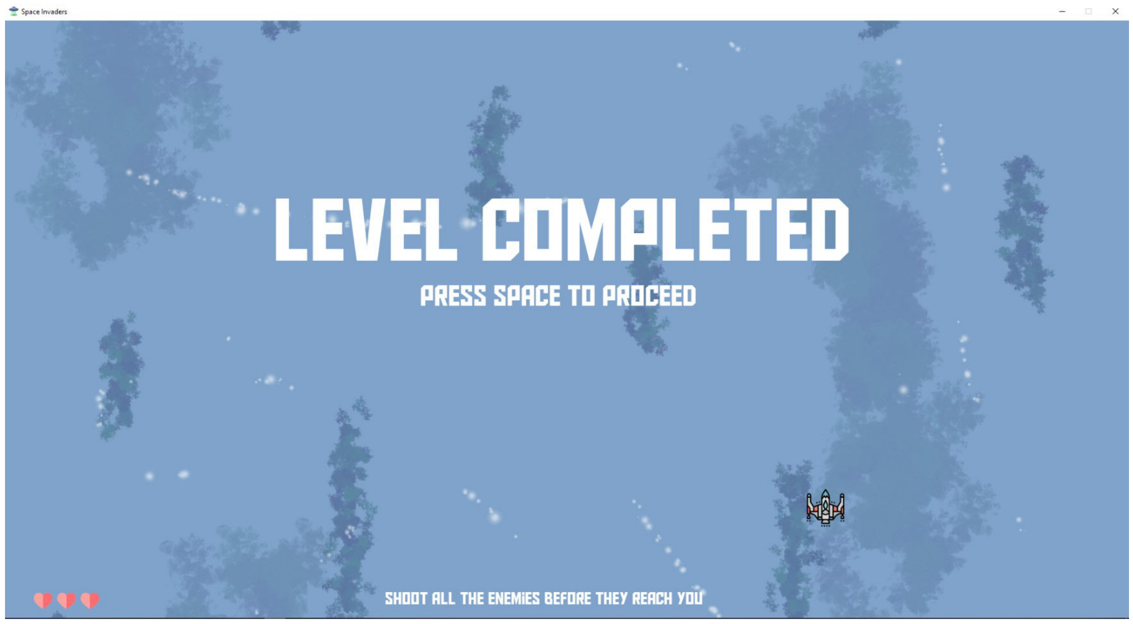
Clicking on 'CONTROLS' takes you to the controls page.



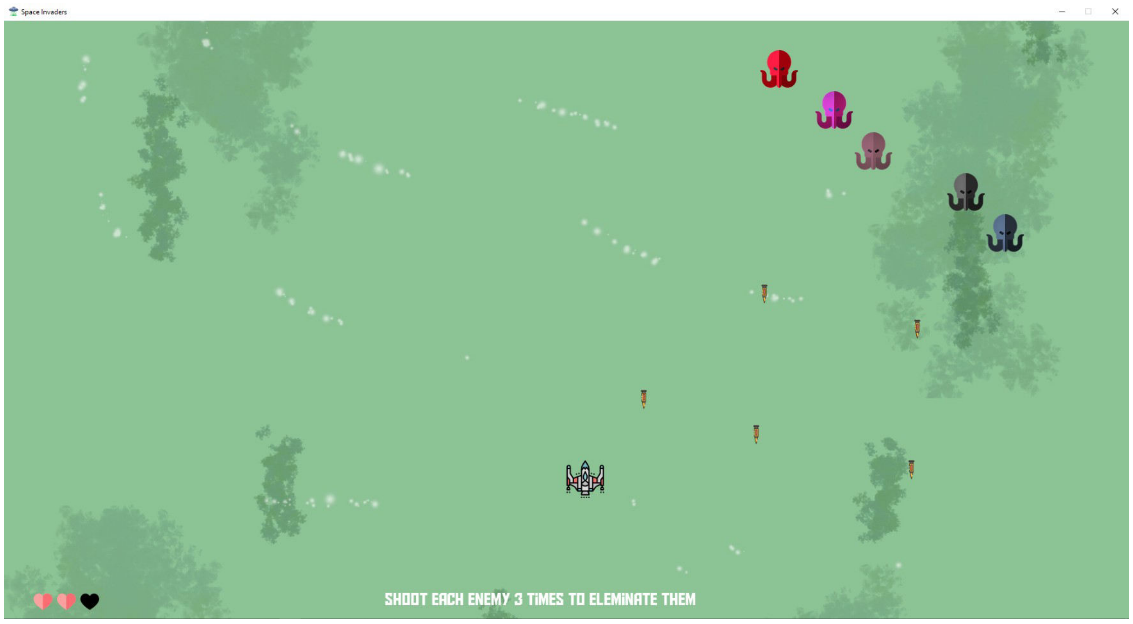
The game starts off with a simple level – destroy all enemy spaceships before they reach the bottom of the screen.

