Project Report

On

**ENEMY SHOOTER**

**(A GAME MADE IN PYTHON)**

Submitted to:

**Panjab University, Chandigarh**

In the partial fulfilment of the requirement for the degree of

**Bachelor of Computer Applications (B.C.A.)**

**(Session – 2020-2021)**



**Under the Supervision of: Submitted by:**

Dr. Harmunish Taneja Prince Bansal

Asstt. Prof. B.C.A 6th Semester

Deptt Of Computer Sc. & IT Roll No. 18046489

**DAV COLLEGE, SECTOR-10, CHANDIGARH**

**DAV COLLEGE, SECTOR-10, CHANDIGARH**

**Certificate**

This is to certify that **Mr. Prince Bansal**, Class Roll No. **8018** a bonafide student of B.C.A. 6th Sem being run by DAV College, Chandigarh of batch 2020-2021 has completed the project entitled **“ Enemy Shooter Game In Python”** under my supervision & Guidance. It is further certified that the work done in this project is a result of candidate’s own efforts.

I wish him/her all success in his/her life.

Date: **Dr. Harmunish Taneja**

Asstt. Prof.

Deptt. of Comp. Sc. & IT

**Acknowledgement**

It is our privilege to express our sincerest regards to our project coordinator, Dr. Harmunish Taneja, for their valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project.

I deeply express our sincere thanks to our Head of Department Dr. Minakshi Bhardwaj encouraging and allowing us to present the project on the topic “Enemy Shooter Game “at our department premises for the partial fulfillment of the requirements leading to the award of

B.C.A degree.

I would like to thank my Principal, Dr. Pawan Kumar Sharma for providing the necessary facilities required for completion of this project.

We take this opportunity to thank all our lecturers who have directly or indirectly helped our project. We pay our respects and love to our parents and all other family members and friends for their love and encouragement through out our career. Last but not the least we express our thanks to our friends for their cooperation and support.

**Prince Bansal**

**BCA 6th Semester**

**Roll No. :- 18046489**

Contents

[1. Introduction to project 1](#_Toc74603252)

[**1.1** **Why I use python?** 1](#_Toc74603253)

[**1.2** **Why pygame?** 2](#_Toc74603254)

[**1.3** **Why this project?** 2](#_Toc74603255)

[**2.** **Aim** 3](#_Toc74603256)

[**3.** **Requirement Analysis** 3](#_Toc74603257)

[**3.1** **Problem Analysis** 3](#_Toc74603258)

[**3.2** **Introduction to SRS** 4](#_Toc74603259)

[**3.3** **General Description of project** 4](#_Toc74603260)

[**3.4** **Certain specific requirements** 5](#_Toc74603261)

[**4.** **Software Design** 7](#_Toc74603262)

[**4.1** **System Design** 7](#_Toc74603263)

[**a)** **Data Flow Diagram** 7](#_Toc74603264)

[**b)** **User Interface Design** 8](#_Toc74603265)

[**5.Coding** 11](#_Toc74603266)

[**6.** **Testing** 35](#_Toc74603267)

[**6.1** **Testing criteria** 35](#_Toc74603268)

[**6.2** **Test cases** 35](#_Toc74603269)

[**a)** **Login System** 35](#_Toc74603270)

[**b)** **Registration System** 37](#_Toc74603271)

[**c)** **Game** 39](#_Toc74603272)

[**7.** **Implementation of project** 40](#_Toc74603273)

[**8.** **Evaluation of project** 49](#_Toc74603274)

[**9.** **Conclusion** 50](#_Toc74603275)

[**10.** **Further Scope of project** 50](#_Toc74603276)

[**11.** **Bibliography** 51](#_Toc74603277)

1. Introduction to project

This project is a 2D video game in python, the goal of using python for developing this game is that it an open source software and have very easy syntax and also an used all over the world for developing various products. The core of this game is, Pygame, a set of python modules designed for game development, which was used for various tasks such as blitting images, moving these images, defining number of frames per second, adding audio to game, etc.

The end product is a level game in which a player has to move towards the end gate in each level, by killing enemies, collecting coins, various item boxes such as Health Box, Grenade Box, Bullet Box coming an the way. The enemy is also intelligent which will also start shooting the player when player comes close to it. The player can use bullets and grenades to kill enemies. But, player has definite number of bullets and grenades for each level . Initially, this game has total of 9 levels but we can add more levels in future if needed.

In this project, before playing the game the player has to login or if player doesn’t want to login then he can also play as Guest in which player will be assigned a unique Guest Id which will be stored in database. After logging in to the game, a level selection page will be displayed in which player had to select the level among the unlocked levels which he want to play. And at the end of each level, the score which he created during the level will be stored in the database.

This project is dynamic in nature, because when player reaches the end of each level, player’s rank will be counted automatically and displayed to him.

* 1. **Why I use python?**

To build a game, I have to use many core programming skills. In game development, I need number of conditional loops, variables, functions, object oriented programming approaches, and a whole bunch of programming techniques and algorithms.

In the python ecosystem, I found a rich set of tools, libraries, modules and frameworks that will help me to build a game more designable, and more easily. In python, there will be various predefined modules are available, such as Pygame, Pyglet , Pyganim,etc which are specially designed to build games.

Another most important reason behind the chosen of python is its simplicity in its syntaxes and its object oriented approach.

* 1. **Why pygame?**

In python, since various modules are available but I choose Pygame because it 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. And also it is

* **It's Simple** and easy to use. Kids and adults make shooter games with pygame. Pygame is used in the OLPC project and has been taught in essay courses to young kids and college students. It's also used by people who first programmed in z80 assembler or c64 basic.
* **Many games have been published**. Including Indie Game Festival finalists, Australian Game festival finalists, popular shareware, multimedia projects and open source games. Over 660 projects have been published on the pygame websites such as: list needed. Many more games have been released with SDL (which pygame is based on), so you can be sure much of it has been tested well by millions of users.
* **You control your main loop**. You call pygame functions, they don't call your functions. This gives you greater control when using other libraries, and for different types of programs.
* **Does not require a GUI to use all functions**. You can use pygame from a command line if you want to use it just to process images, get joystick input, or play sounds.
* **Fast response to reported bugs**. Some bugs are patched within an hour of being reported. Do a search on our mailing list for BUG... you'll see for yourself. Sometimes we suck at bug fixes, but mostly we're pretty good bug fixers. Bug reports are quite rare these days, since a lot of them have been fixed already.
* **Small amount of code**. It does not have hundreds of thousands of lines of code for things you won't use anyway. The core is kept simple, and extra things like GUI libraries, and effects are developed separately outside of pygame.
* **Modular**. You can use pieces of pygame separately. Want to use a different sound library? That's fine. Many of the core modules can be initialized and used separately.
  1. **Why this project?**

Since the 1970s, people started to take interest in using their computers as an entertainment environment, thus, the multi billion game industry was starting to take shape. Having presented earlier the sum of money this industry produces, I decided to have a go and create a game of my own.

As a kid, I was always fascinated by the idea of becoming a game developer, but, as years went by, I have realised this is not exactly what programming and computer science, as a practice, are about and I dropped the idea. However, the third year project offered me the possibility to try and achieve one of my childhood’s dreams and I couldn’t resist the temptation.

1. **Aim**

The aim of this project is to build a 2D video game using python from a scratch, which will be used as a source of entertainment in this COVID pandemic period when whole world in fear of COVID viral disease. Video games will reduces stress, reduces feelings of isolation, reduces feelings of loneliness, increases feelings of competence, increases engagement and empathy

Video games also helps children to learn new skills and engaging in latest technologies.

And also a game, which has a login and register interface so that player can save her scores of each level and the total time he will take to complete each level. So that he will be able to see her previous score in future when he logged in again.

1. **Requirement Analysis**
   1. **Problem Analysis**

The problem here is to develop a 2D video game with database accessibility which will provide various functionality to user like :

* User registration
* User Login System
* Play as a guest option
* Generate a unique Guest Id when guest option is chosen
* Level selection system : So that can select the level which he wants to play
* Unlock new level when user completes the previous level
* Showing total score of all levels and updated rank of player based on total scores of all users when he completes each level.
* High scores option which can be used to display top 6 scores till yet.
* Adding timer to game to calculate the time user takes to complete each level
  1. **Introduction to SRS**

A software requirements specification (**SRS**) is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill all users needs.

An **SRS** minimizes the time and effort required by developers to achieve desired **goals** and also minimizes the development cost. A good **SRS** defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations.

An SRS forms the basis of an organization’s entire project. It sets out the framework that all the development teams will follow. It provides critical information to all the teams, including development, operations, quality assurance ([QA](https://searchsoftwarequality.techtarget.com/definition/quality-assurance)) and maintenance, ensuring the teams are in agreement.

Using the SRS helps an enterprise confirm that the requirements are fulfilled and helps business leaders make decisions about the [lifecycle](https://whatis.techtarget.com/definition/product-lifecycle) of their product, such as when to retire a feature.

* 1. **General Description of project**
     1. **Characteristics**
* The project should be user friendly
* It should be completely consistent and secure .
* It should be reliable and robust.
* It should produce relevant error message.
* It should be easy to follow.
* It shouldn’t have hidden buttons
* It should use high graphics
  + 1. **Functions**
* This project provide user the login and register functionality
* It provide bullets and grenades to kill enemies . Each bullet will decrease enemy health by 20% and each grenade will decrease enemy’s health by 50%
* This game also have various item boxes such as Health box, Bullet box, Grenade box which will be used to increase player’s bullets, grenades and its health.
* Each bullet box will have 15 bullets and each grenade box have 3 grenades . And each health box increase player’s health by 50%.
  + 1. **Constraints**
* Maximum health will be 100%
* If health becomes 0 the player will die.
* Email Id should be unique
* New Users will play only first level . For play more levels, he should have to complete previous levels.
* Existing users can play Levels= Level Completed + 1
* Password can be Alphanumeric
* Initially the total score and total time is 0
  1. **Certain specific requirements**
     1. **Input requirements**
* A,D & W key are required to move forward, move backward and to jump respectively.
* Q key is used to throw grenades
* Space bar key is used to shoot bullets
* Mouse pointer is used to click buttons in game
* Keyboard keys are used to enter data in various text fields
  + 1. **Output requirements**
* Screen of minimum size 800 \* 640
  + 1. **Functional requirements**

Functional requirements of this project are:

* + - 1. To provide functionality of login system to players
      2. The health bar facility to player and enemies so that user can view the remaining health of enemies and himself also.
      3. Functionality to display remaining bullets and grenades during the whole level
      4. Adding health boxes, bullet boxes & grenade boxes so that player can increase her bullets, grenades & health.
      5. Registration system so that new users can register himself into game.
      6. To provide functionality to display rank at the end of each level.
      7. Level unlock system so that when user completes a level it automatically unlocks the next level.
      8. High scores displaying system
      9. Login as a Guest option for those users who doesn’t want to register on the game
      10. Timer functionality to track the time taken to complete each level
    1. **Software requirements**

1. **For development**

* XAMPP server – used for mysql database
* Microsoft Windows 10 – operating system used to develop this project
* Sublime Text 3 – used to write code
* Python 3.9 (64 bit) – used to compile and run .py (Python) files

1. **For deployment**

* Minimum Windows 7
* Mysql Server
  + 1. **Hardware requirements**

1. **For development**

* RAM 4GB
* Hard Disk 1TB
* Intel Processor 7th Gen

1. **For deployment**
   1. **Minimum**

* Pentium III or better processor
* 10 GB hard disk
* 256MB RAM
* 5 MBPS internet connection
  1. **Recommended**
* Intel processor
* 500GB hard disk
* 4GB RAM
* 10-100 MBPS internet connection

1. **Software Design**
   1. **System Design**
      1. **Data Flow Diagram**

LEVEL

SELECTION

Level

Game Inputs

GAMER

Score , Time

END

GAME

Game Outputs

Game Outputs

Game Inputs

Name, Email, Password, Country, Gender

Name, Levels Completed, Score, Time

Guest Id

DATABASE

**Bv**

(Player with 75>health>50)

A **data-flow diagram** is a way of representing a flow of data through a [process](https://en.wikipedia.org/wiki/Process) or a system (usually an [information system](https://en.wikipedia.org/wiki/Information_system)). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a [flowchart](https://en.wikipedia.org/wiki/Flowchart).

