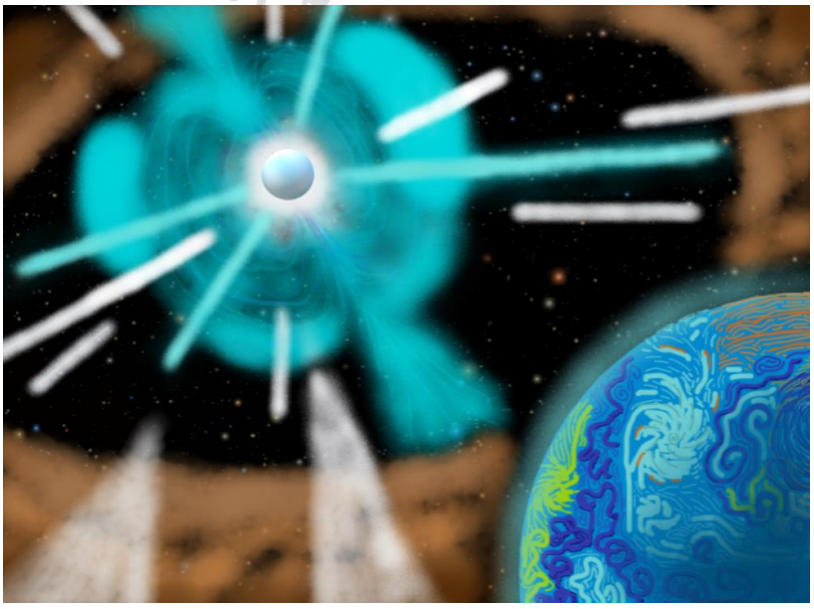
SPACE SHOOTING GAME

PROJECT REPORT

Computer Science



NAME: AARYAN SHARMA
BOARD ROLL NO:
DPS INDIRAPURAM
2020-21

CERTIFICATE

This is to certify that Aaryan Sharma of class XII-D have prepared the project on the topic of "SPACE SHOOTING GAME". The project is the result of their and their team's efforts and endeavours. This project is found worthy of acceptance as the final project report for the subject computer science of class XII. They have prepared this project under my guidance.

Ms. Rinkoo Gupta

(Computer Science Teacher)

(DPS Indirapuram)

ACKNOWLEDGEMENT

I would like to express a deep sense of gratitude towards my computer science teacher Ms. Rinkoo Gupta ma'am for guiding me though the course of my project. She always evinced keen interest in my work and her constructive advice and constant motivation have been responsible for the successful completion of this project.

My sincere thanks goes to Ms. Sangeeta Hajela, our school principal for her coordination in extending every possible support possible in the success of this project.

I would like to thank all those who have helped directly or indirectly in the completion of the project.

Aaryan Sharma

XII-D

INDEX

1. INTRODUCTION TO THE PROJECT

Our CS Project “The Space Walker” is based on game development using PyCharm and Pygame, inspired by the old arcade game ‘Space Invaders’.

Upon beginning, the player interacts with the game by controlling a space ship with arrow keys and space bar.

The goal is to survive waves of enemies for as long as possible. The longer you survive, the harder the game becomes by increasing the level.

Each level has a greater number of enemies that need to be destroyed than the previous level. There’re 5 levels in total in the game, the last level being a boss battle of sorts.

The game is accompanied by a vivid and immersive display of background arts, character sprites (all self-drawn), and audio/music effects.

There’s a score system which is affected by the number of levels you clear, the number of times the player crashes or loses a life by allowing the enemy to pass the screen without being destroyed.

When then game ends, the high-score is displayed on the menu screen.

2. MySQL Tables / CSV / Binary Files used & Their Structure

Modules imported:

- Os.path – For saving data into text files.
- Random – For generating random spawn positions.
- Pygame – Running the modules to make a videogame.
- Pickle – for handling binary files.
- Mixer – For running mp3 files.

Files used:

- Text/Binary files ("score.txt" for storing high-scores.)
- PNG files (Storing images & sprites for the game.)
- Wav & mp3 files (storing music and sound effects.)

Layers, Time & Frame Rates:

- In order: (Start) Screen, Background, UI meters (Level, Lives & Crash), Enemies, Player, Win/Lost (End) Screen.
- Wav & mixer files executed by music.play() command.
- Pause function included, using time.Clock() command.

Contacts & UI (User Interference):

- Classes & Objects used for: Ships, Enemies, Player, and Lasers. Includes measuring dimensions of the images.
- Collision detected by distance function, set to Not None.
- Health bar feature, keeps tally of how many collisions are taking place with the Player.
- Enemy health = 1 hit-point, Player health = 10 hit-points.

3. SOFTWARE & HARDWARE REQUIREMENTS

HARDWARE: Intel Core i5 processor or higher.

SOFTWARE: One of the following Operating Systems:

- Microsoft Windows 8.8.1
- Microsoft Windows 7 SP 1
- Microsoft Windows Server 2012
- Microsoft Windows Server 2008 SP2 (IA-32 Only)
- Microsoft Windows Server 2008 R2 SP1
- Microsoft Windows HPC Server 2008

Compilers:

- Intel® C++ Compiler 13.1 (Intel® Parallel Studio XE 2013 SP 1) or higher.
- Microsoft Visual C++ 10.0 or higher.