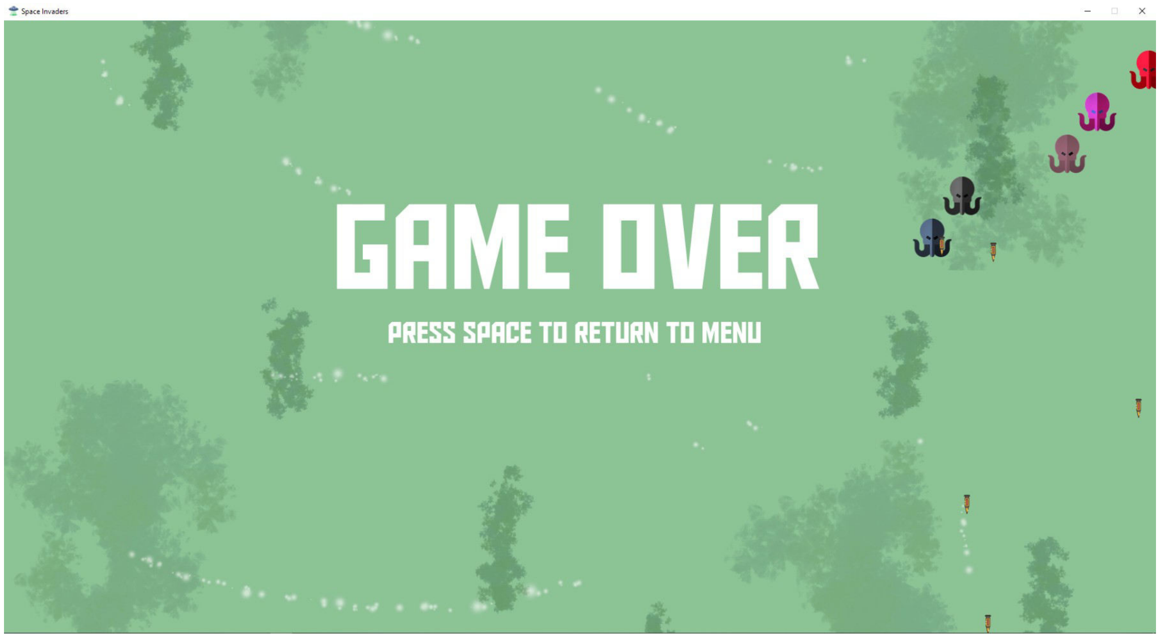


Once a level is completed, you can press the "SPACE" key to continue to the next level