In this project, first flow of data starts from the user end. At first, user has three options i.e. login, register or guest. If user choose login, then user inputs two values i.e. email and password , and then user’s data will be fetched from the database corresponding to the given email and password.

But if user chooses register option, then data such as Username, Email, Password, Gender and Country will be inserted into the database. And if user chooses guest option, then a new Username is generated automatically and new row with this username is inserted in database.

After these options, data like username, total score, total time, levels completed, scores of each levels, time of each levels will be transferred to level selection process.

After level selection process, level number is transferred to main game engine. After main game engine, when player completes the level, its score and time is transferred to end game process and also updated to the database.

* + 1. **User Interface Design**

User interface design or UI design generally refers to the visual layout of the elements that a user might interact with in a software, or technological product. This could be the control buttons of a radio, or the visual layout of a software. User interface designs must not only be attractive to potential users, but must also be functional and created with users in mind.

User interface designs should be optimized so that the user can operate an application as quickly and easily as possible. Many experts believe that UI design should be simple and intuitive, often using metaphors from non-computer systems. With a more intuitive user interface design, users will be able to navigate around a software easily, finding the product or service they want quickly. One way to check the intuitiveness of a user interface design is through usability testing. The feedback from [usability testing](https://pidoco.com/en/help/ux/usability-testing) can then be used to optimize the user interface design of a [prototype](https://pidoco.com/en/help/ux/prototype) or final product.

REGISTER

PAGE

LOGIN PAGE

Login

REGISTER

Guest

LOGIN

REGISTER

START

GAME

PLAY

Shows Top 6 Scores

HIGH SCORES

If player dies

HELP

Plays same lvel

Plays next level

If level completes

Shows Help Box

LOGOUT

YOU

DIED



LEVEL COMPLETED

RESTART

NEXT LEVEL

MAIN MENU

(User Interface diagram of game)

* 1. **Component Level Design**
     1. **Login Page**

**Input**

Email, Password

Select user’s data

Where Email Id= Email

Connecting

To database

If Found

Level Selection Page

Message displayed

“Email & Password doesn’t match”

Yes

No

(Flow diagram of login page)

* + 1. **Register page**

Select user’s data

Where Email Id= Email

Connecting

To database

**Input**

Email, Password, Username, Gender, Country

Insert data in database

Message displayed

“Email already exists”

No

Yes

If Found

Login Page

(Flow diagram of register page)

**5.Coding**

**Main Page**

import pygame

from pygame import mixer

import os

import random

import button

import login

import label

import guest

import level\_selector

import level\_selector1

import csv

import level\_completion

import mysql.connector

import time

mixer.init()

pygame.init()

SCREEN\_WIDTH=800

SCREEN\_HEIGHT=int(SCREEN\_WIDTH\*0.8)

screen= pygame.display.set\_mode((SCREEN\_WIDTH,SCREEN\_HEIGHT))

pygame.display.set\_caption('Shooter')

guest\_no=0

#det frame rate

clock=pygame.time.Clock()

FPS=80

gravity = 0.5

scroll\_thresh=200

ROWS=16

COLS=200

TILE\_SIZE= SCREEN\_HEIGHT//ROWS

TILE\_TYPE=24

shoot=False

grenade=False

grenade\_thrown=False

level=1

level\_completed=0

TOTAL\_SCORE=0

start\_game=False

login\_check=False

screen\_scroll=0

bg\_scroll=0

#define action variables

moving\_left=False

moving\_right=False

bg\_img=pygame.image.load('img/extra/background5.png')

bg\_img=pygame.transform.scale(bg\_img,(SCREEN\_WIDTH,SCREEN\_HEIGHT))

back\_img=pygame.image.load('img/background/background.jfif').convert\_alpha()

back\_img=pygame.transform.scale(back\_img,(800,640))

# buttons images

restart\_img=pygame.image.load('img/button/restart.png').convert\_alpha()

box\_img=pygame.image.load('img/extra/board3.png')

box\_img=pygame.transform.scale(box\_img,(700,600))

box1\_img=pygame.image.load('img/extra/board4.png')

box1\_img=pygame.transform.scale(box1\_img,(600,500))

nextlevel\_img=pygame.image.load('img/button/nextlevel.png')

mainmenu\_img=pygame.image.load('img/button/mainmenu.png').convert\_alpha()

login\_img=pygame.image.load('img/button/login.png').convert\_alpha()

guest\_img=pygame.image.load('img/button/guest.png').convert\_alpha()

#item box image

bulletbox\_img=pygame.image.load('img/icons/bullet.png').convert\_alpha()

bulletbox\_img=pygame.transform.scale(bulletbox\_img,(25,25))

grenadebox\_img=pygame.image.load('img/icons/grenade.png').convert\_alpha()

grenadebox\_img=pygame.transform.scale(grenadebox\_img,(25,25))

healthbox\_img=pygame.image.load('img/icons/health.png').convert\_alpha()

healthbox\_img=pygame.transform.scale(healthbox\_img,(25,25))

coin\_img=pygame.image.load('img/background/17.png').convert\_alpha()

coin\_img=pygame.transform.scale(coin\_img,(30,30))

item\_boxes={

'Health' : healthbox\_img,

'Bullet' : bulletbox\_img,

'Grenade' : grenadebox\_img,

'Coin' : coin\_img

}

#buulet image

bullet\_image=pygame.image.load('img/bullet.png').convert\_alpha()

bullet\_image=pygame.transform.scale(bullet\_image,(8,8))

bullet1\_image=pygame.image.load('img/icons/bullet1.png').convert\_alpha()

bullet1\_image=pygame.transform.scale(bullet1\_image,(10,22))

#grenade image

grenade\_image=pygame.image.load('img/grenade.png').convert\_alpha()

grenade\_image=pygame.transform.scale(grenade\_image,(10,10))

# world data images

img\_list=[]

for x in range(TILE\_TYPE):

img=pygame.image.load(f'img/background/{x}.png').convert\_alpha()

img=pygame.transform.scale(img,(TILE\_SIZE,TILE\_SIZE))

img\_list.append(img)

pass

#define colors

BG= (144,201,120)

ORANGE=(255,140,0)

RED=(255,0,0)

GREEN=(0, 255, 0)

WHITE=(255,255,255)

font=pygame.font.SysFont('Futura',25)

def draw\_bars():

ammo\_bar\_img=bulletbox\_img

ammo\_bar\_img= pygame.transform.scale(ammo\_bar\_img,(25,25))

screen.blit(ammo\_bar\_img,(5,5))

grenade\_bar\_img=grenadebox\_img

grenade\_bar\_img= pygame.transform.scale(grenade\_bar\_img,(25,25))

screen.blit(grenade\_bar\_img,(5,35))

screen.blit(bullet1\_image,(40,5))

grenade\_img=grenade\_image

grenade\_img= pygame.transform.scale(grenade\_img,(15,20))

screen.blit(grenade\_img,(40,35))

img=font.render(f'X {player.ammo}',True, WHITE)

screen.blit(img,(60,5))

img1=font.render(f'X {player.grenades}',True, WHITE)

screen.blit(img1,(60,35))

img2=font.render(f'Score: {player.score}',True, WHITE)

screen.blit(img2,(700,5))

img3=font.render(f'Time: {player.elapsed\_time}',True, WHITE)

screen.blit(img3,(600,5))

#function to fill background color

def draw\_bg():

width=back\_img.get\_width()

for x in range(10):

screen.blit(back\_img,((x\*width)-bg\_scroll\*0.5,0))

pass

#function to create a player or enemy

def reset\_level():

enemy\_group.empty()

bullet\_group.empty()

grenade\_group.empty()

explosion\_group.empty()

item\_box\_group.empty()

decorations\_group.empty()

water\_group.empty()

exit\_box\_group.empty()

data=[]

for row in range(ROWS):

r=[-1]\*COLS

data.append(r)

pass

return data

pass

class Soldier(pygame.sprite.Sprite) :

def \_\_init\_\_(self, char, x, y, scale, speed,ammo,grenades):

pygame.sprite.Sprite.\_\_init\_\_(self)

#character type

self.char=char

self.alive=True

self.speed=speed

self.ammo=ammo

self.start\_ammo=ammo

self.shoot\_cooldown=0

self.grenades=grenades

self.health=100

self.max\_health=self.health

self.direction=1

self.in\_air=True

self.flip=False

self.jump=False

self.vel\_y=0

self.death\_counter=0

self.animatio\_list=[]

self.frame\_index=0

self.action =0

self.update\_time=pygame.time.get\_ticks()

self.score=0

# creating ai variabkes

self.move\_counter=0

self.vision=pygame.Rect(0,0,150,20)

self.idle=False

self.idle\_counter=0

self.time=time.time()

self.elapsed\_time=0

animation\_type=['idle','run','jump','diet']

for animation in animation\_type:

nu\_of\_frame=len(os.listdir(f'img/{self.char}/{animation}'))

temp\_list = []

for i in range(nu\_of\_frame):

img = pygame.image.load(f'img/{self.char}/{animation}/{i}.png').convert\_alpha()

img = pygame.transform.scale(img, (int(img.get\_width()\*scale),TILE\_SIZE))

temp\_list.append(img)

self.animatio\_list.append(temp\_list)

pass

self.image=self.animatio\_list[self.action][self.frame\_index]

self.rect= self.image.get\_rect()

self.rect.center = (x,y)

self.width=self.image.get\_width()

self.height=self.image.get\_height()

def update(self):

self.update\_animation()

self.check\_alive()

if self.shoot\_cooldown>0:

self.shoot\_cooldown-=1

def move(self, move\_left, move\_right):

screen\_scroll=0

dx=0

dy=0

if move\_left:

dx= -self.speed

self.flip=True

self.direction=-1

if move\_right:

dx=self.speed

self.flip=False

self.direction=1

if self.jump == True and self.in\_air==False:

self.vel\_y=-11

self.jump=False

self.in\_air=True

self.vel\_y += gravity

if self.vel\_y > 10:

self.vel\_y

dy += self.vel\_y

#check collision

for tile in world.obstacle\_list:

if tile[1].colliderect(self.rect.x+dx,self.rect.y,self.width,self.height):

dx=0

if self.char=='enemy':

self.direction\*=-1

self.move\_counter=0

pass

if tile[1].colliderect(self.rect.x,self.rect.y+dy,self.rect.width,self.height):

if self.vel\_y<0:

self.vel\_y=0

dy=tile[1].bottom-self.rect.top

elif self.vel\_y>=0:

self.vel\_y=0

self.in\_air=False

dy=tile[1].top-self.rect.bottom

if pygame.sprite.spritecollide(self,water\_group,False):

player.health=0

level\_complete=False

if pygame.sprite.spritecollide(self,exit\_box\_group,False):

dx=0

level\_complete=True

if level\_complete==False:

self.elapsed\_time=int(time.time()-self.time)

if self.rect.bottom>SCREEN\_HEIGHT:

player.health=0

if self.char=='player':

if (self.rect.left + dx<0 or self.rect.right+dx>SCREEN\_WIDTH):

dx=0

pass

pass

self.rect.x += dx

self.rect.y += dy

if self.char=='player':

if(self.rect.right>SCREEN\_WIDTH-scroll\_thresh and bg\_scroll<(world.level\_length\*TILE\_SIZE)-SCREEN\_WIDTH) or (self.rect.left<scroll\_thresh and bg\_scroll>abs(dx)):

self.rect.x-=dx

screen\_scroll=-dx

return screen\_scroll,level\_complete

def shoot(self):

if self.shoot\_cooldown==0 and self.ammo>0:

self.shoot\_cooldown=20

bullet=Bullet(self.rect.centerx+(0.75\*self.rect.size[0]\*self.direction),self.rect.centery-9,self.direction)

bullet\_group.add(bullet)

self.ammo-=1

def ai(self):

if self.alive and player.alive:

if self.idle==False and random.randint(1, 500)==1:

self.update\_action(0)

self.idle=True

self.idle\_counter=30

if self.vision.colliderect(player.rect):

self.update\_action(0)

self.shoot()

else:

if self.idle==False:

if self.direction==1:

ai\_moving\_right=True

else:

ai\_moving\_right=False

ai\_moving\_left=not ai\_moving\_right

self.move(ai\_moving\_left,ai\_moving\_right)

self.update\_action(1)

self.move\_counter+=1

self.vision.center=(self.rect.centerx+75 \* self.direction,self.rect.centery)

if self.move\_counter>10:

self.direction\*= -1

self.move\_counter\*=-1

pass

else:

self.idle\_counter-=1

if self.idle\_counter<=0:

self.idle=False

pass

self.rect.x+=screen\_scroll

def update\_animation(self):