4. CODING

```
import os.path
import pickle
import random

import pygame
from pygame import mixer

pygame.font.init()
pygame.init()

WIDTH, HEIGHT = 800, 600  # THE SCREEN SIZE
DIMENSIONS = pygame.display.set_mode((WIDTH, HEIGHT))

pygame.display.set_caption("The Space Walker")  # THE GAME NAME

REDEX = pygame.image.load("REDEX.png")  # ENEMIES
GENEX = pygame.image.load("GENEX.png")
BREX = pygame.image.load("BREX.png")
BOSS = pygame.image.load("Boss.png")

player = pygame.image.load("BLShip.png")  # THE PLAYER
THESPACEWALKER = pygame.image.load("SPACEWALKER.png")

REDEXASER = pygame.image.load("REDEXASER.png")  # ENEMY LASERS
```

```
GENEXASER = pygame.image.load("GENEXASER.png")
BREXASER = pygame.image.load("BREXASER.png")
BOSSXASER = pygame.image.load("BOSSXASER.png")


LAME = pygame.image.load("SPACEXASER.png")  # PLAYER LASER
NEOXASER = pygame.image.load("NEOXASER.png")


BG = pygame.transform.scale(pygame.image.load("background.png"), (WIDTH,
HEIGHT))  # GAME BACKGROUND


music = pygame.mixer.music.load('background.mp3')  # GAME MUSIC
pygame.mixer.music.play(-1)


# FUNCTIONS BEING USED IN THE GAME  # PAUSE FUNCTION


def pause():
    paused = True
    while paused:
        pause_font = pygame.font.SysFont("freesansbold.ttf", 80)
        con_font = pygame.font.SysFont("freesansbold.ttf", 80)
        for event in pygame.event.get():


            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_c:
                    paused = False
```

```
        pause_label = pause_font.render("Paused", 1, (255, 255, 255))
        DIMENSIONS.blit(pause_label, (WIDTH / 2 - pause_label.get_width() / 2,
250))
        con_label = con_font.render("C to Continue", 1, (255, 255, 255))
        DIMENSIONS.blit(con_label, (WIDTH / 2 - con_label.get_width() / 2, 300))
        pygame.mixer.pause()
        pygame.display.update()
        pygame.time.Clock()


class Laser:
    def __init__(self, x, y, img):
        self.x = x
        self.y = y
        self.img = img
        self.mask = pygame.mask.from_surface(self.img)


    def draw(self, window):
        window.blit(self.img, (self.x, self.y))


    def move(self, vel):
        self.y += vel


    def off_screen(self, height):
        return not (height >= self.y >= 0)


    def collision(self, obj):
```

```
        return collide(self, obj)


class Ship:
    COOLDOWN = 30
    def __init__(self, x, y, health=100):
        self.x = x
        self.y = y
        self.health = health
        self.ship_img = None
        self.laser_img = None
        self.lasers = []
        self.cool_down_counter = 0

    def draw(self, window):
        window.blit(self.ship_img, (self.x, self.y))
        for laser in self.lasers:
            laser.draw(window)

    def move_lasers(self, vel, obj):
        self.cooldown()
        for laser in self.lasers:
            laser.move(vel)
            if laser.off_screen(HEIGHT):
                self.lasers.remove(laser)
            elif laser.collision(obj):
                obj.health -= 10
```

```
            self.lasers.remove(laser)

    def cooldown(self):
        if self.cool_down_counter >= self.COOLDOWN:
            self.cool_down_counter = 0
        elif self.cool_down_counter > 0:
            self.cool_down_counter += 1

    def shoot(self):
        if self.cool_down_counter == 0:
            laser = Laser(self.x - 17, self.y, self.laser_img)
            self.lasers.append(laser)
            self.cool_down_counter = 1
        lsound = mixer.Sound('Lfire.wav')
        lsound.play()

    def get_width(self):
        return self.ship_img.get_width()

    def get_height(self):
        return self.ship_img.get_height()

class Player(Ship):
    def __init__(self, x, y, health=100):
        super().__init__(x, y, health)
        self.ship_img = player
```

```
        self.laser_img = LAME
        self.mask = pygame.mask.from_surface(self.ship_img)
        self.max_health = health

    def move_lasers(self, vel, objs):
        self.cooldown()
        for laser in self.lasers:
            laser.move(vel)
            if laser.off_screen(HEIGHT):
                self.lasers.remove(laser)
            else:
                for obj in objs:
                    if laser.collision(obj):
                        colli = mixer.Sound('coll.wav')
                        colli.play()
                        obj.health -= 10
                        if laser in self.lasers:
                            self.lasers.remove(laser)

    def draw(self, window):
        super().draw(window)
        self.healthbar(window)

    def healthbar(self, window):
        pygame.draw.rect(window, (255, 0, 0),
```

```
                (self.x, self.y + self.ship_img.get_height() + 10,
self.ship_img.get_width(), 10))
        pygame.draw.rect(window, (0, 255, 0), (
            self.x, self.y + self.ship_img.get_height() + 10,
            self.ship_img.get_width() * (self.health / self.max_health),
            10))


class Enemy(Ship):
    COLOR_MAP = {
        "red": (REDEX, REDEXASER),
        "green": (GENEX, GENEXASER),
        "blue": (BREX, BREXASER),
        "black": (BOSS, BOSSXASER)}

    def __init__(self, x, y, color, health=10):
        super().__init__(x, y, health)
        self.ship_img, self.laser_img = self.COLOR_MAP[color]
        self.mask = pygame.mask.from_surface(self.ship_img)

    def move(self, vel):
        self.y += vel

    def shoot(self):
        if self.cool_down_counter == 0:
            laser = Laser(self.x - 20, self.y, self.laser_img)
            self.lasers.append(laser)
```

```
