VIKRAM - THE GHOST

A PROJECT REPORT

Submitted by

S ANBUMANI **[613519104004]**

S THIYANESWARAN **[613519104045]**

K VINO **[613519104048]**

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

GOVERNMENT COLLEGE OF ENGINEERING, DHARMAPURI

ANNA UNIVERSITY: CHENNAI 600 025

JUNE 2022

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"VIKRAM THE GHOST"** is the bonafide work of the following students, **ANBUMANI.S [613519104004] , THIYANESHWARAN.S [613519104045], VINO.K [613519104048],** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Mrs. B. JOTHI, M.E.,	**Mrs. P. SUGAVANESWARI, M.E.,**
HEAD OF THE DEPARTMENT,	**SUPERVISOR,**
	ASSISTANT PROFESSOR,
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
Government College of Engineering, Dharmapuri.	Government College of Engineering, Dharmapuri.

Submitted for the university examination held at Government College of Engineering, Dharmapuri on ……..

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We are personally indebted to a number of people who gave us their useful insights to aid in our overall progress for this project. A complete acknowledgement would therefore be encyclopedic. First of all, we would like to give our deepest gratitude to our parents for permitting us to take up this course.

We feel happy to convey our kind regards and sincere thanks to our beloved **Principal Dr V. SUMATHY, M.E, Ph.D.** who provided her kind concern for carrying out this project work and providing suitable environment to work with.

We express our sincere thanks to **Dr V. RAJKUMAR, Ph.D., Vice-Principal**, for encouraging us to do project.

We wish to express our sense of gratitude and sincere thanks to our **Head of the Department Mrs. B. JOTHI, M.E.,** of Computer Science and Engineering for her valuable guidance in the preparation and presentation of this project.

We express our profound and sincere thanks to our project **Supervisor** and **Coordinator Mrs. P. SUGAVANESWARI, M.E., Assistant professor,** of Computer Science and Engineering for her enlightening thought and remarkable guidance that helped us in the successful completion of our project.

We thank all our Friends, Teaching, Non-Teaching staffs and our well-wishers for their constant support all the time.

TABLE OF CONTENTS

ABSTRACT

Now a days, people are spending time on games more than other system application. Kids like video games more than the outdoor games .They are getting addicted to a non-ending video games, more specifically the online games. So Rather than playing games in online, this game focused on developing an offline and a hard mode game which will be much difficult to play. Vikram – The Ghost is a case study game which in suitable for children as well as adult. This case study meant for implementing the game without losing its attraction. It jusorus with in games comes with a single player mode implemented in python and its graphics of the gameplay system is good and smooth to control for the users. The main objective of the this gaming project is the player have to survive and stay alive for long. The player is named as character Vikram who have to stay alive by escapes from the attack of the enemies and survive by hitting the enemies with shotting gun. When Vikram's life ran out, he will die and users have to restart the game. For hitting every enemy, the player can gain a score and total score is calculated each time. At last highest score will be displayed.

LIST OF FIGURES

LIST OF ABBREVIATIONS

IDE - INTEGRATED DEVELOPMENT ENVIRONMENT

GPL - GENERAL PUBLIC LICENSE

UML – UNIFIED MARKUP LANGUAGE

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The primary objective of the player - VIKRAM is to gain experience by killing the enemies. Throughout the entire arena, AI creates enemies automatically. He can score through shooting the enemies. The highest score will be stored in the local database and displayed. The enemies arrival time will get shorten by every 11^{th} kill. Player can gain extra immunity to keep him alive. This is the thing which makes game more interesting.

1.2 OBJECTIVE

Now a days, people are spending time on games more than other system application. Kids like video games more than the outdoor games .They are getting addicted to a non-ending video games, more specifically the online games. So Rather than playing games in online, this project focused on developing an offline and a hard mode game which will be much difficult to play.

Vikram – The Ghost is a game suitable for children as well as matured peoples. The rules and regulations of this game is to survive and stay alive. The case study meant for implementing the game without losing its attraction. This games comes with a single player mode. This Vikram – The Ghost game is implemented in python and its graphics of the gameplay system is good and smooth to control for the user

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 INTRODUCTION

The requirement specifications is a technical specification of requirements for the hardware products. It is the first step in the requirement analysis process. It lists the requirements of a particular hardware system including functional, performance and safety requirements. The requirements also provide usage scenarios from a user and an operational perspective. The purpose of hardware requirements specification is to provide a detailed overview of the hardware project, its parameters and goals. This describes the project target and its user interface, hardware and software requirements.

2.2 SOFTWARE AND HARDWARE REQUIREMENT

2.2.1 HARDWARE REQUIREMENT

- Hard disk : 100 MB or Above
- RAM : 2GB or Above

2.2.2 SOFTWARE REQUIREMENT

- Operating System : Windows
- Software : Pygame
- Programming Language : Python

2.3 TECHNOLOGIES USED

2.3.1 SOFTWARE

2.3.1.1 VISUAL STUDIO

An *integrated development environment* (IDE) is a feature-rich program that supports many aspects of software development. The Visual Studio IDE is a creative

launching pad that you can use to edit, debug, and build code, and then publish an app. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to enhance the software development process.

- **SQUIGGLES AND QUICK ACTIONS :** Squiggles are wavy underlines that alert you to errors or potential problems in your code as you type. These visual clues help you fix problems immediately, without waiting to discover errors during build or runtime. If you hover over a squiggle, you see more information about the error. A lightbulb might also appear in the left margin showing *Quick Actions* you can take to fix the error.
- **INTELLISENSE :** IntelliSense is a set of features that display information about your code directly in the editor and, in some cases, write small bits of code for you. It's like having basic documentation inline in the editor, so you don't have to look up type information elsewhere.

2.3.1.2 PYGAME

Pygame is cross-platform set of Python modules designed for developing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.

- **HISTROY**

  Pygame was originally written by Pete Shinners to replace PySDL after its development stalled. It has been a community project since 2000 and is released under the free software GNU Lesser General Public License (which "provides for Pygame to be distributed with open source and commercial software").

- **FEATURES**

Pygame uses the Simple Direct Media Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.

Other features that SDL does have include vector math, collision detection, 2D sprite scene graph management, MIDI support, camera, pixel-array manipulation, transformations, filtering, advanced freetype font support, and drawing.

2.3.1.3 ADOBE ILUSTRATOR

Adobe Illustrator is a vector graphics editor and design program developed and marketed by adobe illustrator .Originally designed for the Apple Macintosh, development of Adobe Illustrator began in 1985. Along with Creative Cloud , Illustrator CC was released. The latest version, Illustrator 2022, was released on October 26, 2021, and is the 25th generation in the product line.

2.3.1.4 ADOBE PHOTOSHOP

Adobe Photoshop is a raster graphics editor developed and published by Adobe Inc. for Windows and macOS. It was originally created in 1988 by Thomas and John Knoll. Since then, the software has become the industry standard not only in raster graphics editing, but in digital art as a whole.