Animation\_cool=150

self.image=self.animatio\_list[self.action][self.frame\_index]

if pygame.time.get\_ticks() - self.update\_time > Animation\_cool:

self.frame\_index +=1

self.update\_time=pygame.time.get\_ticks()

if self.frame\_index >= len(self.animatio\_list[self.action]):

if self.action==3:

self.frame\_index=len(self.animatio\_list[self.action])-1

else:

self.frame\_index=0

def update\_action(self, new\_action):

if(new\_action != self.action):

self.action=new\_action

self.frame\_index=0

self.update\_time=pygame.time.get\_ticks()

def check\_alive(self):

if self.health<=0:

self.health=0

self.speed=0

self.alive=False

self.update\_action(3)

def health\_bar(self):

pygame.draw.rect(screen,(204, 255, 255),pygame.Rect(self.rect.left,self.rect.top-20,80,10),0,10)

health\_bar=(self.health/self.max\_health)\*80;

if(self.health>75):

color=GREEN

elif (self.health>50):

color=ORANGE

else :

color=RED

pygame.draw.rect(screen,color,pygame.Rect(self.rect.left,self.rect.top-20,health\_bar,10),0,10)

def draw(self):

screen.blit(pygame.transform.flip(self.image, self.flip, False), self.rect)

self.health\_bar()

class World():

def \_\_init\_\_(self):

self.obstacle\_list=[]

def process\_data(self, data):

self.level\_length=len(data[1])

for y,row in enumerate(data):

for x,tile in enumerate(row):

if tile>=0:

img=img\_list[tile]

img\_rect=img.get\_rect()

img\_rect.x=x\*TILE\_SIZE

img\_rect.y=y\*TILE\_SIZE

tile\_data=(img,img\_rect)

if tile>=0 and tile<=8:

self.obstacle\_list.append(tile\_data)

elif (tile>=9 and tile<=10 or tile==16):

water=Water(img,x\*TILE\_SIZE,y\*TILE\_SIZE)

water\_group.add(water)

elif tile>=11 and tile<=13:

decorations=Decorations(img,x\*TILE\_SIZE,y\*TILE\_SIZE)

decorations\_group.add(decorations)

elif (tile==21):

player=Soldier('player',x\*TILE\_SIZE,y\*TILE\_SIZE,0.35,3, 20,5)

elif (tile==22):

enemy=Soldier('enemy',x\*TILE\_SIZE,y\*TILE\_SIZE,1,2, 100,0)

enemy\_group.add(enemy)

elif tile==17:

item\_box=Itembox('Coin',x\*TILE\_SIZE,y\*TILE\_SIZE)

item\_box\_group.add(item\_box)

elif (tile==18):

item\_box=Itembox('Bullet',x\*TILE\_SIZE,y\*TILE\_SIZE)

item\_box\_group.add(item\_box)

elif (tile==20):

item\_box=Itembox('Health',x\*TILE\_SIZE,y\*TILE\_SIZE)

item\_box\_group.add(item\_box)

elif (tile==19):

item\_box=Itembox('Grenade',x\*TILE\_SIZE,y\*TILE\_SIZE)

item\_box\_group.add(item\_box)

elif (tile==23):

exit\_box=exit(img,x\*TILE\_SIZE,y\*TILE\_SIZE)

exit\_box\_group.add(exit\_box)

return player

def draw(self):

for tile in self.obstacle\_list:

tile[1][0]+=screen\_scroll

screen.blit(tile[0],tile[1])

class Decorations(pygame.sprite.Sprite):

def \_\_init\_\_(self,img,x,y):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.image=img

self.rect=self.image.get\_rect()

self.rect.midtop=(x+TILE\_SIZE//2,y+(TILE\_SIZE-self.image.get\_height()))

def update(self):

self.rect.x+=screen\_scroll

class exit(pygame.sprite.Sprite):

def \_\_init\_\_(self,img,x,y):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.image=img

self.rect=self.image.get\_rect()

self.rect.midtop=(x+TILE\_SIZE//2,y+(TILE\_SIZE-self.image.get\_height()))

def update(self):

self.rect.x+=screen\_scroll

class Water(pygame.sprite.Sprite):

def \_\_init\_\_(self,img,x,y):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.image=img

self.rect=self.image.get\_rect()

self.rect.center=(x+TILE\_SIZE//2,y+TILE\_SIZE//2)

def update(self):

self.rect.x+=screen\_scroll

class Itembox(pygame.sprite.Sprite):

"""docstring for Bullet"""

def \_\_init\_\_(self,item\_type, x, y):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.item\_type=item\_type

self.image=item\_boxes[self.item\_type]

self.rect=self.image.get\_rect()

self.rect.midtop=(x+(TILE\_SIZE//2),y+(TILE\_SIZE-self.image.get\_height()))

def update(self):

self.rect.x+=screen\_scroll

if pygame.sprite.collide\_rect(self,player):

if self.item\_type=='Health':

player.health+=25

if player.health>player.max\_health:

player.health=player.max\_health

pass

elif self.item\_type=='Bullet':

player.ammo+=15

elif self.item\_type=='Grenade':

player.grenades+=3

elif self.item\_type=='Coin':

player.score+=5

self.kill()

class Bullet(pygame.sprite.Sprite):

"""docstring for Bullet"""

def \_\_init\_\_(self, x, y, direction):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.speed = 10

self.image=bullet\_image

self.rect=self.image.get\_rect()

self.rect.center=(x,y)

self.direction=direction

def update(self):

self.rect.x+=(self.direction\*self.speed)+screen\_scroll

if self.rect.right<0 or self.rect.left>SCREEN\_WIDTH:

self.kill()

for tile in world.obstacle\_list:

if tile[1].colliderect(self.rect):

self.kill()

pass

if pygame.sprite.spritecollide(player, bullet\_group,False):

if player.alive:

player.health-=5

self.kill()

for enemy in enemy\_group:

if pygame.sprite.spritecollide(enemy, bullet\_group, False):

if enemy.alive:

enemy.health-=25

self.kill()

class Grenade(pygame.sprite.Sprite):

"""docstring for Bullet"""

def \_\_init\_\_(self, x, y, direction):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.timer=100

self.vel\_y=-11

self.speed = 5

self.image=grenade\_image

self.rect=self.image.get\_rect()

self.rect.center=(x,y)

self.direction=direction

self.width=self.image.get\_width()

self.height=self.image.get\_height()

def update(self):

self.vel\_y+=gravity

dx=self.speed\*self.direction

dy=self.vel\_y

for tile in world.obstacle\_list:

if tile[1].colliderect(self.rect.x+dx,self.rect.y,self.width,self.height):

self.direction\*=-1

dx=self.direction\*self.speed

if tile[1].colliderect(self.rect.x,self.rect.y+dy,self.width,self.height):

if self.vel\_y<0:

self.vel\_y=0

dy=tile[1].bottom-self.rect.top

elif self.vel\_y>=0:

self.vel\_y=0

dy=tile[1].top-self.rect.bottom

self.speed=0

self.rect.x+=dx+screen\_scroll

self.rect.y+=dy

self.timer-=1

if self.timer<=0:

self.kill()

explosion=Explosion(self.rect.x,self.rect.y-20,1)

explosion\_group.add(explosion)

if abs(self.rect.x-player.rect.centerx)<TILE\_SIZE\*2 and \

abs(self.rect.y-player.rect.centery)<TILE\_SIZE\* 2 :

player.health-=50

for enemy in enemy\_group:

if abs(self.rect.x-enemy.rect.centerx)<TILE\_SIZE\*2 and \

abs(self.rect.y-enemy.rect.centery)<TILE\_SIZE\* 2 :

enemy.health-=50

class Explosion(pygame.sprite.Sprite):

"""docstring for Bullet"""

def \_\_init\_\_(self, x, y,scale):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.images=[]

for num in range(0,23):

img=pygame.image.load(f'img/explosion/exp{num}.png').convert\_alpha()

img=pygame.transform.scale(img,(int(img.get\_width()\*scale),int(img.get\_height()\*scale)))

self.images.append(img)

pass

self.frame\_index=0

self.image=self.images[self.frame\_index]

self.rect=self.image.get\_rect()

self.rect.center=(x,y)

self.counter=0

def update(self):

self.rect.x+=screen\_scroll

EXPLOSION\_SPEED=4

self.counter+=1

if self.counter>=EXPLOSION\_SPEED:

self.counter=0

self.frame\_index+=1

if self.frame\_index>=len(self.images):

self.kill()

else:

self.image=self.images[self.frame\_index]

#create sprite groups

enemy\_group=pygame.sprite.Group()

bullet\_group=pygame.sprite.Group()

grenade\_group=pygame.sprite.Group()

explosion\_group=pygame.sprite.Group()

item\_box\_group=pygame.sprite.Group()

decorations\_group=pygame.sprite.Group()

water\_group=pygame.sprite.Group()

exit\_box\_group=pygame.sprite.Group()

#create buttons

nextlevel=button.Button(100,520,nextlevel\_img,1)

restart\_button=button.Button(250,320,restart\_img,1)

mainmenu\_button=button.Button(250,450,mainmenu\_img,1)

login\_button=button.Button(250,300,login\_img,1)

guest\_button=button.Button(250,450,guest\_img,1)

mainmenu=button.Button(400,520,mainmenu\_img,1)

#create empty list

world\_data=[]

for row in range(ROWS):

r=[-1]\*COLS

world\_data.append(r)

pass

record=[]

timer=time.time()

run=True

while run:

clock.tick(FPS)

if login\_check==False:

screen.blit(bg\_img,(0,0))

if login\_button.draw(screen):

login\_check,run,record=login.loop(screen)

elif guest\_button.draw(screen):

uid,name,record=guest.create\_guest()

pygame.time.delay(200)

login\_check=True

elif login\_check==True:

for rec in record:

uid=rec[0]

uname=rec[1]

level\_completed=rec[26]

if start\_game==False:

run,start\_game,level,login\_check=level\_selector.draw(screen,level\_completed,record)

with open(f'level{level}\_data.csv',newline='') as csvfile:

reader=csv.reader(csvfile,delimiter=',')

for x, row in enumerate(reader):

for y, tile in enumerate(row):

world\_data[x][y]=int(tile)

world=World()

player=world.process\_data(world\_data)

else:

draw\_bg()

world.draw()

player.draw()

draw\_bars()

player.update()

for enemy in enemy\_group:

enemy.ai()

enemy.update()

enemy.draw()

#update and draw groups

bullet\_group.update()

grenade\_group.update()

explosion\_group.update()

item\_box\_group.update()

exit\_box\_group.update()

decorations\_group.update()

water\_group.update()

bullet\_group.draw(screen)

grenade\_group.draw(screen)

explosion\_group.draw(screen)

item\_box\_group.draw(screen)

exit\_box\_group.draw(screen)

decorations\_group.draw(screen)

water\_group.draw(screen)