            self.cool_down_counter = 1
        if self.y > 0:
            lsound = mixer.Sound('Lfire.wav')
            lsound.play()


def collide(obj1, obj2):
    offset_x = int(obj2.x - obj1.x)
    offset_y = int(obj2.y - obj1.y)
    return obj1.mask.overlap(obj2.mask, (offset_x, offset_y)) is not None


def text_objects(text, font):
    textSurface = font.render(text, True, (0, 0, 0))
    return textSurface, textSurface.get_rect()


def main():
    run = True
    FPS = 60
    level = 0
    lives = 3
    crashed = 0
    dict1 = {'score': 0}


    main_font = pygame.font.SysFont("freesansbold.ttf", 50)
    winc_font = pygame.font.SysFont("freesansbold.ttf", 70)
    lost_font = pygame.font.SysFont("freesansbold.ttf", 70)
    upgrad_font = pygame.font.SysFont("freesansbold.ttf", 30)
```

```
credit_font = pygame.font.SysFont("freesansbold.ttf", 30)

enemies = []
wave_length = 5
enemy_vel = 1

player_vel = 5
laser_vel = 3

player = Player(400 - 30, 500)
clock = pygame.time.Clock()

lost = False
lost_count = 0

winc = False
winc_count = 0

upgrad = False
upgrad_count = 0

upgraded = False
upgraded_count = 0

def redraw_window():
    DIMENSIONS.blit(BG, (0, 0))
```

```
    # draw text
    lives_label = main_font.render(f"Lives: {lives}", 1, (255, 255, 255))
    level_label = main_font.render(f"Level: {level}", 1, (255, 255, 255))
    crashed_label = main_font.render(f"Crashed: {crashed}", 1, (255, 255,
255))

    DIMENSIONS.blit(lives_label, (10, 10))
    DIMENSIONS.blit(level_label, (WIDTH - level_label.get_width() - 10, 10))
    DIMENSIONS.blit(crashed_label, (20, 550))

    for enemy in enemies:
        enemy.draw(DIMENSIONS)

    player.draw(DIMENSIONS)

    if lost:
        pygame.mixer.music.pause()
        LoseSound = mixer.Sound('LoseSound.wav')
        LoseSound.play()
        lost_label = lost_font.render("GAME OVER", 1, (255, 255, 255))
        DIMENSIONS.blit(lost_label, (WIDTH / 2 - lost_label.get_width() / 2,
150))
        crashed_label = main_font.render(f"You crashed: {crashed} times", 1,
(255, 60, 60))
        DIMENSIONS.blit(crashed_label, (WIDTH / 2 - crashed_label.get_width()
/ 2, 245))
```

```
        credit_label = main_font.render(f"Game by (Aaditya & Aaryan) Sharma", 1, (255, 128, 0))

        DIMENSIONS.blit(credit_label, (WIDTH / 2 - credit_label.get_width() / 2, 285))

        credit1_label = main_font.render(f"Thank you for playing The Space Walker!", 1, (255, 128, 0))

        DIMENSIONS.blit(credit1_label, (WIDTH / 2 - credit1_label.get_width() / 2, 320))

        score_label = main_font.render(f"Score: {int(dict1['score']) - crashed}", 1, (255, 60, 60))

        DIMENSIONS.blit(score_label, (WIDTH / 2 - score_label.get_width() / 2, 205))

        credit2_label = credit_font.render(

            "Music: 'Astra Lost in Space Theme' (from YouTube) - by Masaru Yokoyama.", 1, (255, 255, 0))

        DIMENSIONS.blit(credit2_label, (WIDTH / 2 - credit2_label.get_width() / 2, 365))

        credit3_label = credit_font.render(

            "Background: 'Stars Background' (from Wallpapertip.com) - by Helena Ranaldi.", 1, (255, 255, 0))

        DIMENSIONS.blit(credit3_label, (WIDTH / 2 - credit3_label.get_width() / 2, 390))


    if winc:

        pygame.mixer.music.pause()

        WinSound = mixer.Sound('WinSound.wav')

        WinSound.play()

        winc_label = winc_font.render("You Win!", 1, (255, 255, 255))
```

```
        DIMENSIONS.blit(winc_label, (WIDTH / 2 - winc_label.get_width() / 2,
150))

        crashed_label = main_font.render(f"You crashed: {crashed} times", 1,
(255, 60, 60))

        DIMENSIONS.blit(crashed_label, (WIDTH / 2 - crashed_label.get_width()
/ 2, 245))

        credit_label = main_font.render(f"Game by (Aaditya & Aaryan) Sharma",
1, (255, 128, 0))

        DIMENSIONS.blit(credit_label, (WIDTH / 2 - credit_label.get_width() / 2,
285))

        credit1_label = main_font.render(f"Thank you for playing The Space
Walker!", 1, (255, 128, 0))

        DIMENSIONS.blit(credit1_label, (WIDTH / 2 - credit1_label.get_width() /
2, 320))

        score_label = main_font.render(f"Score: {int(dict1['score']) - crashed}",
1, (255, 60, 60))

        DIMENSIONS.blit(score_label, (WIDTH / 2 - score_label.get_width() / 2,
205))

        credit2_label = credit_font.render(

            "Music: 'Astra Lost in Space Theme' (from YouTube) - by Masaru
Yokoyama.", 1, (255, 255, 0))

        DIMENSIONS.blit(credit2_label, (WIDTH / 2 - credit2_label.get_width() /
2, 365))

        credit3_label = credit_font.render(

            "Background: 'Stars Background' (from Wallpapertip.com) - by Helena
Ranaldi.", 1, (255, 255, 0))

        DIMENSIONS.blit(credit3_label, (WIDTH / 2 - credit3_label.get_width() /
2, 390))


    if upgrad:
```

```
        upgrad_label = upgrad_font.render("Your ship is being upgraded!", 1, (255, 255, 255))
        DIMENSIONS.blit(upgrad_label, (WIDTH / 2 - upgrad_label.get_width() / 2, 270))
        upgrad_label = upgrad_font.render("Survive this level!!", 1, (255, 255, 255))
        DIMENSIONS.blit(upgrad_label, (WIDTH / 2 - upgrad_label.get_width() / 2, 290))


    if upgraded:
        upgraded_label = winc_font.render("Final Wave: Boss!!", 1, (255, 60, 60))
        DIMENSIONS.blit(upgraded_label, (WIDTH / 2 - upgraded_label.get_width() / 2, 250))
        upgraded_label = upgrad_font.render("HEALTH RESTORED!", 1, (0, 255, 0))
        DIMENSIONS.blit(upgraded_label, (WIDTH / 2 - upgraded_label.get_width() / 2, 300))
        upgraded_label = upgrad_font.render("Upgrade completed!", 1, (255, 255, 255))