2.3.1.5 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985-

1990. Like Perl, Python source code is also available under the GNU General Public License(GPL).

WHY TO LEARN PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

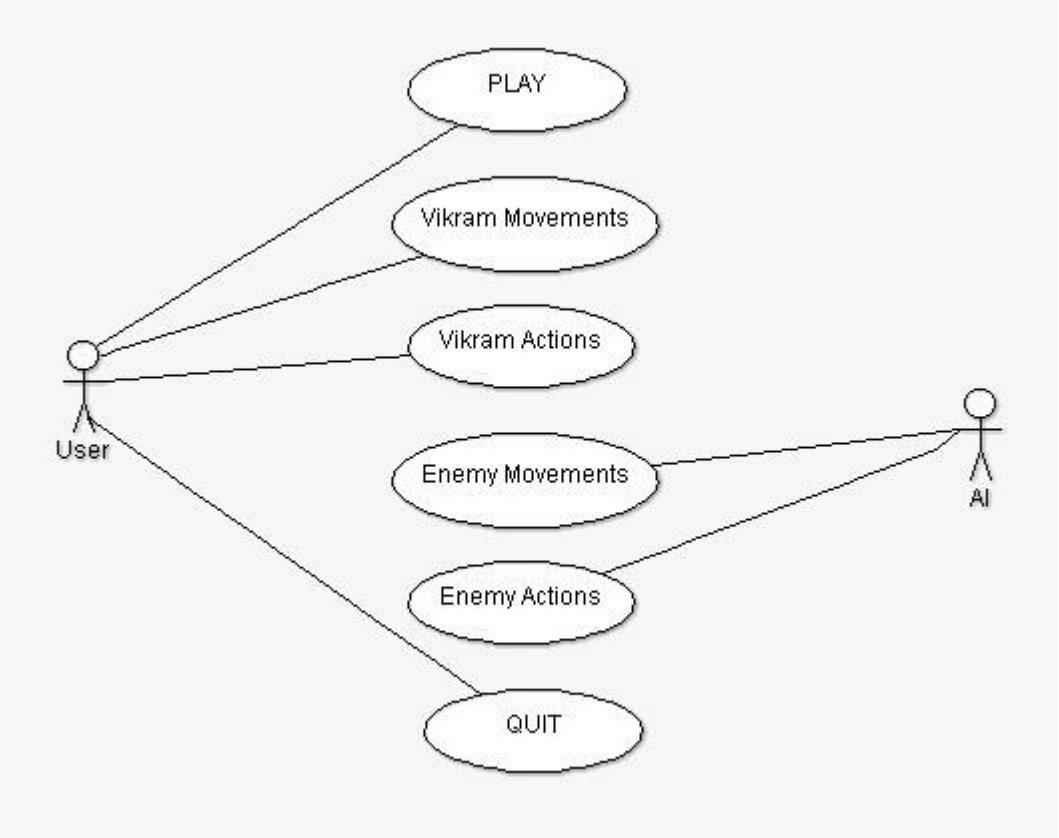
CHAPTER 3

DIAGRAM

3.1 UML DIAGRAM

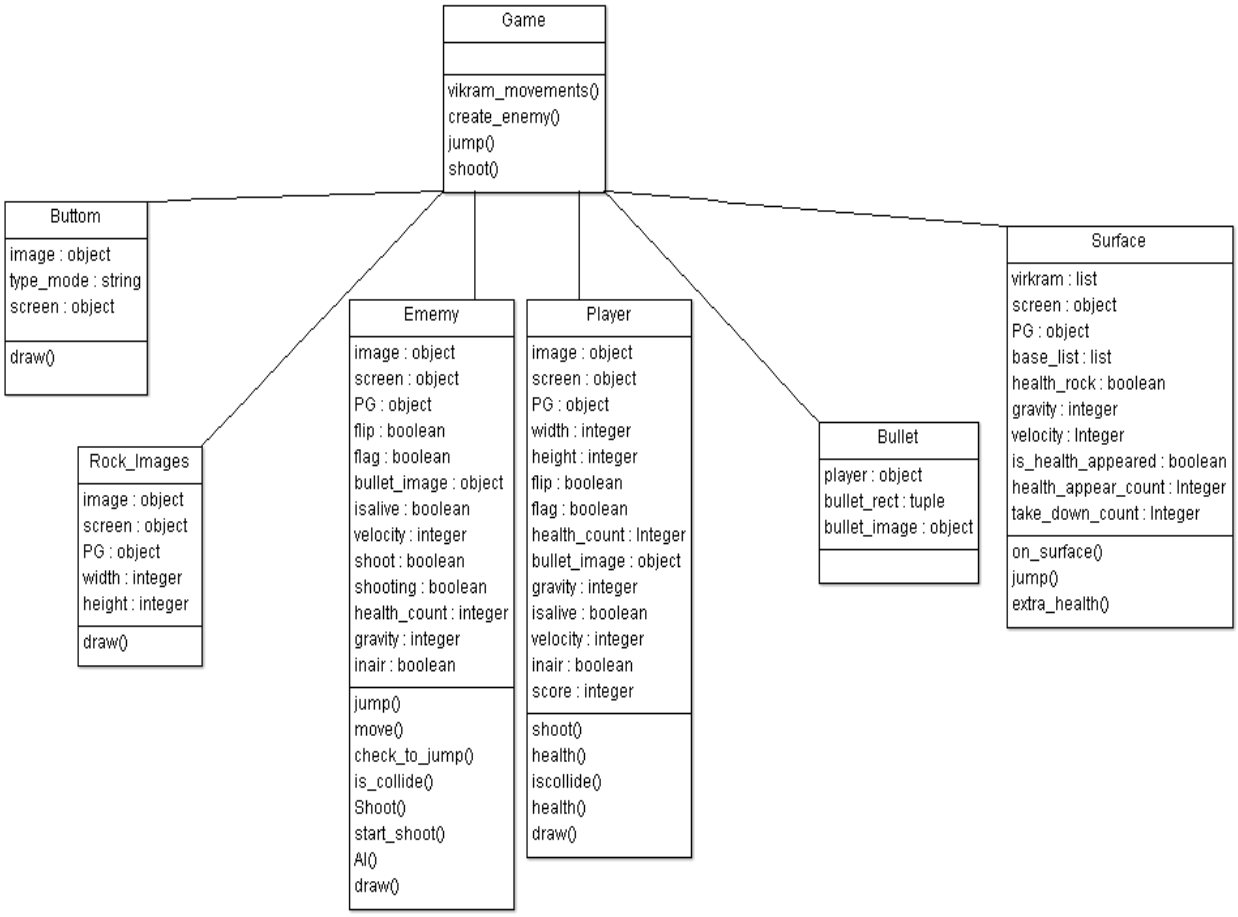
3.1.1 USECASE DIAGRAM

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It is represented using ellipse. Actor is any external entity that makes use of the system being modeled. It is represented using stick figure.