#updating player action

if player.alive:

#shoot bullets

if shoot:

player.shoot()

elif grenade and grenade\_thrown==False and player.grenades>0:

grenade=Grenade(player.rect.centerx+(0.5\*player.rect.size[0]\*player.direction),player.rect.centery-9,player.direction)

grenade\_group.add(grenade)

grenade\_thrown=True

player.grenades-=1

if player.in\_air:

player.update\_action(2)

elif moving\_left or moving\_right:

player.update\_action(1)

else:

player.update\_action(0)

screen\_scroll,level\_complete=player.move(moving\_left, moving\_right)

bg\_scroll-=screen\_scroll

if level\_complete:

print(player.elapsed\_time)

if level>=level\_completed:

level\_completed+=1

print(level)

screen.blit(box\_img,(50,10))

conn=mysql.connector.connect(host="localhost",port=3306,user="root",password="",database="game")

cursor=conn.cursor()

cursor.execute("update users set %s\_score=%s,%s\_time=%s,Levelcompleted=%s where Id=%s",(level+1,player.score,level+1,player.elapsed\_time,level\_completed,uid))

conn.commit()

cursor.execute(f"select (1\_score+2\_score+3\_score+4\_score+5\_score+6\_score+7\_score+8\_score+9\_score),(1\_time+2\_time+3\_time+4\_time+5\_time+6\_time+7\_time+8\_time+9\_time) from users where Id={uid}")

record=cursor.fetchall()

for rec in record:

tot\_score=rec[0]

tot\_time=rec[1]

label.draw(f'{tot\_score}',(0,0,0),270,140,screen,28,'Joker Man')

cursor.execute("update users set Totalscore=%s,Totaltime=%s where Id=%s",(tot\_score,tot\_time,uid))

conn.commit()

cursor.execute("select Id,Totalscore from users order by Totalscore desc");

record=cursor.fetchall()

count=0

for rec in record:

count+=1

if rec[0]==uid:

rank=count

pass

label.draw(f'{rank}',(0,0,0),570,140,screen,28,'Joker Man')

cursor.execute(f"select \* from users where Id={uid}")

record=cursor.fetchall()

if nextlevel.draw(screen):

level+=1

bg\_scroll=cursor.execute("select Name,Totalscore from users order by Totalscore desc limit 3")

record=cursor.fetchall()

i=0

for rec in record:

label.draw(f'{rec[0]}',(0,0,0),250,270+i,screen,28,'Joker Man')

label.draw(f'{rec[1]}',(0,0,0),550,270+i,screen,28,'Joker Man')

i+=70

pass

world\_data=reset\_level()

with open(f'level{level}\_data.csv',newline='') as csvfile:

reader=csv.reader(csvfile,delimiter=',')

for x, row in enumerate(reader):

for y, tile in enumerate(row):

world\_data[x][y]=int(tile)

world=World()

player=world.process\_data(world\_data)

if mainmenu.draw(screen):

run,start\_game,level=level\_selector.draw(screen,level\_completed,record)

bg\_scroll=0

world\_data=reset\_level()

with open(f'level{level}\_data.csv',newline='') as csvfile:

reader=csv.reader(csvfile,delimiter=',')

for x, row in enumerate(reader):

for y, tile in enumerate(row):

world\_data[x][y]=int(tile)

world=World()

player=world.process\_data(world\_data)

pass

else:

screen\_scroll=0

screen.blit(box1\_img,(100,100))

if restart\_button.draw(screen):

bg\_scroll=0

world\_data=reset\_level()

with open(f'level{level}\_data.csv',newline='') as csvfile:

reader=csv.reader(csvfile,delimiter=',')

for x, row in enumerate(reader):

for y, tile in enumerate(row):

world\_data[x][y]=int(tile)

world=World()

player=world.process\_data(world\_data)

if mainmenu\_button.draw(screen):

run,start\_game,level=level\_selector.draw(screen,level\_completed,record)

bg\_scroll=0

world\_data=reset\_level()

with open(f'level{level}\_data.csv',newline='') as csvfile:

reader=csv.reader(csvfile,delimiter=',')

for x, row in enumerate(reader):

for y, tile in enumerate(row):

world\_data[x][y]=int(tile)

world=World()

player=world.process\_data(world\_data)

pass

for event in pygame.event.get():

if event.type == pygame.QUIT:

run=False

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_a:

moving\_left=True

if event.key == pygame.K\_SPACE:

shoot=True

if event.key == pygame.K\_q:

grenade=True

if event.key == pygame.K\_d:

moving\_right=True

if event.key == pygame.K\_w and player.alive:

player.jump=True

if event.type == pygame.KEYUP:

if event.key == pygame.K\_a:

moving\_left=False

if event.key == pygame.K\_d:

moving\_right=False

if event.key == pygame.K\_q:

grenade=False

grenade\_thrown=False

if event.key == pygame.K\_SPACE:

shoot=False

pygame.display.update()

pygame.quit()

**Background Editor page**

import pygame

import button

import pickle

import csv

pygame.init()

SCREEN\_WIDTH=800

SCREEN\_HEIGHT=640

LOWER\_MARGIN=100

SIDE\_MARGIN=300

screen=pygame.display.set\_mode((SCREEN\_WIDTH+SIDE\_MARGIN,SCREEN\_HEIGHT+LOWER\_MARGIN))

pygame.display.set\_caption('Level Editor')

#define game variables

scroll\_left=False

scroll\_right=False

scroll=0

scroll\_speed=3

TILE\_TYPE=24

level=0

current\_tile=0

ROWS=16

MAX\_COLS=200

TILE\_SIZE= SCREEN\_HEIGHT//ROWS

#load images

back\_img=pygame.image.load('img/background/background.jfif').convert\_alpha()

back\_img=pygame.transform.scale(back\_img,(800,640))

save\_img=pygame.image.load('img/background/save.png').convert\_alpha()

save\_img=pygame.transform.scale(save\_img,(150,60))

load\_img=pygame.image.load('img/background/load.png').convert\_alpha()

load\_img=pygame.transform.scale(load\_img,(250,130))

img\_list=[]

for x in range(TILE\_TYPE):

img=pygame.image.load(f'img/background/{x}.png').convert\_alpha()

img=pygame.transform.scale(img,(TILE\_SIZE,TILE\_SIZE))

img\_list.append(img)

pass

#define color

red=(255,0,0)

green=(144,201,120)

WHITE= (255,255,255)

font=pygame.font.SysFont('Futura',30)

#create empty list

world\_data=[]

for row in range(ROWS):

r=[-1]\*MAX\_COLS

world\_data.append(r)

pass

for tile in range(0, MAX\_COLS):

world\_data[ROWS-1][tile]=3

world\_data[ROWS-2][tile]=3

world\_data[ROWS-3][tile]=3

pass

def draw\_text(text,font,col,x,y):

img=font.render(text,True,col)

screen.blit(img,(x,y))

#creating background

def draw\_bg():

screen.fill(green)

width=back\_img.get\_width()

for x in range(10):

screen.blit(back\_img,((x\*width)-scroll,0))

pass

def grid():

#vertical lines

for c in range(MAX\_COLS + 1 ):

pygame.draw.line(screen,WHITE,(c\*TILE\_SIZE - scroll,0),(c\*TILE\_SIZE - scroll,SCREEN\_HEIGHT))

pass

#horizontal lines

for c in range(ROWS + 1 ):

pygame.draw.line(screen,WHITE,(0,c\*TILE\_SIZE),(SCREEN\_WIDTH, c\*TILE\_SIZE))

pass

def draw\_world():

for y,row in enumerate(world\_data):

for x,tile in enumerate(row):

if tile>=0 :

screen.blit(img\_list[tile],(x\*TILE\_SIZE-scroll,y\*TILE\_SIZE))

pass

#create buttons

save\_button=button.Button(150,650,save\_img,1)

load\_button=button.Button(500,650,load\_img,1)

button\_list=[]

button\_col=0

button\_row=0

for i in range(len(img\_list)):

tile\_button = button.Button(SCREEN\_WIDTH + (75 \* button\_col) + 50 , (75 \* button\_row + 50), img\_list[i], 1)

button\_list.append(tile\_button)

button\_col+=1

if button\_col==3:

button\_col=0

button\_row+=1

pass

pass

run=True;

button\_count = 0

while run:

draw\_bg()

grid()

draw\_world()

draw\_text(f'Level: {level}',font,WHITE,10,SCREEN\_HEIGHT+LOWER\_MARGIN-50)

draw\_text(f'Press up and down to change level',font,WHITE,10,SCREEN\_HEIGHT+LOWER\_MARGIN-30)

abc=pygame.draw.rect(screen,green,(SCREEN\_WIDTH,0,SIDE\_MARGIN,SCREEN\_HEIGHT))

#choose tile

for button\_count, i in enumerate(button\_list):

if i.draw(screen):

current\_tile = button\_count

pygame.draw.rect(screen, red, button\_list[current\_tile].rect, 3)

if save\_button.draw(screen):

with open(f'level{level}\_data.csv','w',newline='') as csvfile:

writer=csv.writer(csvfile,delimiter=',')

for row in world\_data:

writer.writerow(row)

if load\_button.draw(screen):

scroll=0

with open(f'level{level}\_data.csv',newline='') as csvfile:

reader=csv.reader(csvfile,delimiter=',')

for x, row in enumerate(reader):

for y, tile in enumerate(row):

world\_data[x][y]=int(tile)

if scroll\_left==True and scroll>0:

scroll-=5 \* scroll\_speed

if scroll\_right==True and scroll<(MAX\_COLS\*TILE\_SIZE)-SCREEN\_WIDTH:

scroll+=5 \* scroll\_speed

#get mouse position

pos=pygame.mouse.get\_pos()

x=(pos[0]+scroll)//TILE\_SIZE

y= pos[1] // TILE\_SIZE

if pos[0]<SCREEN\_WIDTH and pos[1]<SCREEN\_HEIGHT:

if pygame.mouse.get\_pressed()[0]==1:

if world\_data[y][x]!= current\_tile:

world\_data[y][x]=current\_tile

pass

if pygame.mouse.get\_pressed()[2] == 1:

world\_data[y][x] = -1

for event in pygame.event.get():

if event.type==pygame.QUIT:

run=False

if event.type==pygame.KEYDOWN:

if event.key==pygame.K\_UP:

level+=1

if event.key==pygame.K\_DOWN and level>0:

level-=1

if event.key==pygame.K\_RIGHT:

scroll\_right=True

if event.key==pygame.K\_LEFT:

scroll\_left=True

if event.type==pygame.KEYUP:

if event.key==pygame.K\_LEFT:

scroll\_left=False

if event.key==pygame.K\_RIGHT:

scroll\_right=False

pygame.display.update()

pygame.quit()

**login.py**

import pygame

import button

import textbox

import label

import register

import mysql.connector

bg\_img=pygame.image.load('img/extra/background2.jpg')

bg\_img=pygame.transform.scale(bg\_img,(800,int(800\*0.8)))

login\_img=pygame.image.load('img/button/login.png')

login\_img=pygame.transform.scale(login\_img,(150,80))

back\_img=pygame.image.load('img/button/back.png')

back\_img=pygame.transform.scale(back\_img,(50,50))

register\_img=pygame.image.load('img/button/register.png')

register\_img=pygame.transform.scale(register\_img,(150,65))

box\_img=pygame.image.load('img/extra/box.png')

button1=button.Button(200,420,login\_img,1)

button2=button.Button(350,425,register\_img,1)

back=button.Button(170,170,back\_img,1)