        DIMENSIONS.blit(upgraded_label, (WIDTH / 2 - upgraded_label.get_width() / 2, 330))


    pygame.display.update()


while run:
    clock.tick(FPS)
    redraw_window()
```

```
    if lives <= 0 or player.health <= 0:
        lost = True
        lost_count += 1


    if lost:
        if lost_count > FPS * 8.9:
            run = False
        else:
            continue


    if level < 6:
        if len(enemies) == 0:
            if level < 5:
                dict1['score'] += 20
            level += 1
            wave_length += 1
            for i in range(wave_length):
                if level == 1:
                    enemy = Enemy(random.randrange(25, WIDTH - 80),
random.randrange(-1200, -100),
                                  random.choice(["blue"]))
                    enemies.append(enemy)
                if level == 2:
                    enemy = Enemy(random.randrange(25, WIDTH - 80),
random.randrange(-1200, -100),
                                  random.choice(["red"]))
```

```
                enemies.append(enemy)
            if level == 3:
                enemy = Enemy(random.randrange(25, WIDTH - 80),
random.randrange(-1200, -100),
                              random.choice(["green"]))
                enemies.append(enemy)
            if level == 4:
                enemy = Enemy(random.randrange(12, WIDTH - 80),
random.randrange(-1200, -100),
                              random.choice(["red", "green", "blue"]))
                enemies.append(enemy)
        wave_length = 15
        for i in range(wave_length):
            if level == 5:
                enemy_vel = 1.3
                player.health = 100
                player_vel = 3
                player.ship_img = THESPACEWALKER
                player.laser_img = NEOXASER
                enemy = Enemy(random.randrange(0, WIDTH - 172),
random.randrange(-1200, -100),
                              random.choice(["black"]))
                enemies.append(enemy)

    if level > 5:
        winc = True
        winc_count += 1
```

```
if level == 4:
    upgrad = True
    upgrad_count += 1

if winc:
    if winc_count > FPS * 9:
        run = False
    else:
        continue

if upgrad:
    if upgrad_count > FPS * 3:
        upgrad = False
    else:
        continue

if level == 5:
    upgraded = True
    upgraded_count += 1

if upgraded:
    if upgraded_count > FPS * 2:
        upgraded = False
    else:
        continue
```

```
    if lost or winc:
        f = open("score.dat", "ab")
        score = dict1['score']
        crashed = crashed
        d = f'{score - crashed}\n'
        pickle.dump(d, f)
        f.close()

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

    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT] and player.x - player_vel > 0:  # left
        player.x -= player_vel
    if keys[pygame.K_RIGHT] and player.x + player_vel + player.get_width() <
WIDTH:  # right
        player.x += player_vel
    if keys[pygame.K_SPACE]:
        player.shoot()
    if keys[pygame.K_p]:
        pause()

    for enemy in enemies[:]:
        enemy.move(enemy_vel)
```

```
            enemy.move_lasers(laser_vel, player)

            if enemy.health == 0:
                enemies.remove(enemy)

            if random.randrange(0, 2 * 60) == 1:
                enemy.shoot()

            if collide(enemy, player):
                player.health -= 10
                enemies.remove(enemy)
                crashed += 1

            elif enemy.y + enemy.get_height() > HEIGHT:
                lives -= 1
                enemies.remove(enemy)

        player.move_lasers(-laser_vel, enemies)

def button1():
    mouse = pygame.mouse.get_pos()
    if WIDTH / 2 - 75 + 150 > mouse[0] > WIDTH / 2 - 75 and 265 + 50 > mouse[1]
> 265:
        pygame.draw.rect(DIMENSIONS, (255, 128, 0), (WIDTH / 2 - 75, 265, 150,
50))
    else:
```

```
    pygame.draw.rect(DIMENSIONS, (255, 255, 255), (WIDTH / 2 - 75, 265,
150, 50))


  smallText = pygame.font.Font("freesansbold.ttf", 20)
  textSurf, textRect = text_objects("BLAST OFF!", smallText)
  textRect.center = ((WIDTH / 2), (265 + (50 / 2)))
  DIMENSIONS.blit(textSurf, textRect)


  for event in pygame.event.get():
    if WIDTH / 2 - 75 + 150 > mouse[0] > WIDTH / 2 - 75 and 265 + 50 >
mouse[1] > 265:
      if event.type == pygame.MOUSEBUTTONDOWN:
        main()


def Score1():
  if os.path.isfile("score.dat"):
    d = open("score.dat", "rb")  # d.closed == True, the file is open now.
    n = 0
    while True:
      try:
        rec = pickle.load(d)
        print("Checking for start-up: " + rec)
        n += 1
      except EOFError:
        d.close()
        break
```

```
    if n == 0:
        pygame.draw.rect(DIMENSIONS, (255, 128, 0), (WIDTH / 2 - 100, 165, 200, 60))
        pygame.draw.rect(DIMENSIONS, (255, 255, 255), (WIDTH / 2 - 95, 170, 190, 50))
        smallText = pygame.font.Font("freesansbold.ttf", 20)
        textSurf, textRect = text_objects(f"HIGH SCORE: 00", smallText)
        textRect.center = ((WIDTH / 2), (170 + (50 / 2)))
        DIMENSIONS.blit(textSurf, textRect)
        print("EMPTY?! Who resetted the scores?!")  # File is empty. Somebody erased the scores for a fresh start.

    else:
        f = open("score.dat", "rb")
        n = 0
        scr = []
        while True:
            try:
                rec = pickle.load(f)
                print("Reading from Bfile (rec): " + rec)
                scr.append(rec)
                n += 1
            except EOFError:
                f.close()
                break
```