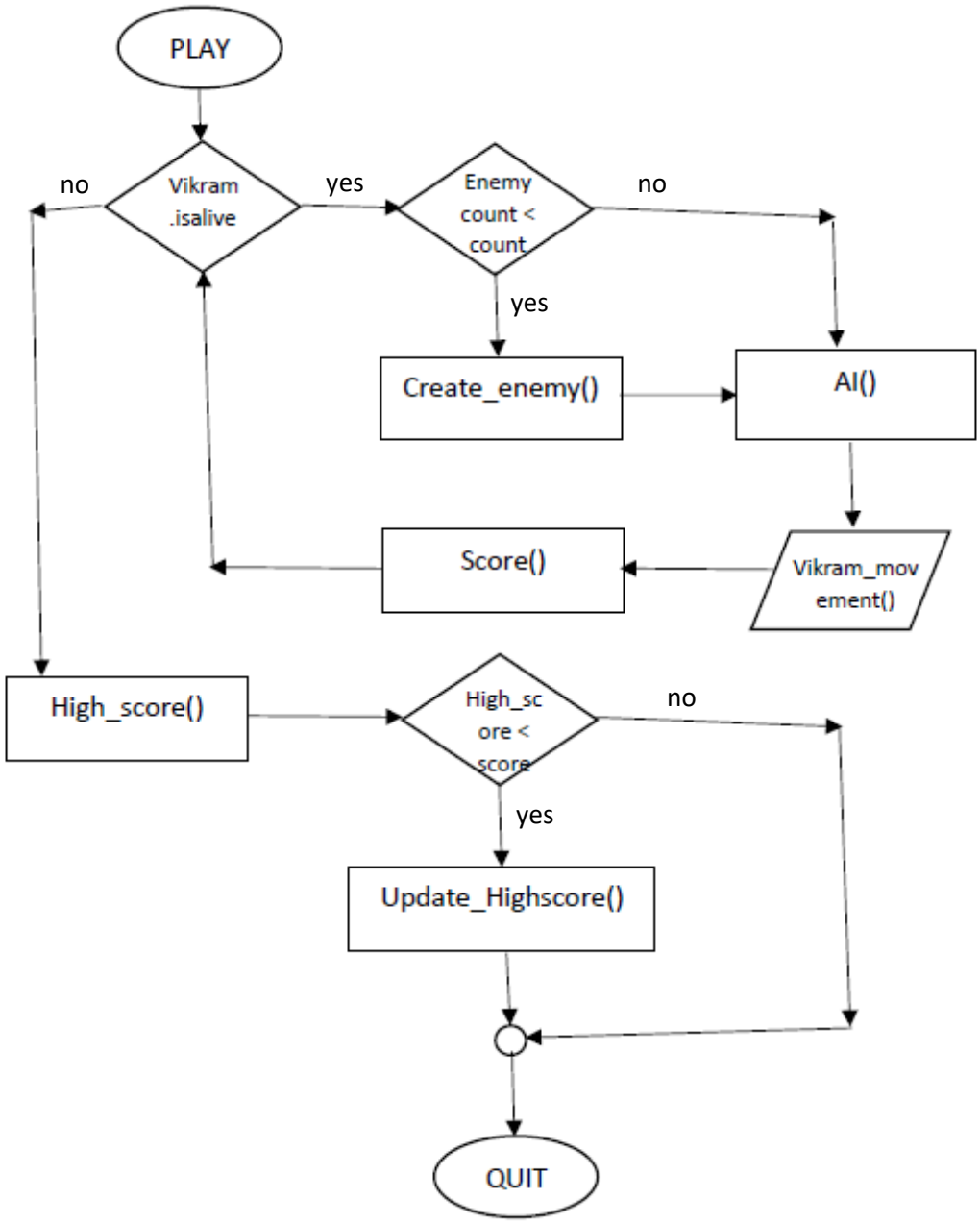
3.1.2 CLASS DIAGRAM

A class diagram in the unified markup language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. It is represented using a rectangle with three compartments. Top compartment have the class name, middle compartment the attributes and the bottom compartment with operations.



3.1.3 FLOW CHART DIAGRAM

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.



CHAPTER 4

SYSTEM DESIGN

4.1 ANIMATION

Animation is a method of photographing successive drawings, models, or even puppets, to create an illusion of movement in a sequence.

4.1.1 GRAPHICS

A graphic is an image or visual representation of an object.

4.1.1.1 CHARACTERS

4.1.1.1.1 VIKRAM

Vikram – He is the player who is used by the players to encounter the enemies and level up their skills. He entered into the arena with five heart which is used to make him alive whenever he killed by enemies. And also he can improve the life by observing the heart which is occur in the game occasionally.

The following images are referred from the https://secrethideout.itch.io/team-wars-platformer-battle. And all are rendered to looks like the character is in motion.

- **Idle position:**

Fig 4.1: Vikram - Idle position

- **Jump:**

Fig 4.2: Vikram - Jumping

- **Run:**

Fig 4.3: Vikram - Running

- **Death:**

Fig 4.4: Vikram - Death

4.1.1.1.2 ENEMIES

Enemies – All are red in colour. And being produced by the AI throughout the game until the player-VIKRAM is shoot down without having a heart (life to alive). He is entered into the arena with five hearts, also he can increase his life by observing the heart while playing.

The following images are referred from the https://secrethideout.itch.io/team-wars-platformer-battle. And all are rendered to looks like the character is in motion.

The count of the enemies will increase when the player attain its every 11^{th} point by encountering the enemies.

- **IDLE:**

Fig 4.5: Enemy - Idle position

- **JUMP:**

Fig 4.6: Enemy - Jumping

- **RUN:**

Fig 4.7: Enemy - Running

- **DEATH:**

Fig 4.8: Enemy - Dead

4.1.1.2 OBJECTS

4.1.1.2.1 GUN

The gun, named as Shotgun is used in this game VIKRAM THE GHOST. Gun is used by both the player and enemies. And both of them use that to take them down vise versa. It can fire one bullet at one time and the one bullet is enough to encounter the enemies as well as the player.

4.1.1.2.2 BULLET

Fig 4.9: Bullet

The bullet is fired though the shotgun. It can fire one bullet every time when the player as well as the enemy trigger the shotgun. Each bullet which are fired though the shotgun has the power to kill enemies and the player vikram.

4.1.1.2.3 HEALTH

Fig 4.10: Heart

The heart, used by the player to stay alive in the game. The player enter into the arena with five hearts. Whenever he shot by the enemies, he lost his heart. And also he can increase his life by observing the extra heart appears while playing.\

4.1.1.2.4 ARENA

The battle ground is composed of following figures named as Background, Main rock, Big rock, Floating rock 1, Floating rock 2, Floating rock 3.