BG= (144,201,120)

def draw\_bg(screen):

screen.blit(bg\_img,(0,0))

screen.blit(box\_img,(100,100))

pass

def checkdata(username,password,screen):

record=[]

uid=0

uname=0

levelcompleted=0

if username=='':

msg1="Email is empty"

return False,record,msg1

pass

else:

conn=mysql.connector.connect(host="localhost",port=3306,user="root",password="",database="game")

cursor=conn.cursor()

cursor.execute("select \* from users where Email=%s AND Password=%s",(username,password))

record=cursor.fetchall()

if record:

msg2="Login Sucessfull"

return True,record,msg2

else:

msg3="Email & password does not match"

return False,record,msg3

cursor.close()

conn.close()

def loop(screen):

login=True

username=textbox.Textbox(screen,220,270,280,40)

password=textbox.Textbox(screen,220,360,280,40)

check=True

record=[]

while login:

draw\_bg(screen)

username.draw('simple')

password.draw('password')

label.draw('Login',(255,255,255),250,150,screen,60,'Arial Black')

label.draw('Email :',(255,255,255),170,230,screen,36,'Cambria')

label.draw('Password: ',(255,255,255),170,320,screen,36,'Cambria')

if check==False:

label.draw(msg,(255,0,0),190,480,screen,18,'Cambria')

pass

pos=pygame.mouse.get\_pos()

username.update(pos)

password.update(pos)

if button1.draw(screen):

uname=username.gettext()

passw=password.gettext()

checklogin,record,msg=checkdata(uname,passw,screen)

if checklogin:

pygame.time.delay(500)

check=True

return True,True,record

else:

check=False

if button2.draw(screen):

register.loop(screen)

if back.draw(screen):

return False,True,record

for event in pygame.event.get():

if event.type==pygame.QUIT:

return False,False,record

if event.type==pygame.KEYDOWN:

username.user\_input(event)

password.user\_input(event)

pygame.display.update()

pass

**register.py**

import pygame

import button

import textbox

import label

import mysql.connector

bg\_img=pygame.image.load('img/extra/background2.jpg')

bg\_img=pygame.transform.scale(bg\_img,(800,int(800\*0.8)))

register\_img=pygame.image.load('img/button/register.png')

register\_img=pygame.transform.scale(register\_img,(250,80))

box\_img=pygame.image.load('img/extra/box.png')

box\_img=pygame.transform.scale(box\_img,(700,630))

button2=button.Button(260,450,register\_img,1)

back\_img=pygame.image.load('img/button/back.png')

back\_img=pygame.transform.scale(back\_img,(50,50))

back=button.Button(150,90,back\_img,1)

BG= (144,201,120)

def draw\_bg(screen):

screen.blit(bg\_img,(0,0))

screen.blit(box\_img,(50,-5))

pass

def register(uname,email\_id,country\_val,gender\_val,cpassword\_val,password\_val):

conn=mysql.connector.connect(host="localhost",port=3306,user="root",password="",database="game")

cursor=conn.cursor()

if email\_id=='':

msg="Email is compulsary"

return msg,False

elif uname=='':

msg="User name is compulsary"

return msg,False

elif password\_val!=cpassword\_val or cpassword\_val=='' or password\_val=='':

msg="Password doesn't match"

return msg,False

pass

else:

cursor.execute("select Email from users")

record=cursor.fetchall()

for rec in record:

if rec[0]==email\_id:

msg="Email already exists"

return msg,False

cursor.execute("insert into users (Name,Email,Country,Gender,Password) values(%s,%s,%s,%s,%s)",(uname,email\_id,country\_val,gender\_val,password\_val))

conn.commit()

cursor.close()

conn.close()

msg='Register sucessfully'

return msg,True

def loop(screen):

login=True

check=True

username=textbox.Textbox(screen,150,200,200,40)

email=textbox.Textbox(screen,400,200,200,40)

country=textbox.Textbox(screen,150,300,200,40)

gender=textbox.Textbox(screen,400,300,200,40)

password=textbox.Textbox(screen,150,400,200,40)

cpassword=textbox.Textbox(screen,400,400,200,40)

while login:

draw\_bg(screen)

username.draw('simple')

email.draw('simple')

country.draw('simple')

gender.draw('simple')

cpassword.draw('password')

password.draw('password')

label.draw('Sign Up',(255,255,255),250,70,screen,60,'Arial Black')

label.draw('Email\* :',(255,255,255),400,150,screen,28,'Cambria')

label.draw('User Name\* :',(255,255,255),150,150,screen,28,'Cambria')

label.draw('Country :',(255,255,255),400,250,screen,28,'Cambria')

label.draw('Gender :',(255,255,255),150,250,screen,28,'Cambria')

label.draw('Password\* :',(255,255,255),150,350,screen,28,'Cambria')

label.draw('Confirm Password\* : ',(255,255,255),400,350,screen,28,'Cambria')

if check==False:

label.draw(msg,(0,0,255),300,550,screen,28,'Cambria')

if back.draw(screen):

login=False

pos=pygame.mouse.get\_pos()

username.update(pos)

uname=username.gettext()

email.update(pos)

email\_id=email.gettext()

country.update(pos)

country\_val=country.gettext()

gender.update(pos)

gender\_val=gender.gettext()

cpassword.update(pos)

cpassword\_val=cpassword.gettext()

password.update(pos)

password\_val=password.gettext()

if button2.draw(screen):

msg,check=register(uname,email\_id,country\_val,gender\_val,

cpassword\_val,password\_val)

if check==True:

login=False

pass

for event in pygame.event.get():

if event.type==pygame.QUIT:

pass

if event.type==pygame.KEYDOWN:

username.user\_input(event)

email.user\_input(event)

country.user\_input(event)

gender.user\_input(event)

cpassword.user\_input(event)

password.user\_input(event)

pygame.display.update()

**level\_selector.py**

import pygame

import label

import button

import help\_page

import highscore\_page

pygame.init()

SCREEN\_WIDTH=800

SCREEN\_HEIGHT=int(SCREEN\_WIDTH\*0.8)

img\_list=[]

active\_img\_list=[]

inactive\_img\_list=[]

#bg\_img=pygame.image.load('img/extra/box.png')

#bg\_img1=pygame.image.load('img/extra/box1.png')

bg\_img=pygame.image.load('img/extra/background4.jpg')

bg\_img=pygame.transform.scale(bg\_img,(SCREEN\_WIDTH,SCREEN\_HEIGHT))

play\_img=pygame.image.load('img/button/play.png')

help\_img=pygame.image.load('img/button/help.png')

logout\_img=pygame.image.load('img/button/logout.png')

highscore\_img=pygame.image.load('img/button/highscore.png')

for x in range(9):

img=pygame.image.load('img/button/inactive\_level.png')

img=pygame.transform.scale(img,(80,80))

inactive\_img\_list.append(img)

pass

for x in range(9):

img=pygame.image.load(f'img/button/{x}.png')

img=pygame.transform.scale(img,(80,80))

active\_img\_list.append(img)

pass

abutton\_list=[]

def draw(screen,level\_completed,record):

level\_completed=level\_completed

button\_list=[]

button\_col=0

button\_row=0

abutton\_col=0

abutton\_row=0

current\_level=0

run=True

for i in range(level\_completed+1):

tile\_button = button.Button((100 \* button\_col) + 450 , (150 \* button\_row) + 220, active\_img\_list[i], 1)

button\_list.append(tile\_button)

button\_col+=1

if button\_col==3:

button\_col=0

button\_row+=1

play\_button=button.Button(50 , 500, play\_img, 1)

help\_button=button.Button(50 , 370, help\_img, 1)

highscore\_button=button.Button(50 , 220, highscore\_img, 1)

logout\_button=button.Button(50 , 70, logout\_img, 1)

while run:

screen.blit(bg\_img,(0,0))

img\_col=3

img\_row=3

button\_col=0

button\_row=0

for x in range(len(inactive\_img\_list)-level\_completed-1):

screen.blit(inactive\_img\_list[x],((100 \* img\_col) +350 , (130 \* img\_row )+100))

img\_col-=1

if img\_col==0:

img\_col=3

img\_row-=1

pass

for rec in record:

for x in range(8,level\_completed+9):

label.draw(f'{rec[x]}',(255,255,255),(100 \* button\_col) + 470 , (140 \* button\_row) + 310,screen,34,'Copper')

label.draw(f'{rec[x+9]} s',(255,255,255),(100 \* button\_col) + 460 , (150 \* button\_row) + 200,screen,24,'Copper')

button\_col+=1

if button\_col==3:

button\_col=0

button\_row+=1

label.draw(f'{rec[1]}',(255,255,255),70,25,screen,36,'Joker Man')

label.draw(f'{rec[6]}',(255,255,255),730,130,screen,24,'Copper')

label.draw(f'{rec[7]} s',(255,255,255),730,170,screen,24,'Copper')

for button\_count, i in enumerate(button\_list):

if i.draw(screen):

current\_level=button\_count

pygame.draw.rect(screen, (255,255,255), button\_list[current\_level].rect, 3)

if play\_button.draw(screen):

return True,True,current\_level,True

pass

if help\_button.draw(screen):

help\_page.loop(screen)

pass

if highscore\_button.draw(screen):

highscore\_page.loop(screen)

if logout\_button.draw(screen):

return True,False,current\_level,False

for event in pygame.event.get():

if event.type==pygame.QUIT:

return False,False,current\_level,False

pygame.display.update()

**guest.py**

import pygame

import mysql.connector

def create\_guest():

conn=mysql.connector.connect(host="localhost",port=3306,user="root",password="",database="game")

cursor=conn.cursor()

cursor.execute("insert into users (Name,Email) values('','')")

conn.commit()

cursor.execute("select \* from users where Email=''")

record=cursor.fetchall()

for rec in record:

gid=rec[0]

guest\_name="Guest"+f'{gid}'

cursor.execute("update users set Name=%s,Email=%s where id=%s",(guest\_name,guest\_name,gid))

conn.commit()

cursor.close()

conn.close()

gid,name,record=get\_data(guest\_name)

return gid,name,record

def get\_data(guest\_name):

guest=guest\_name

conn1=mysql.connector.connect(host="localhost",port=3306,user="root",password="",database="game")

cursor1=conn1.cursor()

cursor1.execute(f"select \* from users where Email='{guest}'")

record=cursor1.fetchall()

for rec in record:

gid=rec[0]

name=rec[1]

return gid,name,record

cursor1.close()

conn1.close()

**highscore\_page.py**

import pygame

import button

import mysql.connector

import label

bg\_img=pygame.image.load('img/extra/background2.jpg')

bg\_img=pygame.transform.scale(bg\_img,(800,int(800\*0.8)))

box\_img=pygame.image.load('img/extra/scoreboard.png')

box\_img=pygame.transform.scale(box\_img,(600,500))

back\_img=pygame.image.load('img/button/back.png')

back\_img=pygame.transform.scale(back\_img,(50,50))

back=button.Button(170,130,back\_img,1)

def loop(screen):

loop=True

while loop:

screen.blit(bg\_img,(0,0))

screen.blit(box\_img,(100,100))

conn=mysql.connector.connect(host="localhost",port=3306,user="root",password="",database="game")

cursor=conn.cursor()

cursor.execute("select Name,Totalscore from users order by Totalscore desc limit 6")

record=cursor.fetchall()

i=0

for rec in record:

label.draw(f'{rec[0]}',(0,0,0),260,210+i,screen,28,'Joker Man')

label.draw(f'{rec[1]}',(0,0,0),520,210+i,screen,28,'Joker Man')

i+=60

pass

if back.draw(screen):

loop=False

pygame.time.delay(200)

pass

for event in pygame.event.get():

if event.type==pygame.QUIT:

loop=False

pygame.display.update()

pass

**help\_page.py**

import pygame

import button

bg\_img=pygame.image.load('img/extra/background2.jpg')

bg\_img=pygame.transform.scale(bg\_img,(800,int(800\*0.8)))

box\_img=pygame.image.load('img/extra/helpboard.png')

box\_img=pygame.transform.scale(box\_img,(600,500))

back\_img=pygame.image.load('img/button/back.png')

back\_img=pygame.transform.scale(back\_img,(50,50))

back=button.Button(170,130,back\_img,1)

def loop(screen):

loop=True

while loop:

screen.blit(bg\_img,(0,0))

screen.blit(box\_img,(100,100))

if back.draw(screen):

loop=False

pygame.time.delay(200)

pass

for event in pygame.event.get():

if event.type==pygame.QUIT:

loop=False

pygame.display.update()