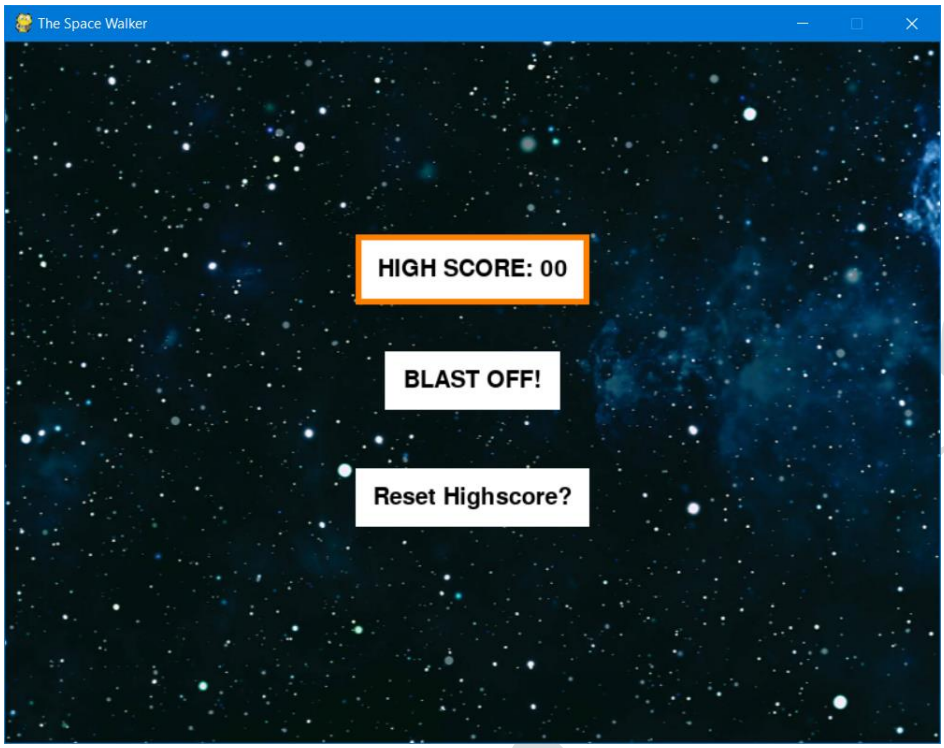
```
            if n >= 2:
                highscore = max(scr)
                pygame.draw.rect(DIMENSIONS, (255, 128, 0), (WIDTH / 2 - 100,
165, 200, 60))
                pygame.draw.rect(DIMENSIONS, (255, 255, 255), (WIDTH / 2 - 95,
170, 190, 50))
                smallText = pygame.font.Font("freesansbold.ttf", 20)
                textSurf, textRect = text_objects(f"HIGH SCORE: {highscore}",
smallText)
                textRect.center = ((WIDTH / 2), (170 + (50 / 2)))
                DIMENSIONS.blit(textSurf, textRect)
                print("True highscore")  # This is how it should be.

            elif n == 1:
                highscore = scr[0]
                pygame.draw.rect(DIMENSIONS, (255, 128, 0), (WIDTH / 2 - 100,
165, 200, 60))
                pygame.draw.rect(DIMENSIONS, (255, 255, 255), (WIDTH / 2 - 95,
170, 190, 50))
                smallText = pygame.font.Font("freesansbold.ttf", 20)
                textSurf, textRect = text_objects(f"HIGH SCORE: {highscore}",
smallText)
                textRect.center = ((WIDTH / 2), (170 + (50 / 2)))
                DIMENSIONS.blit(textSurf, textRect)
                print("First play")  # Played for the 1st time.

        print("Reading of scr: ")
```

```
        print(scr)  # This tells me what the program reads from score.dat i.e. what all scores are saved now.