Fig 4.11: Background image

Fig 4.12: Main rock

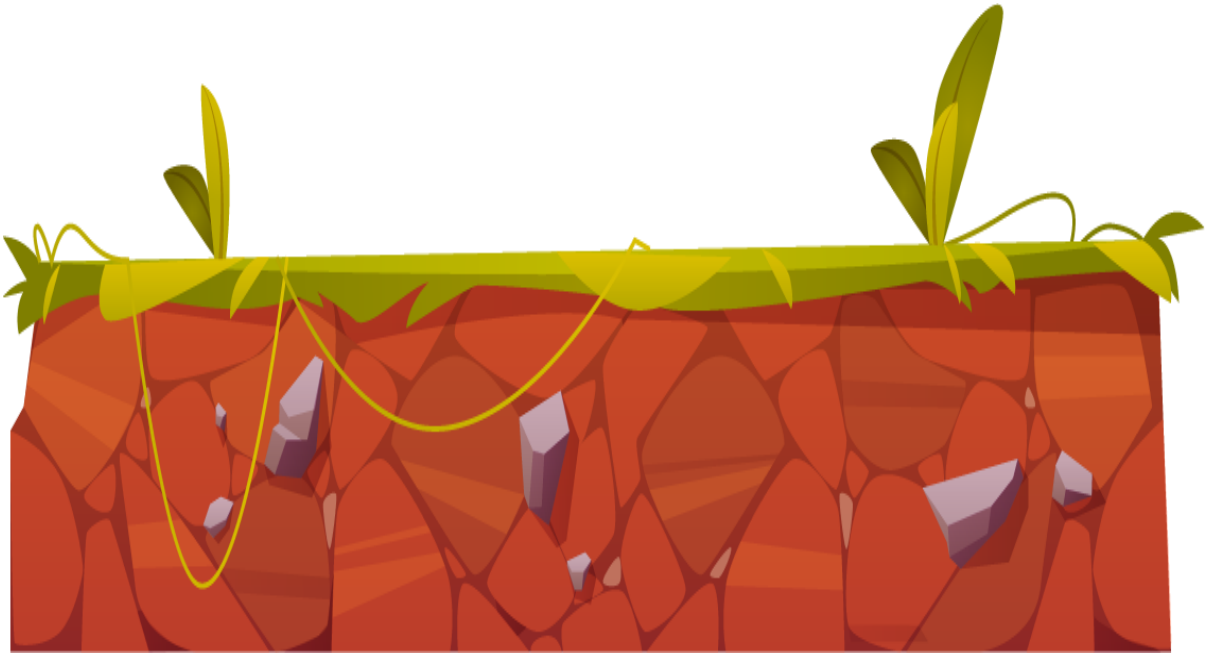


Fig 4.13: Big rock

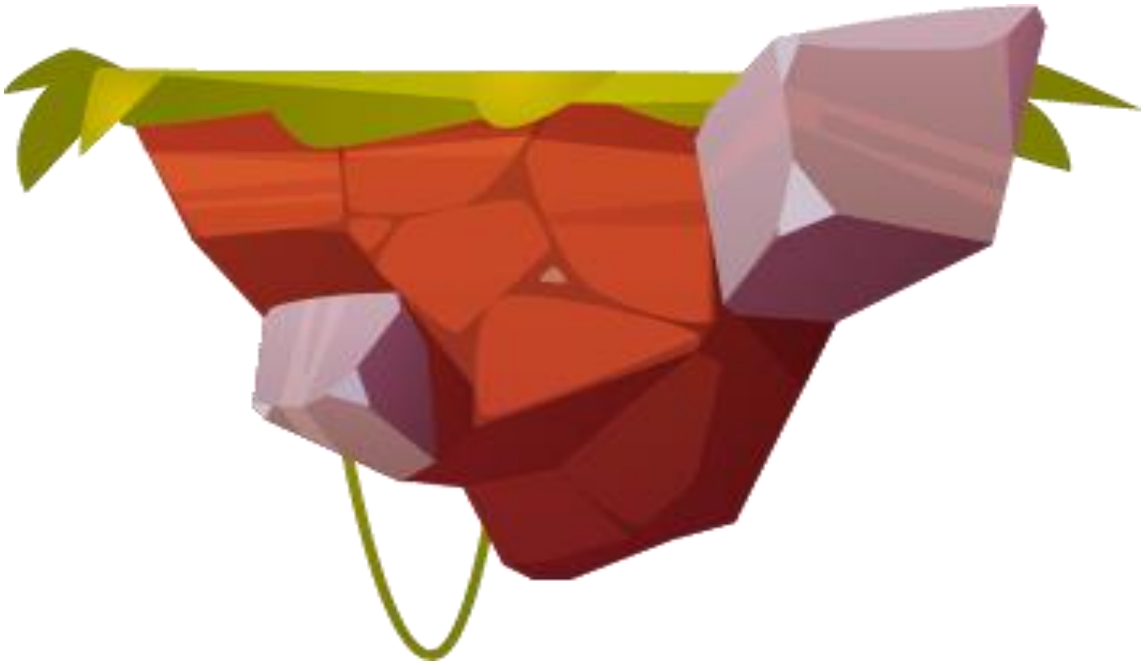


Fig 4.14: Floating rock 1

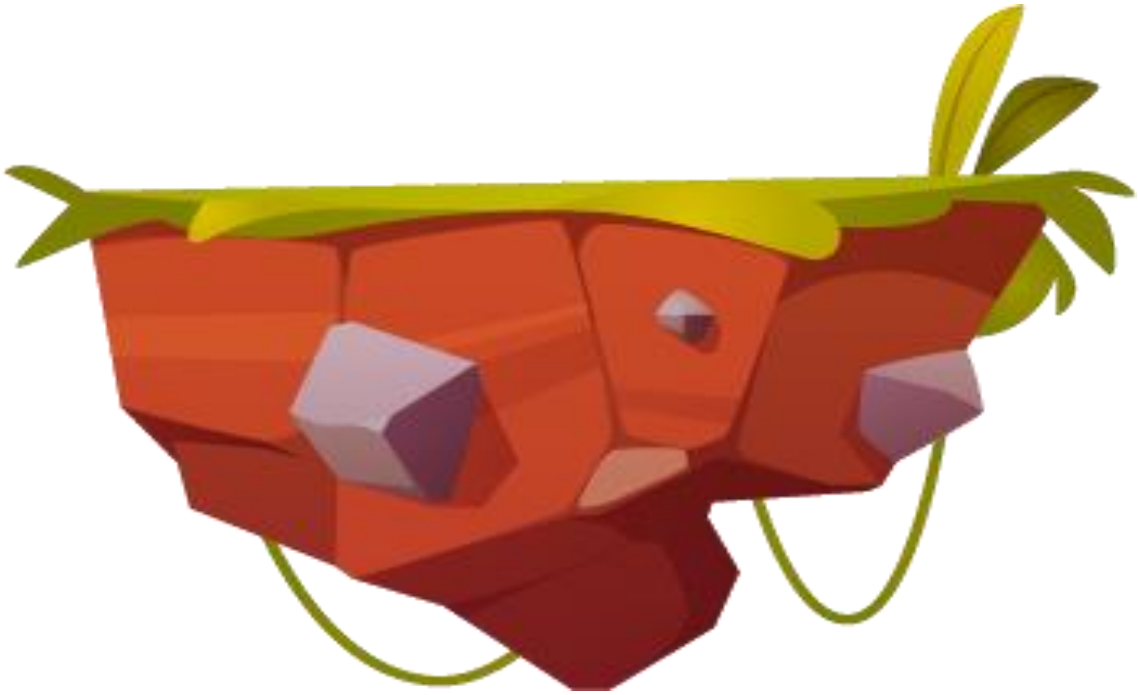


Fig 4.15: Floating rock 2

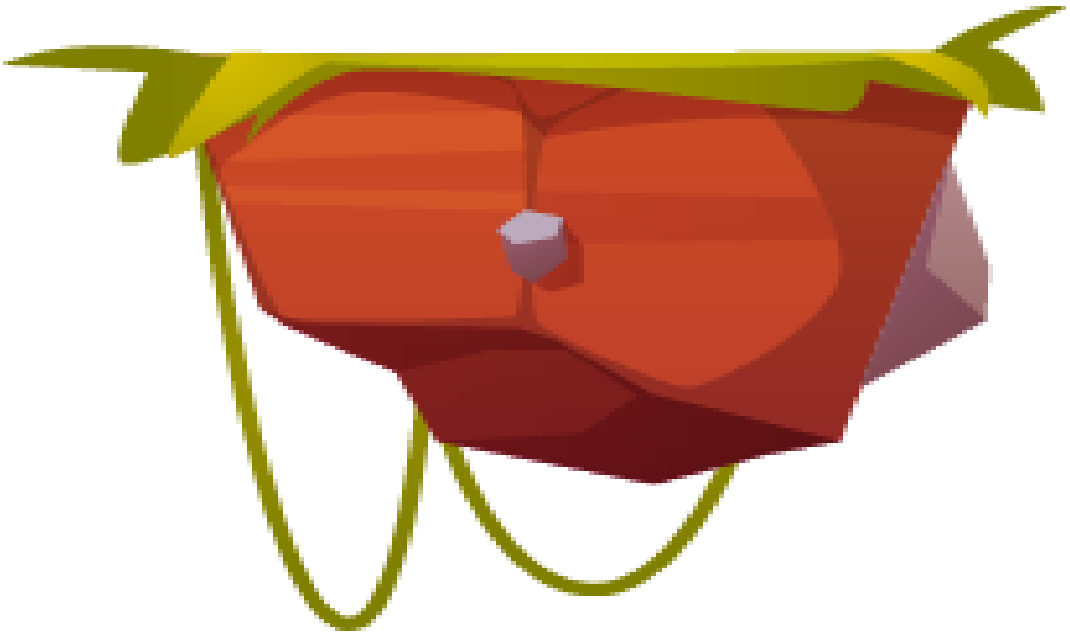


Fig 4.16: Floating rock 3

CHAPTER 5

CODING

5.1 CODING

Coding is the generic term which can have a different meaning based on the context in which it is used. It is a process to write a specific set of instructions that a computer can understand and execute. Computer coding is term used for writing codes and executing it for getting desired output.

5.1.1 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.