**textbox.py**

import pygame

vec=pygame.math.Vector2

class Textbox:

"""docstring for Textbox"""

def \_\_init\_\_(self, surface,x,y,width,height,state='',color=(176,176,176),active\_color=(255,255,255),border=True,

border\_color=(0,0,0),border\_width=2,text\_name='jokerman',text\_size=20,text\_color=(0,0,0)):

self.x = x

self.y = y

self.pos = vec(x,y)

self.width = width

self.height = height

self.surface=surface

self.image = pygame.Surface((width,height))

self.rect = self.image.get\_rect()

self.rect.topleft = self.pos

self.state = state

self.color = color

self.active\_color = active\_color

self.border = border

self.border\_color = border\_color

self.border\_width = border\_width

self.text = []

self.font\_name = text\_name

self.font\_size = text\_size

self.text\_color=text\_color

self.cursor\_pos=0

self.active=False

self.hovered=False

self.click=False

def update(self,pos):

if self.is\_hovered(pos):

if pygame.mouse.get\_pressed()[0] == 1:

self.click=True

self.hovered=True

self.active=True

else:

if pygame.mouse.get\_pressed()[0] == 1:

self.click=False

self.hovered=False

self.active=False

def draw(self,typ):

if self.border:

if self.click:

self.image.fill((0,0,255))

pygame.draw.rect(self.image,self.active\_color,(self.border\_width,self.border\_width,self.width-(self.border\_width\*2),self.height-(self.border\_width\*2)))

else:

self.image.fill((0,0,0))

pygame.draw.rect(self.image,self.active\_color,(self.border\_width,self.border\_width,self.width-(self.border\_width\*2),self.height-(self.border\_width\*2)))

else:

if self.active:

self.image.fill(self.active\_color)

else:

self.image.fill(self.color)

if len(self.text)>0:

self.show\_text(typ)

self.surface.blit(self.image,self.pos)

def is\_hovered(self,pos):

if pos[0]>self.pos.x and pos[0]<self.pos.x+self.width:

if pos[1]>self.pos.y and pos[1]<self.pos.y+self.height:

return True

return False

def show\_text(self,typ):

text=''.join(self.text)

ptext=''

font=pygame.font.SysFont(self.font\_name,self.font\_size)

if typ=='password':

for x in text:

ptext=ptext+'\*'

text=font.render(ptext,False,self.text\_color)

else:

text=font.render(text,False,self.text\_color)

size=text.get\_size()

if size[0]+10>self.width+10:

x,y=self.width-(size[0]+10),(self.height//2)-(size[1]//2)

else:

x,y=10,(self.height//2)-(size[1]//2)

pos=(x,y)

self.image.blit(text,pos)

def gettext(self):

data=''

for x in range(0,len(self.text)):

data=data+self.text[x]

return data

pass

def user\_input(self,event):

if self.click:

if event.key != 13 and event.key != 273 and event.key != 274 and event.key != 275 and event.key != 276 and event.key != 8 and event.key != 127:

self.text.insert(self.cursor\_pos,event.unicode)

self.cursor\_pos+=1

elif event.key==8 and self.cursor\_pos>0 and len(self.text)>0:

del self.text[self.cursor\_pos-1]

self.cursor\_pos-=1

**label.py**

import pygame

pygame.init()

def draw(text,col,x,y,screen,size,font):

font=pygame.font.SysFont(font,size,True)

img=font.render(text,True,col)

screen.blit(img,(x,y))

**button.py**

import pygame

#button class

class Button():

def \_\_init\_\_(self,x, y, image, scale):

width = image.get\_width()

height = image.get\_height()

self.image = pygame.transform.scale(image, (int(width \* scale), int(height \* scale)))

self.rect = self.image.get\_rect()

self.rect.topleft = (x, y)

self.clicked = False

def draw(self, surface):

action = False

#get mouse position

pos = pygame.mouse.get\_pos()

#check mouseover and clicked conditions

if self.rect.collidepoint(pos):

if pygame.mouse.get\_pressed()[0] == 1 and self.clicked == False:

action = True

self.clicked = True

if pygame.mouse.get\_pressed()[0] == 0:

self.clicked = False

#draw button

surface.blit(self.image, (self.rect.x, self.rect.y))

return action

1. **Testing**

Testing is one of the most critical processes of the Software Development Lifecycle (SDLC). It helps companies to perform a comprehensive assessment of software and ensure that their product fulfils the client’s needs.

The testing phases of the software development lifecycle help companies to identify all the bugs and errors in the software before the implementation phase begins. If software bugs are not resolved before deployment, they can adversely affect the client’s business.

Besides that, trying to resolve these issues at a later stage can incur substantial costs. The more you delay the detection of these issues, the greater the cost you are likely to face.

There are several ways to perform testing in the software development lifecycle. The techniques used for testing can vary depending on the software development model, the stage of the process, and the objectives of the testing procedure.

* 1. **Testing criteria**
* In each form, no field should be left blank.
* An error message will be generated when there will some error in any operation
* All primary keys should automatically generated to prevent user from entering existing values.
  1. **Test cases**
     1. **Login System**

|  |  |
| --- | --- |
| Test Description: | The user leaves the email field blank |
| Test data: | Email Id |
| Expected result: | Error message “Email id is empty” is displayed |
| Output | (Login page) |

|  |  |
| --- | --- |
| Test Description: | The user leaves the password field blank |
| Test data: | Password |
| Expected result: | Error message “Password is empty” is displayed |
| Output | (Login page) |

|  |  |
| --- | --- |
| Test Description: | The user entered incorrect values for email and password |
| Test data: | Email & Password |
| Expected result: | Error message “Email & Password does not match” is displayed |
| Output | (Login page) |

### **Registration System**

|  |  |
| --- | --- |
| Test Description: | The user leaves the email field blank |
| Test data: | Email Id |
| Expected result: | Error message “Email id is compulsary” is displayed |
| Output | (Registration page) |

|  |  |
| --- | --- |
| Test Description: | The user leaves the User name field blank |
| Test data: | User name |
| Expected result: | Error message “User name is compulsary” is displayed |
| Output | (Registration page) |

|  |  |
| --- | --- |
| Test Description: | The user entered different values for password and confirm password |
| Test data: | Password & Confirm password |
| Expected result: | Error message “Email & Password does not match” is displayed |
| Output | (Registration page) |

|  |  |
| --- | --- |
| Test Description: | The user entered email which is already registered |
| Test data: | Email |
| Expected result: | Error message “Email already exists” is displayed |
| Output | (Registration page) |

* + 1. **Game**

|  |  |
| --- | --- |
| Test Description: | The user collides with water or its health becomes 0 |
| Test data: | Player |
| Expected result: | Error message “You died” is displayed |
| Output | (When player dies) |

|  |  |
| --- | --- |
| Test Description: | The user collides with exit door |
| Test data: | Player |
| Expected result: | Error message “Level completed” is displayed |
| Output | (Level Completion page) |

1. **Implementation of project**

At first the main menu screen is displayed, which contain two buttons i.e. Login and Guest. Login button is for those users who want to give her data and play with her name.

And guest option is for those users who doesn’t want to give or store her data in game’s database. So for those users we create a empty row with a unique Username and Email. In guest option, only Name and Email will be filled and everything will remain blank. To generate a unique username and email we use the following code:

def create\_guest():

conn=mysql.connector.connect(host="localhost",port=3306,user="root",password="",database="game")

cursor=conn.cursor()

#first we create an empty row with blank email and name

cursor.execute("insert into users (Name,Email) values(' ',' ')")

conn.commit()

#then I select the row with blank email and get his id

cursor.execute("select \* from users where Email=''")

record=cursor.fetchall()

for rec in record:

gid=rec[0]

guest\_name="Guest"+f'{gid}'

#then name and email is updated by guest name which is Guest + Id

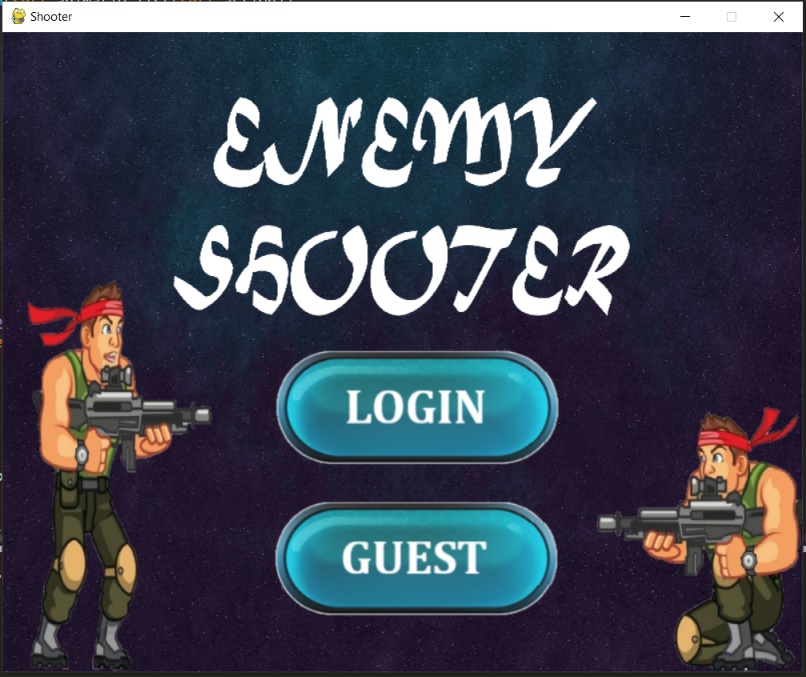
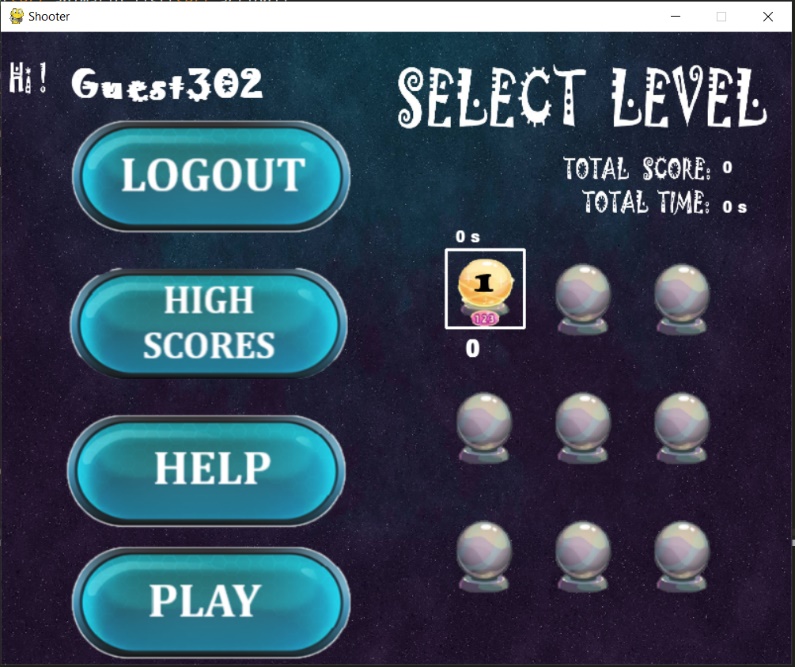
cursor.execute("update users set Name=%s,Email=%s where id=%s",(guest\_name,guest\_name,gid))

conn.commit()

cursor.close()

conn.close()

gid,name,record=get\_data(guest\_name)

 return gid,name,record

(After clicking guest, guest Id will be generated and displayed to level selection page)