    else:
        pygame.draw.rect(DIMENSIONS, (255, 128, 0), (WIDTH / 2 - 100, 165, 200, 60))
        pygame.draw.rect(DIMENSIONS, (255, 255, 255), (WIDTH / 2 - 95, 170, 190, 50))
        smallText = pygame.font.Font("freesansbold.ttf", 20)
        textSurf, textRect = text_objects(f"HIGH SCORE: 00", smallText)
        textRect.center = ((WIDTH / 2), (170 + (50 / 2)))
        DIMENSIONS.blit(textSurf, textRect)
        print("Playing for the 1st time? Good luck!")  # You've either resetted the scores or nobody has played yet.


def REScore1():
    mouse = pygame.mouse.get_pos()
    if WIDTH / 2 - 100 + 200 > mouse[0] > WIDTH / 2 - 100 and 365 + 50 > mouse[1] > 365:
        pygame.draw.rect(DIMENSIONS, (100, 100, 100), (WIDTH / 2 - 100, 365, 200, 50))
    else:
        pygame.draw.rect(DIMENSIONS, (255, 255, 255), (WIDTH / 2 - 100, 365, 200, 50))


    smallText = pygame.font.Font("freesansbold.ttf", 20)
    textSurf, textRect = text_objects("Reset Highscore?", smallText)
    textRect.center = ((WIDTH / 2), (365 + (50 / 2)))
```

```
    DIMENSIONS.blit(textSurf, textRect)


    for event in pygame.event.get():
        if WIDTH / 2 - 100 + 200 > mouse[0] > WIDTH / 2 - 100 and 365 + 50 >
mouse[1] > 365:
            if event.type == pygame.MOUSEBUTTONDOWN:
                f = open("score.dat", "rb+")