WHY WE USE PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

CHAPTER 6

SOURCE CODE

1) Firstly we import needed packages

```
import pygame as PG
import random
import Resources.Modules.Resource as R
```

2) Main menu for game processing

```
def MainMenu():
    tmd = PG.font.Font(r'Resources\Fonts\TMD.ttf',24)
    main_menu_image = PG.image.load('Resources\Images\Main\Mainmenu Image.png')
    main_menu_image = PG.transform.scale(main_menu_image, (650,600))
    start_img = PG.image.load('Resources\Images\Main\play.png').convert_alpha()
    start_img = PG.transform.scale(start_img, (int(start_img.get_width() * 0.1), int(start_img.get_height() *0.1)))
    start_button = R.Button(640 , 480 , start_img, 'start', screen, PG)
    running = True
    while running:
        high_score = highscore()
        screen.fill((255,255,255))
        start_key = start_button.draw()
        if start_key:
            Game()
        for event in PG.event.get():
            if event.type == PG.QUIT:
```

```
            running = False
        new_record = tmd.render("High Score : " + str(high_score),True,(0,0,0))
        screen.blit(new_record,[20,540])
        screen.blit(main_menu_image,(75, -20))
        PG.display.update()
MainMenu()
```

3) Pygame initialition

```
PG.init()
Screen_width = 800
Screen_height = 580
screen = PG.display.set_mode([Screen_width,Screen_height])
clock = PG.time.Clock()
FPS = 60
def highscore():
    f = open('Resources\Temp\Cache.txt', mode = 'r')
    score = f.read()
    hscore = int(score.split("= ")[1])
    return hscore
```

4) Loading images and Rock creating

```
def Game():
    big_rock_1a =
PG.image.load('Resources\Images\Backgrounds\Big_rock.png')
    main_rock_1a =
PG.image.load('Resources\Images\Backgrounds\Main_rock.png')
    float_rock_1a = PG.image.load('Resources\Images\Backgrounds\Floating
rock 1.png')
    float_rock_2a = PG.image.load('Resources\Images\Backgrounds\Floating
rock 2.png')
    float_rock_3a = PG.image.load('Resources\Images\Backgrounds\Floating
rock 3.png')
```

```
    bullet_img = PG.image.load('Resources\Images\Action\Bullet.png')
    firing_img = PG.image.load('Resources\Images\Action\Firing.png')
    health_img = PG.image.load('Resources\Images\Health\Health.png')

    big_rock_1a = R.Rock_Images(Screen_width // 2, Screen_height, 0.3, 250,
big_rock_1a, screen, PG)
    float_rock_1a = R.Rock_Images(100, Screen_height - 120, 0.5, 50,
float_rock_1a, screen, PG)
    float_rock_2a = R.Rock_Images(300, Screen_height - 210, 0.4, 50,
float_rock_2a, screen, PG)
    float_rock_3a = R.Rock_Images(750, Screen_height - 120, 0.4, 50,
float_rock_3a, screen, PG)
    main_rock_1a = R.Rock_Images(Screen_width + 300, Screen_height, 0.3,
250, main_rock_1a, screen, PG)
    base_list = [big_rock_1a, float_rock_1a, float_rock_1b, float_rock_2a,
float_rock_2b, float_rock_2c, float_rock_3a, float_rock_3b, float_rock_3c,
main_rock_1a]
    float_rock_1 = [float_rock_1a, float_rock_1b]
    float_rock_2 = [float_rock_2a, float_rock_2b, float_rock_2c]
    float_rock_3 = [float_rock_3a, float_rock_3b, float_rock_3c]
    float_rock = [float_rock_1, float_rock_2, float_rock_3]
```

5) Vikram movement

```
def vikram_movement(moving_left, moving_right, shoot, base_list,
vikram_idle, vikram_run, virkam_jump):
    global vikram_run_key
    global shooting
    if vikram_idle[0].isalive:
        if moving_left:
            if vikram_idle[0].rect.x < 200:
                for i in base_list:
                    i.rect.x += 2
                for i in enemy_list:
                    for j in i:
                        for k in j:
                            k.rect.x += 2
                            #print(k.rect.x)
```

```
            for i in vikram_idle:
                i.flip = True
            for i in vikram_run:
                i.flip = True
            for i in vikram_jump:
                i.flip = True
            for i in vikram_death:
                i.flip = True
        else:
            for i in vikram_idle:
                i.rect.x -= 2
                #i.bullet_rect.x -=2
                i.flip = True
            for i in vikram_run:
                i.rect.x -= 2
                i.flip = True
            for i in vikram_jump:
                i.rect.x -= 2
                i.flip = True
            for i in vikram_death:
                i.rect.x -= 2
                i.flip = True

    elif moving_right:
        if vikram_idle[0].rect.x > 500:
            for i in base_list:
                i.rect.x -= 2
            for i in enemy_list:
                for j in i:
                    for k in j:
                        k.rect.x -= 2
                        #print(k.rect.x)
            for i in vikram_idle:
                i.flip = False
            for i in vikram_run:
                i.flip = False
            for i in vikram_jump:
                i.flip = False
```

```
                for i in vikram_death:
                    i.flip = False
            else:
                for i in vikram_idle:
                    i.rect.x += 2
                    #i.bullet_rect.x +=2
                    i.flip = False
                for i in vikram_run:
                    i.rect.x += 2
                    i.flip = False
                for i in vikram_jump:
                    i.rect.x += 2
                    i.flip = False
                for i in vikram_death:
                    i.rect.x += 2
                    i.flip = False
        if (shoot or shooting or vikram[1][0].bullet_list != []):

            if surface.vikram_inair == False:
                if shoot and len(vikram[1][0].bullet_list) < 2:
                    bullet = R.Bullet(vikram[1][0], [], True, bullet_img, PG)
                    vikram[1][0].bullet_list.append(bullet)
            if vikram[1][0].bullet_list != []:
                for i in vikram[1][0].bullet_list:
                    shooting = vikram_run[0].Shoot(vikram_run[0].rect.x +
vikram_run[0].width, shoot, i, vikram_run[0], enemy_list)
            shoot = False

        if len(enemy_list):
            enemy_list[0][0][0].AI(enemy_list, vikram_idle, enemy_state,
surface, e_start_jump, e_jump, e_inair)

        surface.extra_health()
```

6) Enemy creation

```
def create_enemy(e_c_c, e_c_t):
    global enemy_creation_count, enemy_creation_time, ec, enemy_limit
    #print(e_c_c, e_c_t)
```

```
        if e_c_c > e_c_t and len(enemy_list) < enemy_limit:
            #print(1)
            enemy_creation_count = 0
            enemy_creation_time = random.randint(ec, ec+2)
            enemy_creation_location = random.randint(0,len(base_list)-1)
            location = base_list[enemy_creation_location]
            enemy_idle, enemy_run, enemy_jump, Death = [], [], [], []
            for i in range(5):
                enemy = PG.image.load('Resources\Images\Enemy\Idle\%d.png' %
i)
                #print('Resources\Images\Vikram\Idle\%d.png' % i)
                enemy = R.Enemy(location.rect.x + (location.width // 2),
location.rect.top, location, 0.7, 0, enemy, bullet_img, firing_img, vikram,
screen, PG)
                enemy_idle.append(enemy)

            for i in range(6):
                enemy = PG.image.load('Resources\Images\Enemy\Run\%d.png' %
i)
                #print('Resources\Images\Vikram\Run\%d.png' % i)
                enemy = R.Enemy(location.rect.x + (location.width // 2) ,
location.rect.top, location, 0.7, 0, enemy, bullet_img, firing_img, vikram,
screen, PG)
                enemy_run.append(enemy)

            for i in range(2):
                enemy = PG.image.load('Resources\Images\Enemy\Jump\%d.png'
% i)
                #print('Resources\Images\Vikram\Run\%d.png' % i)
                enemy = R.Enemy(location.rect.x + (location.width // 2) ,
location.rect.top, location, 0.7, 0, enemy, bullet_img, firing_img, vikram,
screen, PG)
                enemy_jump.append(enemy)

            for i in range(8):
                enemy = PG.image.load('Resources\Images\Enemy\Death\%d.png'
% i)
```

```
            enemy = R.Enemy(location.rect.x + (location.width // 2) , location.rect.top, location, 0.7, 0, enemy, bullet_img, firing_img, vikram, screen, PG)
            Death.append(enemy)

        enemy_idle[0].death = Death
        enemy_list.append([enemy_idle, enemy_run, enemy_jump, Death])
        enemy_state.append([False, True, False])
        e_start_jump.append(False)
        e_jump.append(False)
        e_inair.append(False)
```