In login option, if user is already registered then he/she should enter her email & password and click on login button. But if user is not registered, then he/she has to register first. For registration he has to click on Register button and then fill all her details and click on register. If all the details are correct, then user will be registered successfully and comes back to login screen. The code used for registration will be:

def register(uname,email\_id,country\_val,gender\_val,cpassword\_val,password\_val):

#create connection to database

conn=mysql.connector.connect(host="localhost",port=3306,user="root",password="",database="game")

cursor=conn.cursor()

#check whether email is filled or not

if email\_id==' ':

msg="Email is compulsary"

return msg,False

#check whether user name is filled or not

elif uname==' ':

msg="User name is compulsary"

return msg,False

#check whether password is filled correctly

elif password\_val!=cpassword\_val or cpassword\_val==' ' or password\_val==' ':

msg="Password doesn't match"

return msg,False

pass

#check whether email already exists or not

else:

cursor.execute("select Email from users")

record=cursor.fetchall()

for rec in record:

if rec[0]==email\_id:

msg="Email already exists"

return msg,False

#insert data into the database

cursor.execute("insert into users (Name,Email,Country,Gender,Password) values(%s,%s,%s,%s,%s)",(uname,email\_id,country\_val,gender\_val,password\_val))

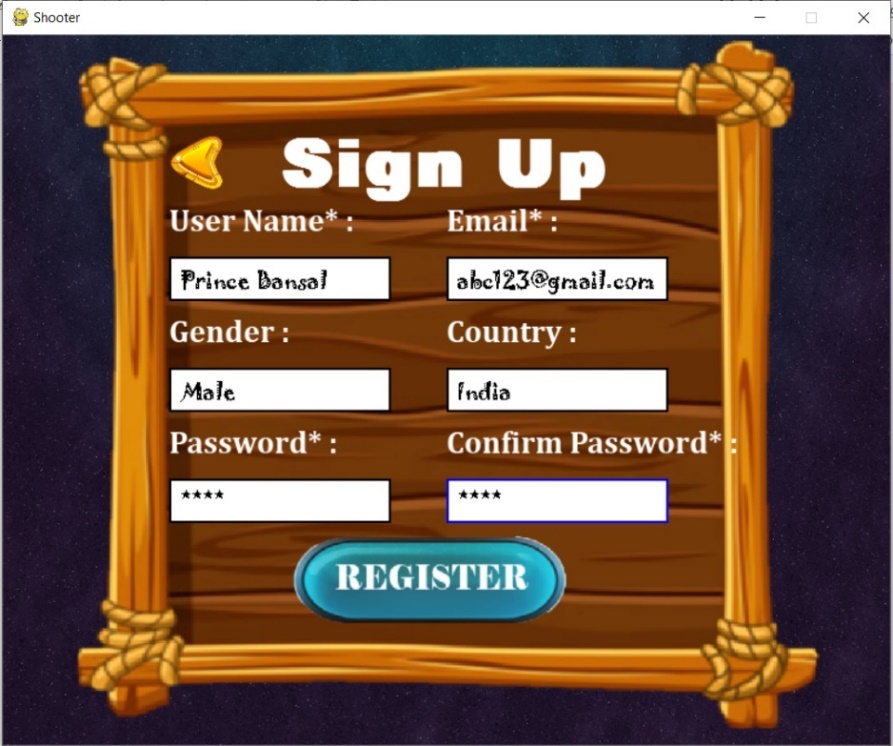
conn.commit()

cursor.close()

conn.close()

msg='Register sucessfully'

return msg,True



After registration user has to login with his email an password. After login, Level Selection page will open and User name, His total score, total time, Score of each completed level, time of each completed level is displayed. Also level selection page has 9 level buttons , of which only “ Numbers of Level Completed +1” buttons are active and others will be inactive. First user has to choose the level among active levels and then click on the play button to start the level. This page also has three more buttons which are Logout ( use to logout), High Scores( use to display top 10 score) & Help (use to display help box) .

After clicking on play button, the game will starts. During execution of The game, various collisions should be checked continuously. The items by which collision should be checked were:

* Item boxes like coins, bullet box, grenade box, health box
* Bullets
* Grenades
* Water
* Exit Gate

Different functioning will occur with collision of different items. Various functioning will be:

* Collision with Item boxes

If collision of player rectangle takes place with Item box rectangle, then the item box will get disappear and function corresponding to each item box takes place. If collision takes place with coins, then score will be increased by 5. And if collision takes place by bullet box, the number of bullets will increased by 15. If collision takes place by grade box, the number of grenades increased by 3. And if collision takes place by Health box, then player’s health will increase by 25%.

Code behind this will be:

class Itembox(pygame.sprite.Sprite):

"""docstring for Itembox"""

def \_\_init\_\_(self,item\_type, x, y):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.item\_type=item\_type

self.image=item\_boxes[self.item\_type]

self.rect=self.image.get\_rect()

self.rect.midtop=(x+(TILE\_SIZE//2),y+(TILE\_SIZE-self.image.get\_height()))

def update(self):

self.rect.x+=screen\_scroll

#check collision with item boxes

if pygame.sprite.collide\_rect(self,player):

if self.item\_type=='Health':

#increase health by 25%

player.health+=25

if player.health>player.max\_health:

player.health=player.max\_health

pass

elif self.item\_type=='Bullet':

#increase bullets by 15

player.ammo+=15

elif self.item\_type=='Grenade':

#increase grenades by 3

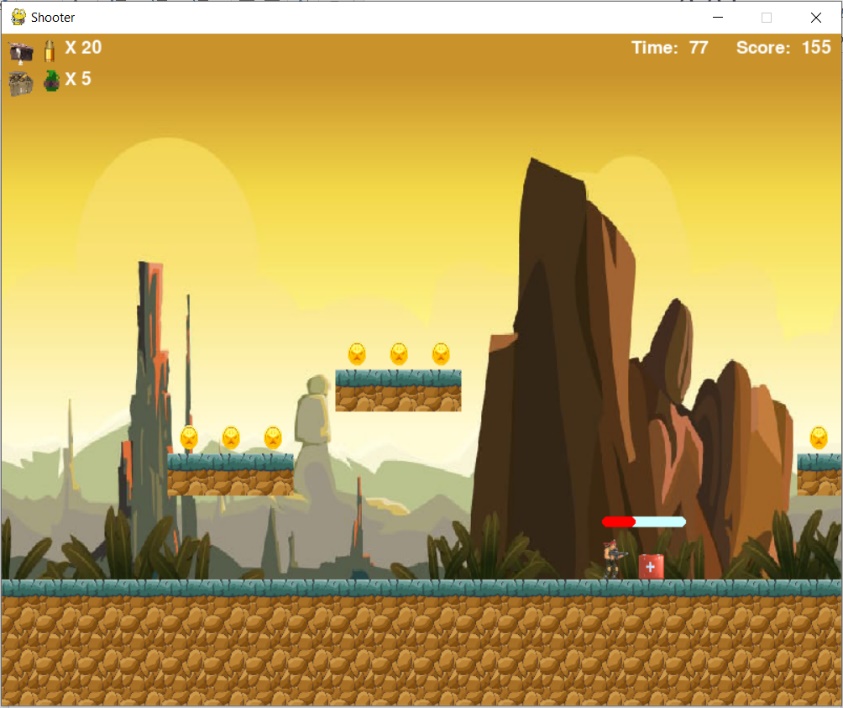
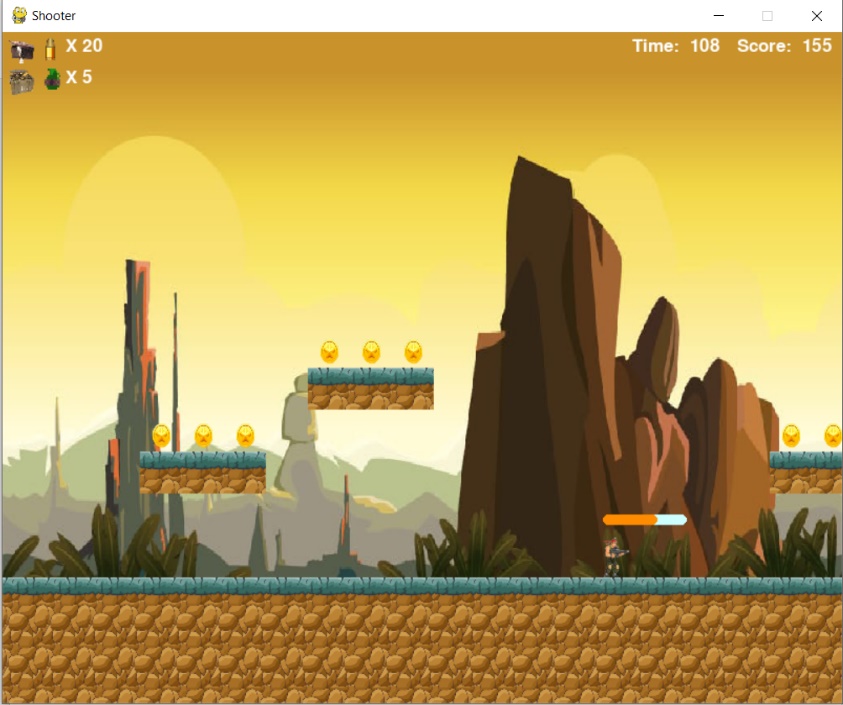
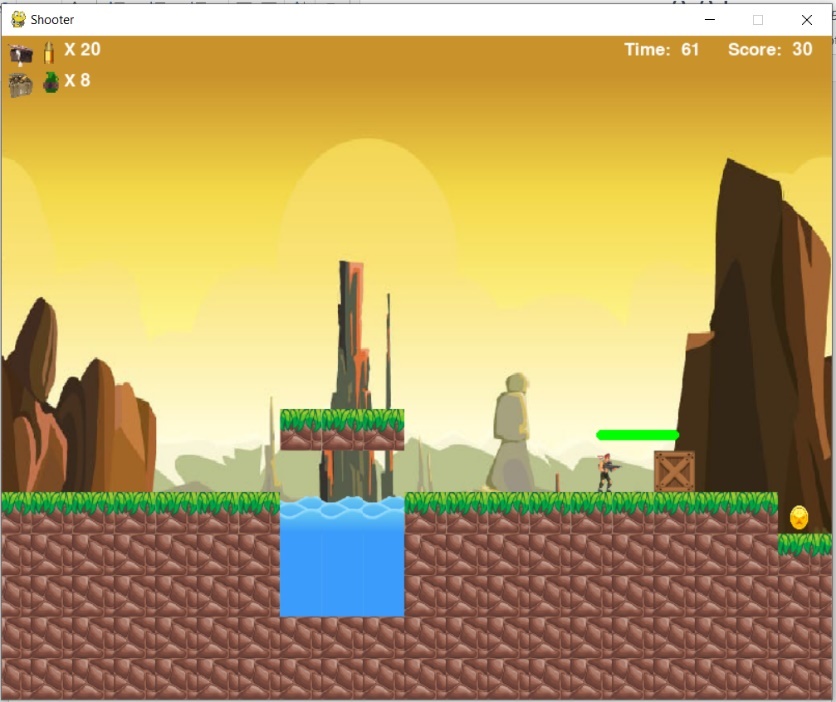
player.grenades+=3