                f.truncate()
                f.close()


def start_screen():
    run = True
    while run:
        DIMENSIONS.blit(BG, (0, 0))


        button1()
        Score1()
        REScore1()


        pygame.display.update()


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

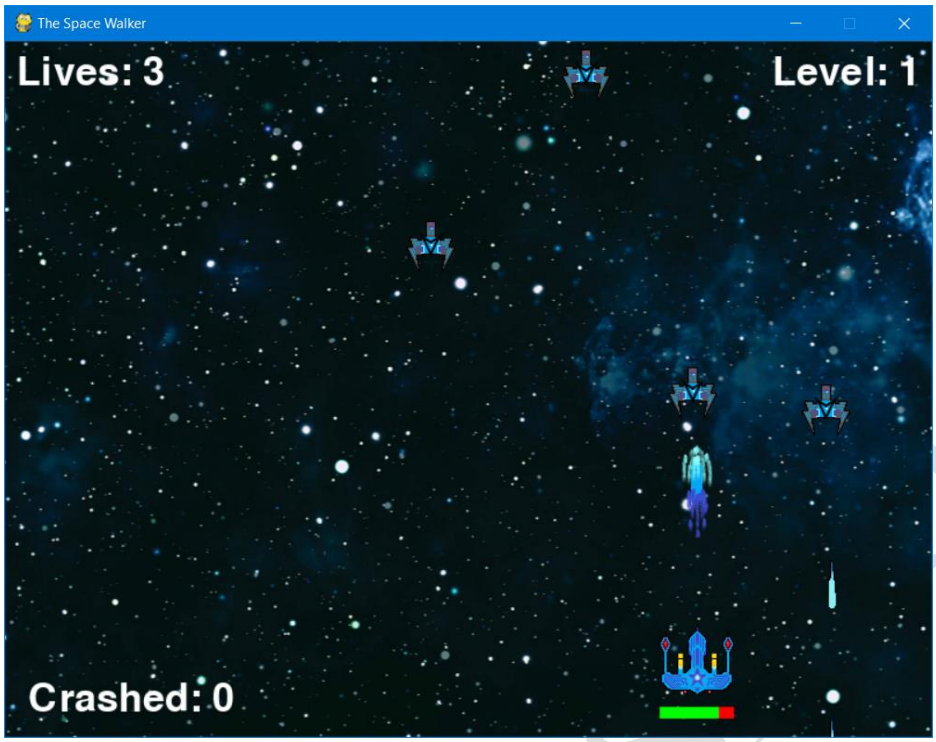
5. OUTPUTS

START-UP (first time):

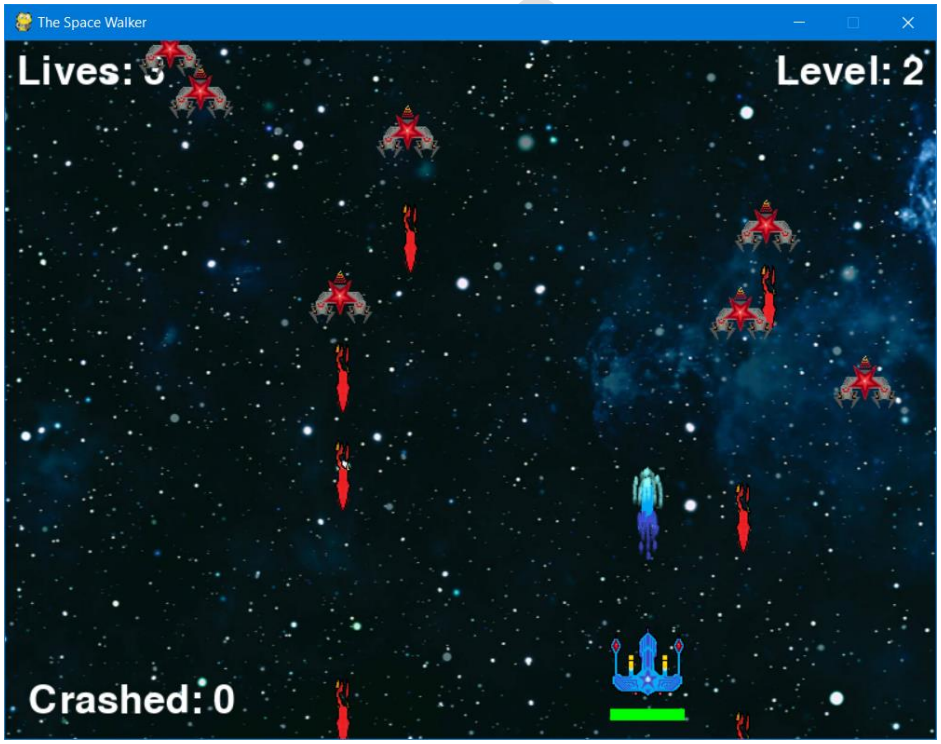


"Blast off!" first screen:

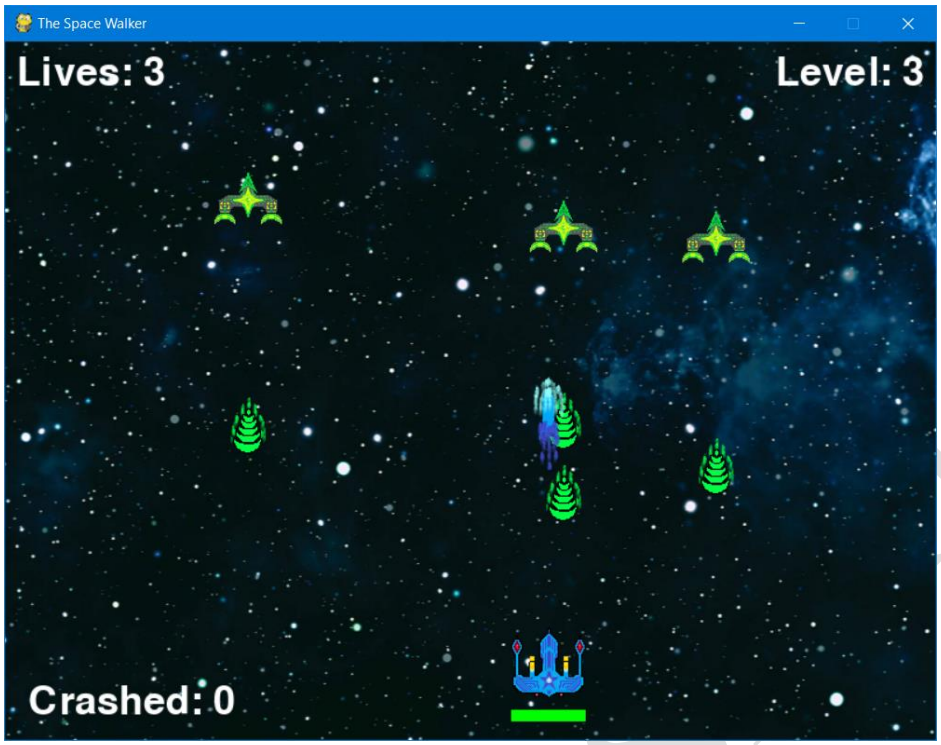
Level 1:



Level 2:



Level 3:



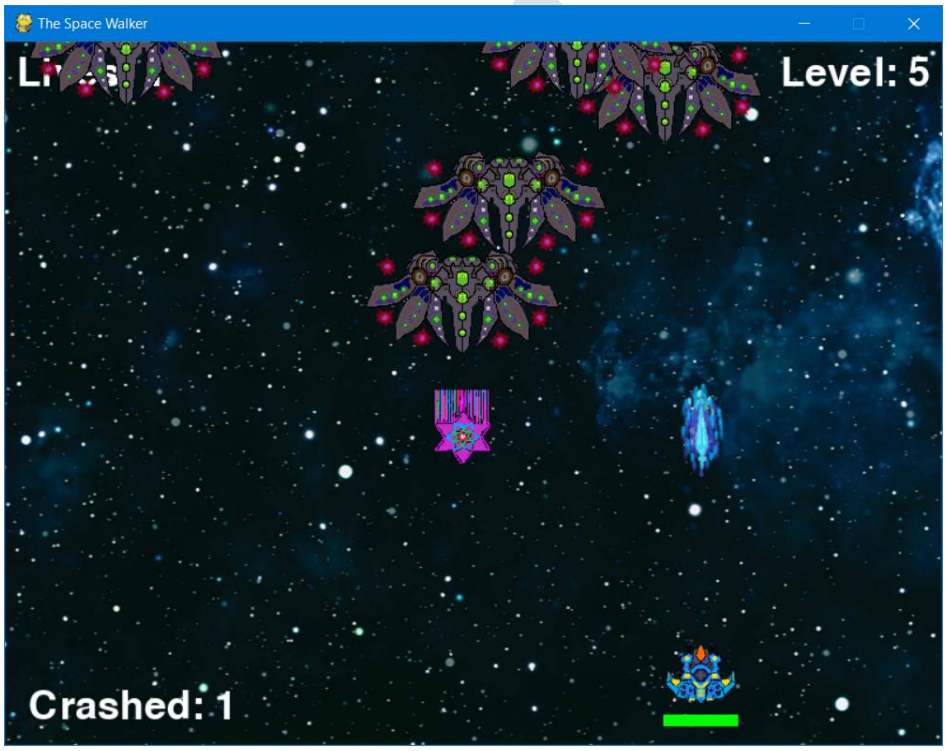
Mid-Notification (1):

Pause Screen:

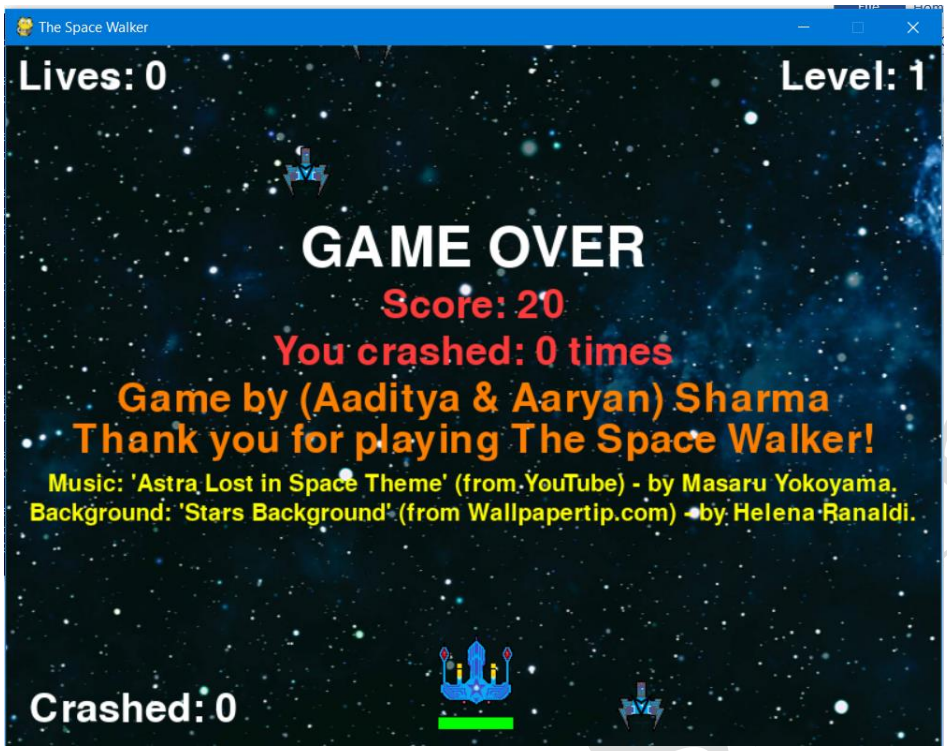
Level 4:

Mid-Notification (2):

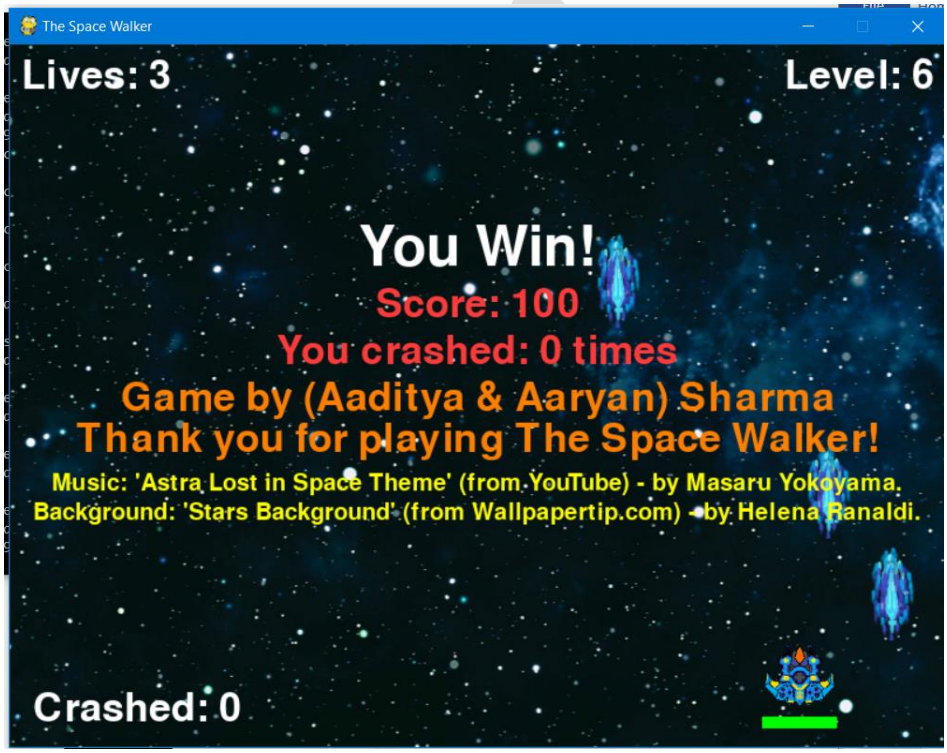
Level 5:



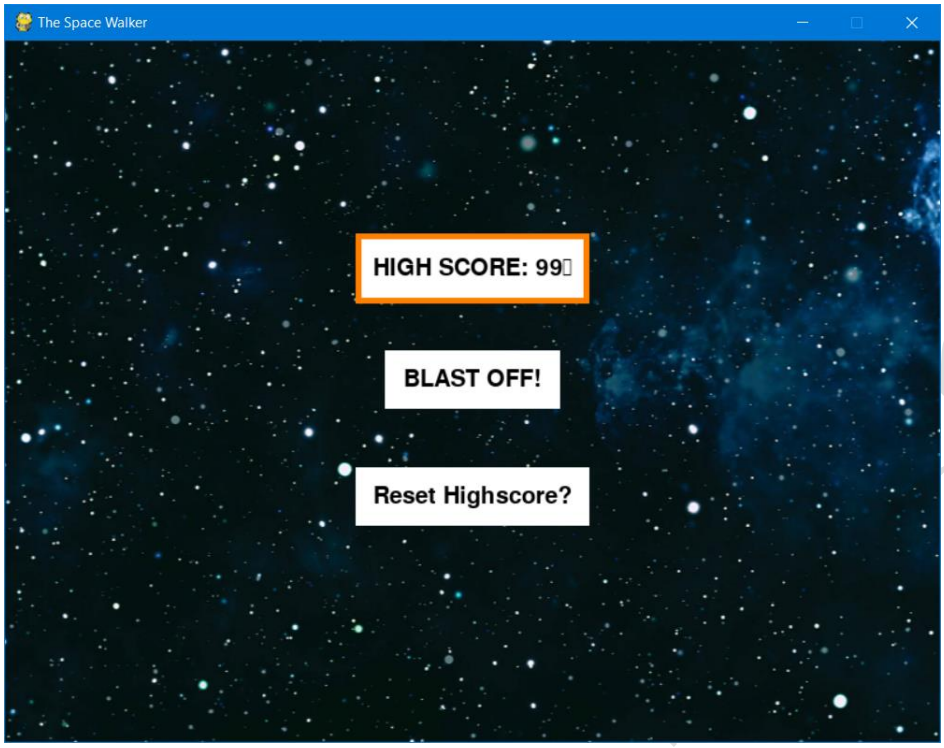
End (Lose) Screen:



End (Win) Screen:



Menu Screen (Updated):



6. CONCLUSION

A game called 'The Space Walker' based off of the arcade game 'Space Invaders', developed using PyCharm and Pygame, and Python.

Project Synopsis:

TOPIC: Game Development

- Python Library: Pygame and PyCharm
- This project will be used for development of a game.
- Data will be stored regarding number of enemies, obstacles etc.

Features:

- Decent frame rate
- Aesthetically pleasing backgrounds
- Sound effects
- Collision detection
- Drawings on screen

7. FUTURE ENHANCEMENTS