7) Artificial Intelligence for Enemy

```
def AI(self, enemy_list, vikram, enemy_state, surface, e_start_jump, e_jump, e_inair):

    for index1, enemy in enumerate(enemy_list):
        if enemy[0][0].isalive:
            #used to detect direction if enemy and hero not in same y
            local_temp_boolean_1 = False

            if enemy[0][0].rect.y < vikram[0].rect.top and enemy[0][0].rect.y not in range(vikram[0].rect.y - 40, vikram[0].rect.y + 40) and not enemy[0][0].inair:
                #print(1)
                if enemy[0][0].rect.y not in range(vikram[0].rect.y - 40, vikram[0].rect.y + 40) and not surface.vikram_inair:
                    #print(2)
                    if enemy[0][0].rect.x > vikram[0].rect.x - 600 and enemy[0][0].rect.x < vikram[0].rect.x + 600:
                        #print(3)
                        temp_boolean_1 = self.move(enemy, enemy_state, index1, vikram, 50)

                    else:
                        #print(4)
                        enemy_state[index1][1] = True
```

```
                    enemy_state[index1][0] = False
                    for i in enemy:
                        for j in i:
                            j.rect.x -= 1
                            j.flip = True
                else:
                    #print(6)
                    enemy_state[index1][1] = False
                    enemy_state[index1][0] = True
                #used to check whether we have to find nearby stone or already did
                enemy[0][0].temp_boolean_1 = True

            elif enemy[0][0].rect.y in range(vikram[0].rect.y - 40, vikram[0].rect.y + 40) and not enemy[0][0].inair:
                    #print(7)
                    local_temp_boolean_1 = self.move(enemy, enemy_state, index1, vikram, self.distance_range)
                    #used to check whether we have to find nearby stone or already did
                    enemy[0][0].temp_boolean_1 = True
                    #if enemy[0][0].x in range((location.rect.x + (location.width//2) - 10
                    #change_location()
                    temp = surface.on_surface(enemy, 0)
                    if enemy[1][0].bullet_list == [] and temp == [] and enemy[1][0].bullet_timer > 120:
                        enemy[1][0].shoot = True
                        enemy[1][0].bullet_timer = 0
                    enemy[1][0].bullet_timer += 3
                    enemy[1][0].start_shoot(enemy, vikram)
            else:
                #print(15)
                '''if enemy[0].temp_boolean_1:
                    print('if')
                    #local_temp_boolean_1 = self.move(enemy, enemy_state, index1, vikram, self.distance_range)'''
                if True:
```

```
                #print('else')
                self.check_to_jump(enemy, surface.base_list,
enemy[0][0].temp_boolean_1, enemy_state, index1, e_start_jump, surface,
vikram)
                enemy[0][0].temp_boolean_1 = False
        else:
            enemy_state[index1][2] = False
            enemy_state[index1][1] = False
            enemy_state[index1][0] = False
            if enemy[0][0].i < 47:
                enemy[0][0].death[int(enemy[0][0].i//6)].draw()
                enemy[0][0].i += 1
            else:
                enemy_list.remove(enemy)
```

8) Shooting

```
def Shoot(self, limit, shoot, i, player, enemy_list):
    if shoot:
        i.bullet_rect.y = player.rect.y + ((player.height // 2) - 2)
        firing =
self.PG.transform.scale(self.firing_image,(int(self.image.get_width() *
0.2),int(self.image.get_height() * 0.1)))

        if self.flip:
            self.flag = 1
            i.bullet_rect.x = (player.rect.x + 2)
            self.screen.blit(self.PG.transform.flip(firing, self.flip, False),
(i.bullet_rect.x - 10.5, i.bullet_rect.y-0.8))
        else:
            self.flag = 0
            i.bullet_rect.x = (player.rect.x + player.width) - 2
            self.screen.blit(self.PG.transform.flip(firing, self.flip, False),
(i.bullet_rect.x, i.bullet_rect.y-0.8))
    if self.flag:
        i.bullet_rect.x -= 15
    else:
        i.bullet_rect.x += 15
```

```
        local_boolean = self.iscollide(i, enemy_list, player)
        if i.bullet_rect.x > limit + 350 or i.bullet_rect.x < limit - 400 or
    local_boolean:
            #print("i am Staying here")
            if self.flag == 1:
                i.bullet_rect.x = (player.rect.x + 2)
            else:
                i.bullet_rect.x = (player.rect.x + self.width) - 2
            i.bullet_rect.y = player.rect.y + ((player.height // 2) - 2)
            player.bullet_list.remove(i)
            return False
        self.screen.blit(self.PG.transform.flip(i.bullet, self.flip, False),
    (i.bullet_rect.x, i.bullet_rect.y))
```

9) Health

```
    def health(self):
        x = 10
        for i in range(self.health_count):
            self.screen.blit(self.p_health, (self.h_rect.x + x, self.h_rect.y))
            x = x + 25
        if self.health_count < 1:
            self.isalive = False
```

10) Bullet

```
    class Bullet:
def __init__(self, player, enemy, boolean, bullet_image, PG):
   self.player = player
   self.enemy = enemy
   self.bullet_image = bullet_image
```

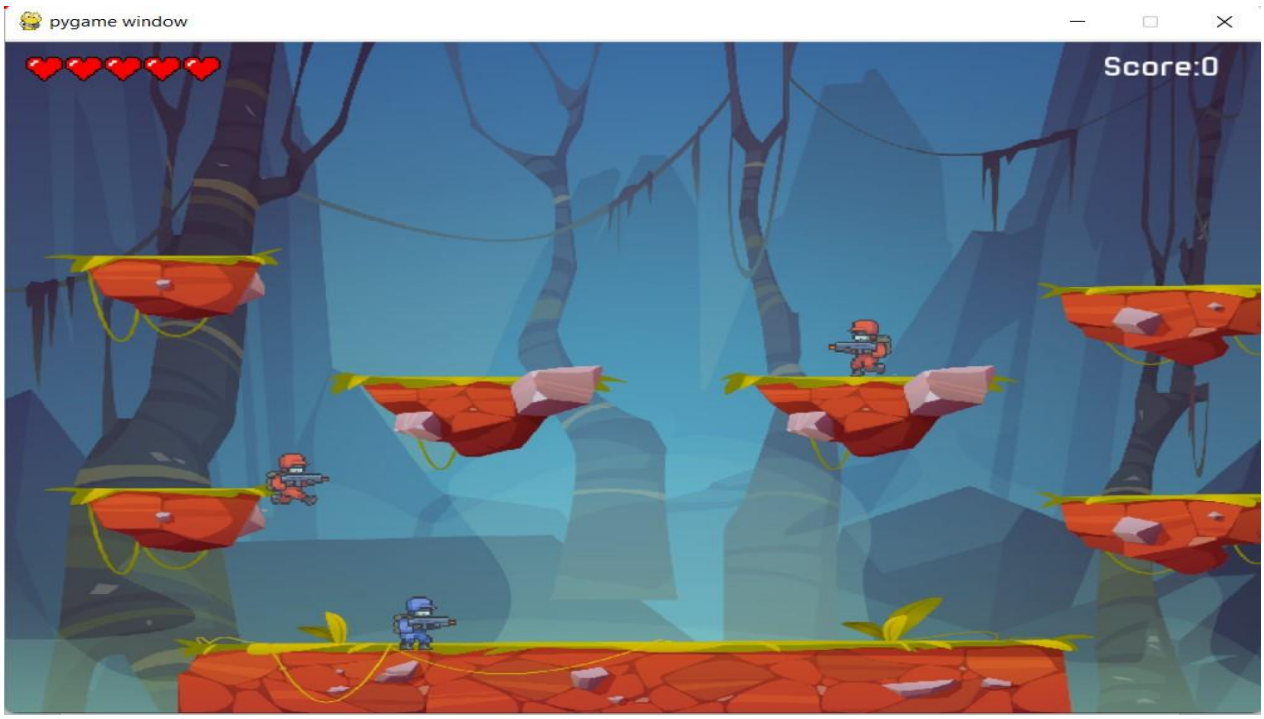
```
        self.bullet = PG.transform.scale(self.bullet_image,(int(player.image.get_width() * 0.2),int(player.image.get_height() * 0.05)))

        self.bullet_rect = self.bullet.get_rect()

        self.bullet_rect.x = (player.rect.x + player.width) - 2

        self.bullet_rect.y = player.y - ((player.height // 2) + 3)
```