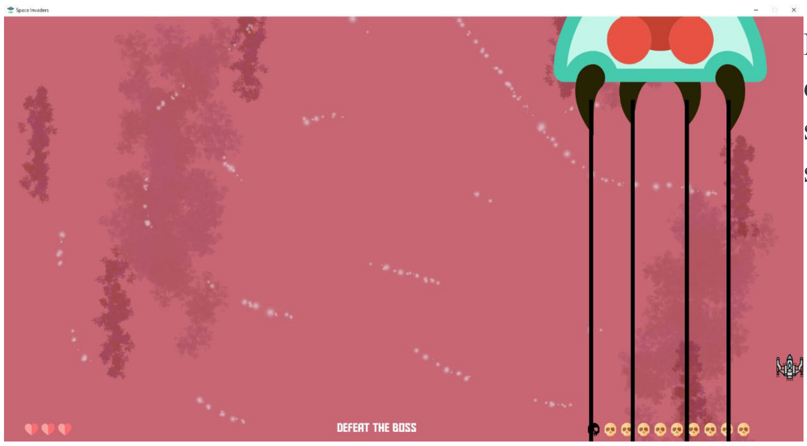


From level two onwards, even the enemy aliens can shoot you, so be careful! Be on the lookout for your health in the bottom left corner.



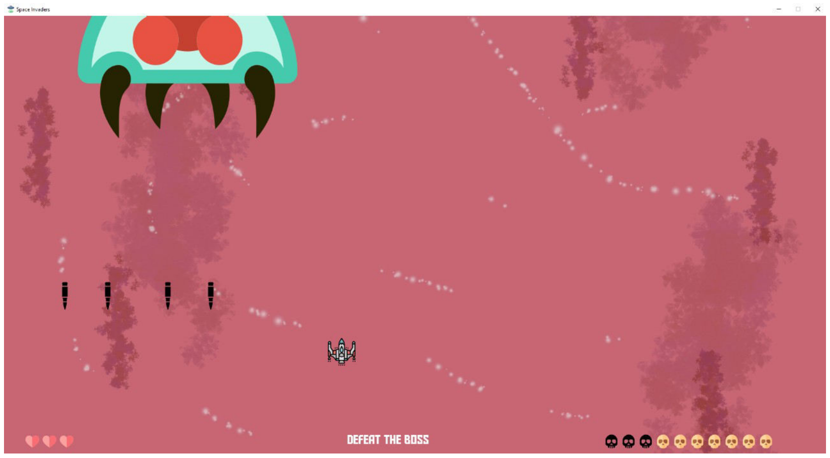


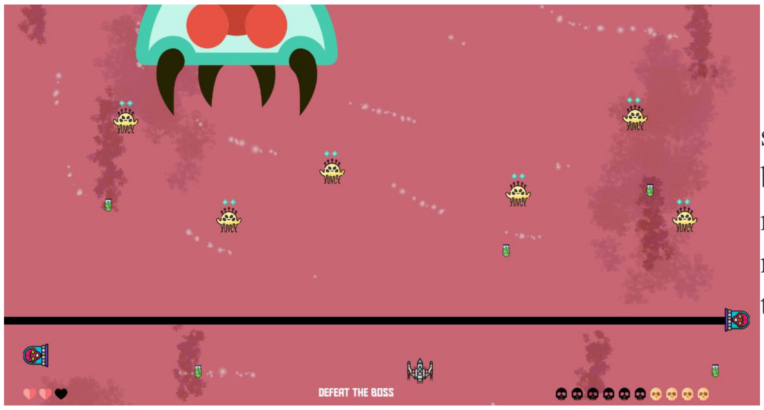
Finally, the boss level! The enemy boss is a big alien with a huge health pool. As the alien's health decreases, it changes its attack pattern, each pattern being more difficult than the previous.



First mechanism – the alien starts off by shooting four lasers. If you stand in them for too long, you will start taking damage

Second mechanism – the alien now changes to shooting bullets, each one of them being very deadly





Final mechanism – you have successfully paralyzed the alien boss, but now it has called its minions to defeat you. Finish off the minions and the boss to complete the game.

CONCLUSIONS AND FUTURE ENHANCEMENTS

These are a few additions which would enhance the game experience:-

- use inheritance to reduce redundancy of code
- a score system which is saved locally
- a leaderboard to show top scores
- model animations for different actions
- music and sounds

REFERENCES

Basic pygame tutorial - https://www.youtube.com/watch?v=FfWpgLFMI7w&t=1s

Scrolling background - https://www.youtube.com/watch?v=US3HSusUBeI&t=508s

Models - https://www.flaticon.com/
(respective artists have been credited in the "icon credits" file)