- The high score system has a limit, a maximum possible score per says, which ruins the purpose of a high score if any player achieves it.
- Bug-fixes: High score sometimes doesn't display actual high score after the 2nd game, unless it's on PyCharm.
- A player-account system which records the username and high-score to display the scores of players in a tabular format, in descending order, giving ranks accordingly.
- Better background and sprite artwork and soundtracks.
- A secret level that can be unlocked under special conditions.
- A story line/plot to not just have a casual game. To potentially make players more attracted to the game.
- Better UI.

8. BIBLIOGRAPHY

Original help for saving highscore in python:

https://stackoverflow.com/questions/16726354/saving-the-highscore-for-a-python-game.

File Handling help:

https://stackoverflow.com/questions/82831/how-do-i-check-whether-a-file-exists-without-exceptions.

Class PPTs referred to (by Ms. Rinkoo Gupta):

"File Handling"
"Numbers & Strings"

Inspirations & other online references:

Pygame Tutorial - Creating Space Invaders.
Game Development in Python 3 With Pygame - 1 - Intro.
Python / Pygame Tutorial: Creating multiple stages in a game.
Game Development in Python 3 With Pygame - 14 - Button Function.

Tools used along the way:

MS Paint & Paint 3D
Audacity [All sound effects are from YouTube.]