11) Extra health

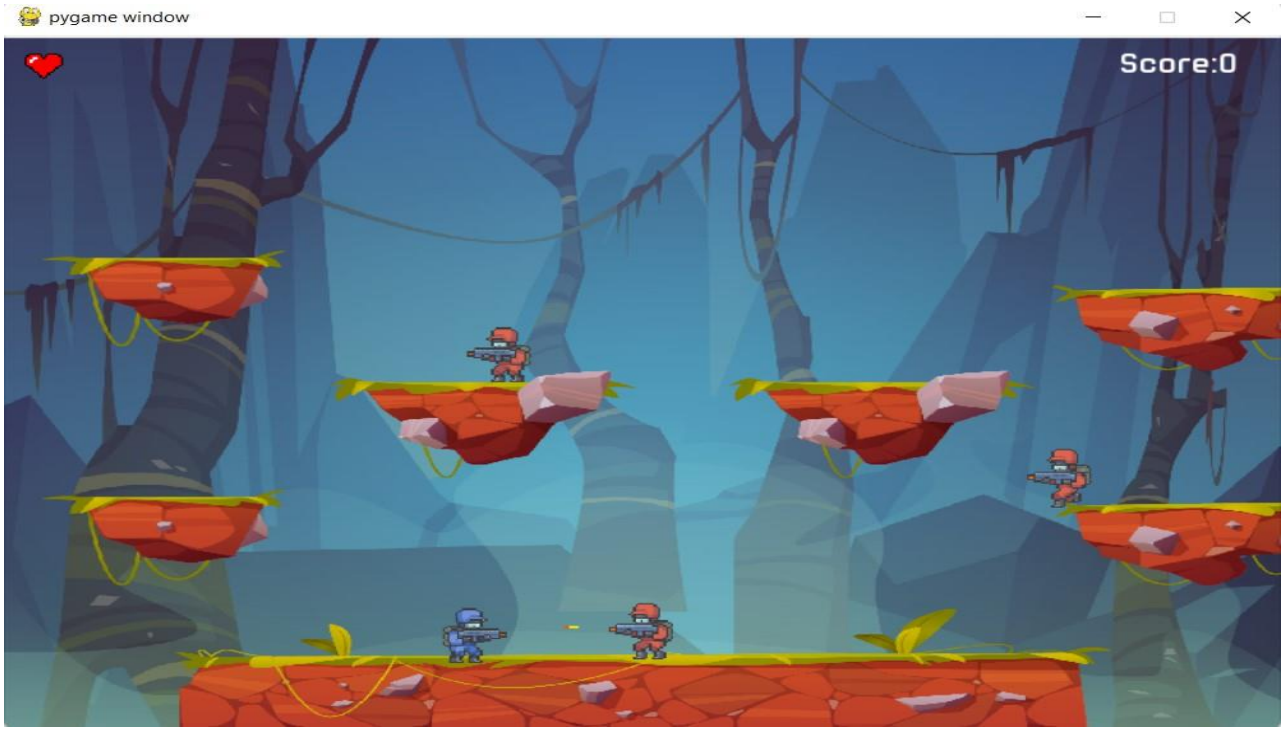
```
    def extra_health(self):
        self.health_appear_count += 1
        #print(self.health_appear_count)
        if not self.is_health_appeared:
            if self.health_appear_count > 360:
                self.health_appear_count = 0
                self.is_health_appeared = True
        else:
            #print('heart')

    self.screen.blit(self.vikram[0][0].p_health,(int(self.float_rock[2][2].rect.x + (self.float_rock[2][2].width//2) - 10 ),int(self.float_rock[2][2].rect.y)-10))
            #print(self.vikram[0][0].rect.x, self.float_rock[2][2].rect.x + (self.float_rock[2][2].width//2) - 10, self.vikram[0][0].rect.bottom, self.float_rock[2][2].rect.y-10)
            if self.vikram[0][0].rect.x in range(int(self.float_rock[2][2].rect.x + (self.float_rock[2][2].width//2) - 38 ),int(self.float_rock[2][2].rect.x + (self.float_rock[2][2].width//2) - 18 )) and int(self.vikram[0][0].rect.bottom) in range(self.float_rock[2][2].rect.top +28, self.float_rock[2][2].rect.top + 57):
                self.is_health_appeared = False
                self.health_appear_count = 0
                self.vikram[0][0].health_count += 1
            if self.health_appear_count > 200:
                self.is_health_appeared = False
                self.health_appear_count = 0
```

CHAPTER 7
SCREENSHOTS

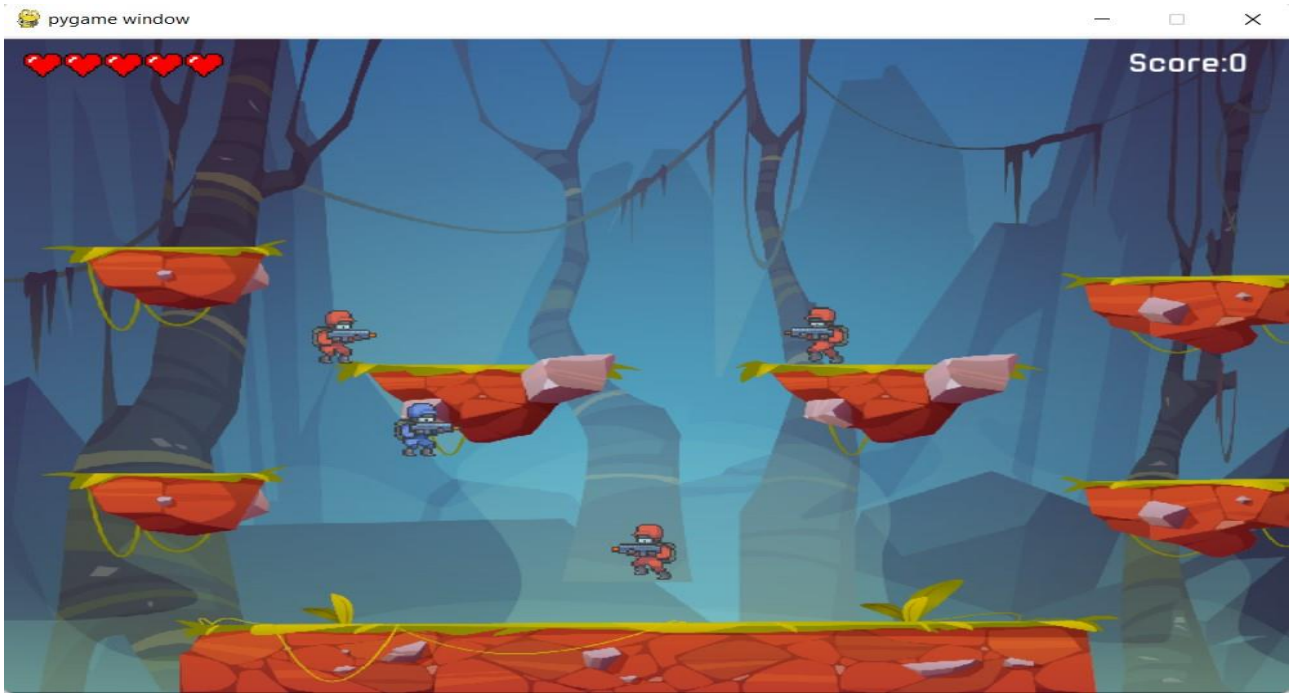
MAIN MENU



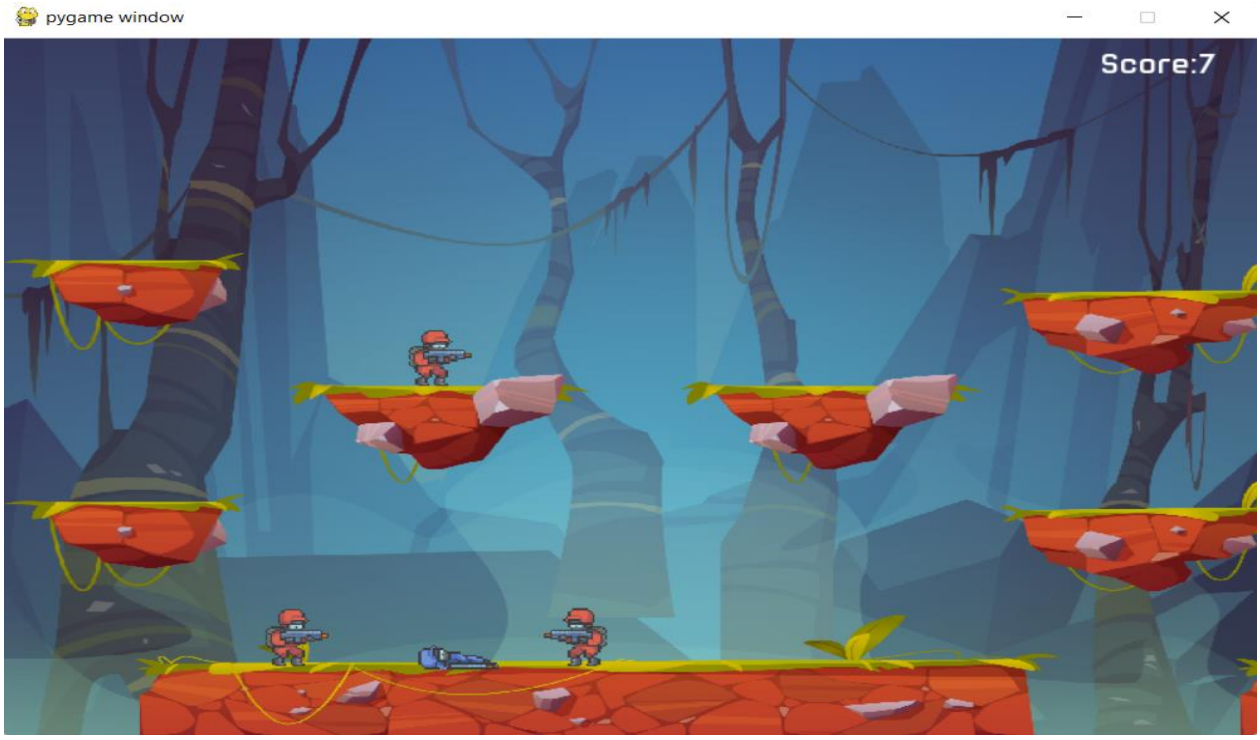
GAME STARTING



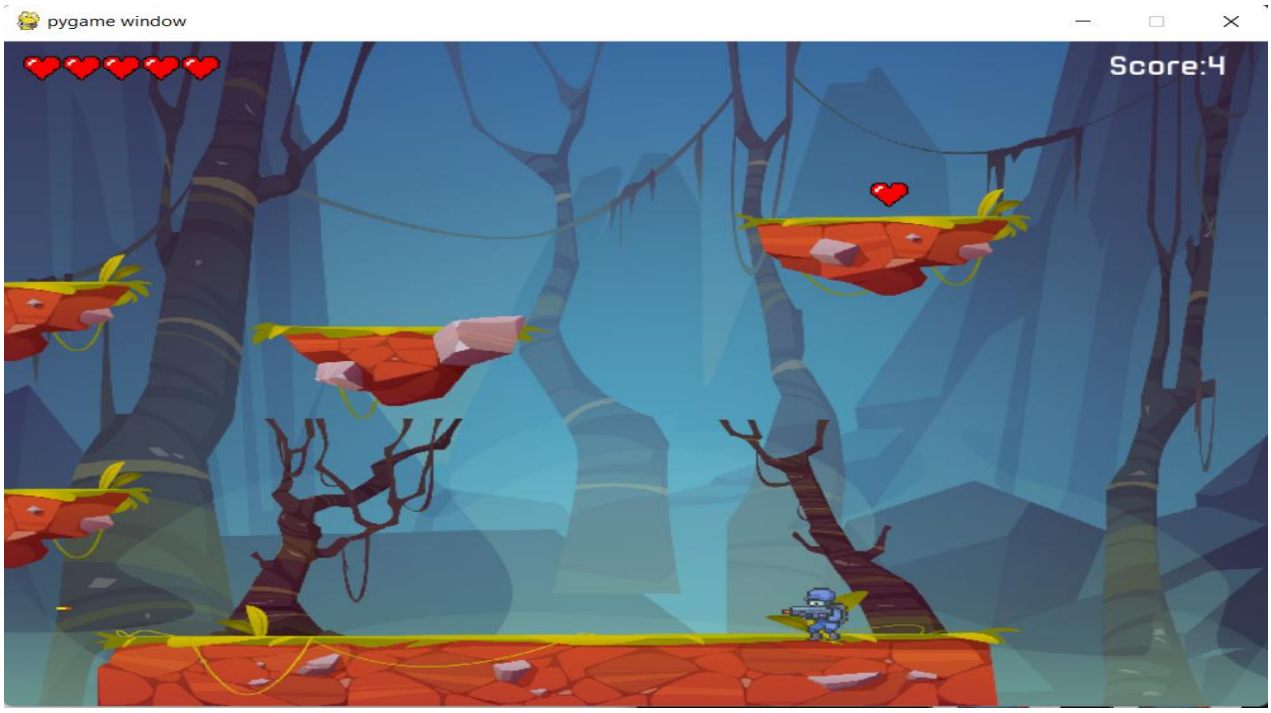
SHOOTING



JUMPING



DEATH OF VIKRAM



EXTRA HEALTH

SCORE OF PLAYER

HIGH SCORE

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

In this game, we have implemented the single player mode game, the player is named as Vikram and AI generating enemies with following the idea of OOP. Player and Enemies are provided with the shotgun to encounter their enemies. Player can gain extra health to make the gameplay more intractable, also added animations to all of the possible game objects including the user Interface.

This game is solely made for the user's entertainment purpose and its principle objective was to make a game that is similar to old Pixel Art RPG games but yet to have modem vibe.

8.2 FUTURE ENHANCEMENT

This game is made for PC at the present. There are many scopes available for the improvement of this game by including some additional functionalities.

Some of the plans in future of this application could possibly be:

- To make an Android and iOS version.
- To add guns with various kind of firing rate and damage.
- Adding levels to level up the character Vikram.
- To change the game art entirely since now game is implemented with free assets to design the levels.

REFERENCES

1. Background image and surface rock image. Available from: Free Vector | Game platform cartoon forest landscape, 2d design (freepik.com)
2. Vikram characters. Available from: https://secrethideout.itch.io/team-wars-platformer-battle
3. Enemy characters. Available from: https://secrethideout.itch.io/team-wars-platformer-battle
4. Pygame Documentation. Pygame Front Page — pygame v2.1.1 documentation
5. Python Documentation. 3.10.5 Documentation (python.org)
6. Real python. Python Tutorials – Real Python
7. Stack Overflow. Stack Overflow - Where Developers Learn, Share, & Build Careers