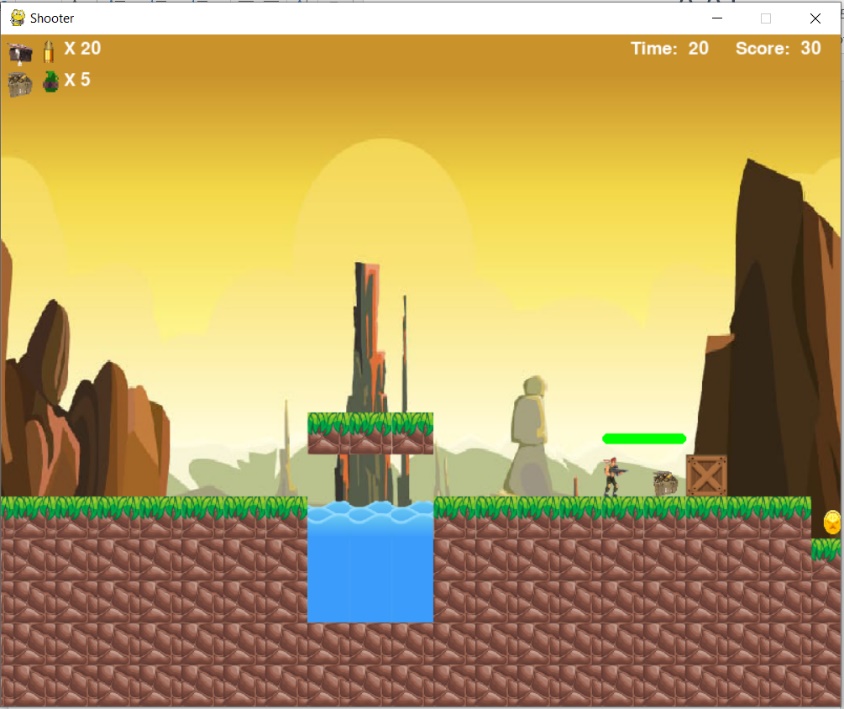
elif self.item\_type=='Coin':

#increase score by 5

player.score+=5

self.kill()

Before Collision After Collision



(Player after collision with item boxes)

(Player before collision with item boxes)

* Collision with bullet

When player’s rectangle collides with bullet , then player’s health will decrease by 5%. And if bullet collides with enemy’s rectangle then enemy’s health will decrease by 25%.

class Bullet(pygame.sprite.Sprite):

"""docstring for Bullet"""

def \_\_init\_\_(self, x, y, direction):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.speed = 10

self.image=bullet\_image

self.rect=self.image.get\_rect()

self.rect.center=(x,y)

self.direction=direction

def update(self):

self.rect.x+=(self.direction\*self.speed)+screen\_scroll

if self.rect.right<0 or self.rect.left>SCREEN\_WIDTH:

self.kill()

for tile in world.obstacle\_list:

if tile[1].colliderect(self.rect):

self.kill()

pass

if pygame.sprite.spritecollide(player, bullet\_group,False):

if player.alive:

player.health-=5

self.kill()

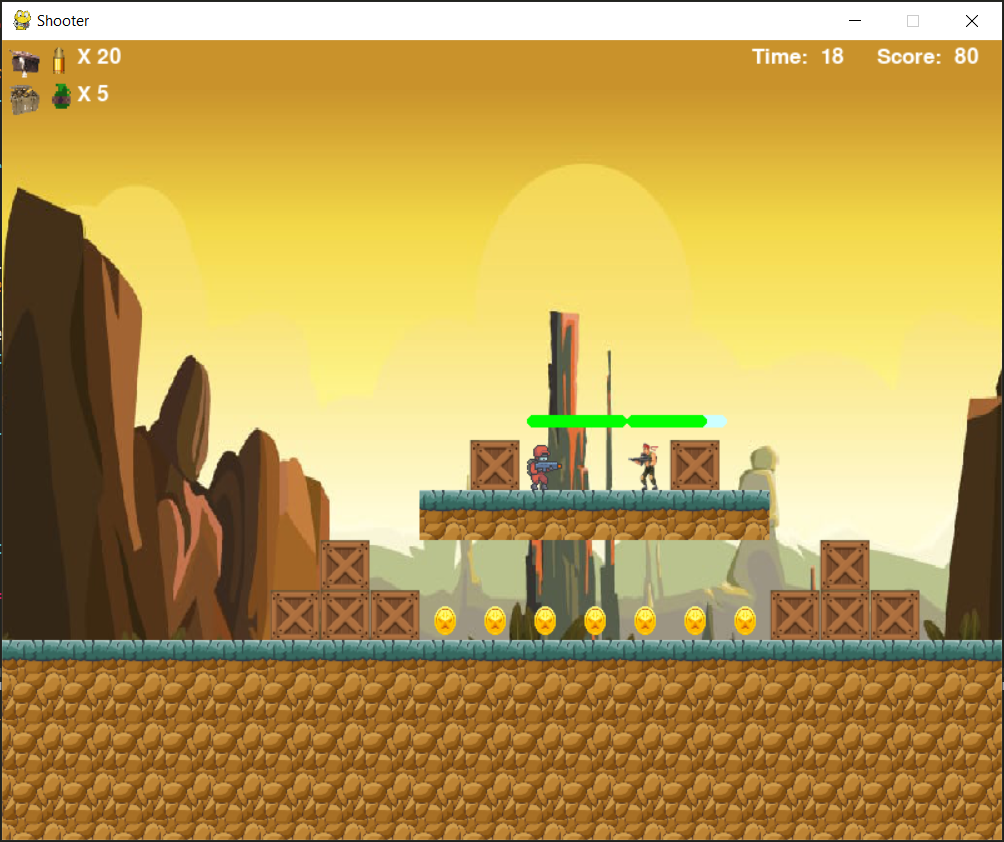
for enemy in enemy\_group:

if pygame.sprite.spritecollide(enemy, bullet\_group, False):

if enemy.alive:

enemy.health-=25

self.kill()





(Health decreases when bullet collides with player)

* Collision with Grenade explosion

When grenade is thrown by the player, it creates explosion when grenade touches the ground and when player’s rectangle collides with the grenade explosion , then player’s health will decrease by 50%. And if explosion collides with enemy’s rectangle then enemy’s health will also decrease by 50%.

class Grenade(pygame.sprite.Sprite):

"""docstring for Grenade"""

def \_\_init\_\_(self, x, y, direction):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.timer=100

self.vel\_y=-11

self.speed = 5

self.image=grenade\_image

self.rect=self.image.get\_rect()

self.rect.center=(x,y)

self.direction=direction

self.width=self.image.get\_width()

self.height=self.image.get\_height()

def update(self):

self.vel\_y+=gravity

dx=self.speed\*self.direction

dy=self.vel\_y

for tile in world.obstacle\_list:

#check collision with ground and boundaries

if tile[1].colliderect(self.rect.x+dx,self.rect.y,self.width,self.height):

self.direction\*=-1

dx=self.direction\*self.speed

if tile[1].colliderect(self.rect.x,self.rect.y+dy,self.width,self.height):

if self.vel\_y<0:

self.vel\_y=0

dy=tile[1].bottom-self.rect.top

elif self.vel\_y>=0:

self.vel\_y=0

dy=tile[1].top-self.rect.bottom

self.speed=0

self.rect.x+=dx+screen\_scroll

self.rect.y+=dy

self.timer-=1

if self.timer<=0:

self.kill()

explosion=Explosion(self.rect.x,self.rect.y-20,1)

explosion\_group.add(explosion)

#check collision with player

if abs(self.rect.x-player.rect.centerx)<TILE\_SIZE\*2 and \

abs(self.rect.y-player.rect.centery)<TILE\_SIZE\* 2 :

player.health-=50

#check collision with enemy

for enemy in enemy\_group:

if abs(self.rect.x-enemy.rect.centerx)<TILE\_SIZE\*2 and \

abs(self.rect.y-enemy.rect.centery)<TILE\_SIZE\* 2 :

enemy.health-=50

* Collision with water

When player’s rectangle with water boxes, then player’s health will become zero . And You died message is displayed. Along with this message two buttons will also displayed i.e. Restart and Main Menu . On clicking Restart button, game will started again and on clicking Main Menu , Level Selection Interface will be displayed.

class Water(pygame.sprite.Sprite):

def \_\_init\_\_(self,img,x,y):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.image=img

self.rect=self.image.get\_rect()

self.rect.center=(x+TILE\_SIZE//2,y+TILE\_SIZE//2)

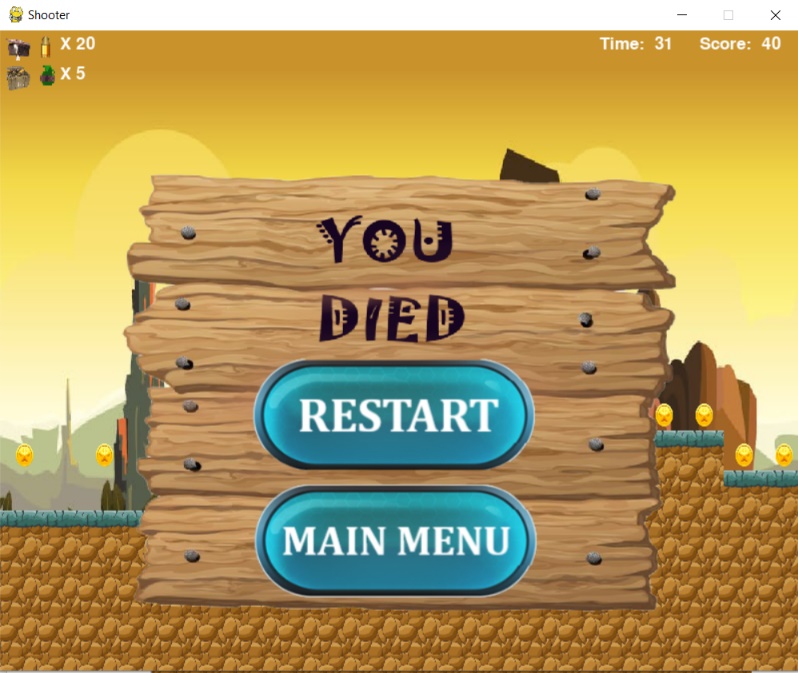
def update(self):

self.rect.x+=screen\_scroll

if pygame.sprite.spritecollide(self,water\_group,False):

player.health=0



Before Collision After Collision

(Player after collision with water or health=0)

(Player before collision with water)

* Collision with exit Gate

When player collides with exit gate, the Level completed interface is displayed to user. Also at level completion player’s level score and time will be updated into the database. Also player’s rank and total score will be calculated . And top 3 scores will be displayed. Code used for all these are:

#check collision with exit gate

if pygame.sprite.spritecollide(self,exit\_box\_group,False):

dx=0

level\_complete=True

if level\_complete:

if level>=level\_completed:

level\_completed+=1

screen.blit(box\_img,(50,10))

conn=mysql.connector.connect(host="localhost",port=3306,user="root",password="",database="game")

cursor=conn.cursor()

cursor.execute("update users set %s\_score=%s,%s\_time=%s,Levelcompleted=%s where Id=%s",(level+1,player.score,level+1,player.elapsed\_time,level\_completed,uid))

conn.commit()

cursor.execute(f"select (1\_score+2\_score+3\_score+4\_score+5\_score+6\_score+7\_score+8\_score+9\_score),(1\_time+2\_time+3\_time+4\_time+5\_time+6\_time+7\_time+8\_time+9\_time) from users where Id={uid}")

record=cursor.fetchall()

for rec in record:

tot\_score=rec[0]

tot\_time=rec[1]

label.draw(f'{tot\_score}',(0,0,0),270,140,screen,28,'Joker Man')

cursor.execute("update users set Totalscore=%s,Totaltime=%s where Id=%s",(tot\_score,tot\_time,uid))

conn.commit()

cursor.execute("select Id,Totalscore from users order by Totalscore desc");

record=cursor.fetchall()

count=0

for rec in record:

count+=1

if rec[0]==uid:

rank=count

pass

label.draw(f'{rank}',(0,0,0),570,140,screen,28,'Joker Man')

cursor.execute("select Name,Totalscore from users order by Totalscore desc limit 3")

record=cursor.fetchall()

i=0

for rec in record:

label.draw(f'{rec[0]}',(0,0,0),250,270+i,screen,28,'Joker Man')

label.draw(f'{rec[1]}',(0,0,0),550,270+i,screen,28,'Joker Man')

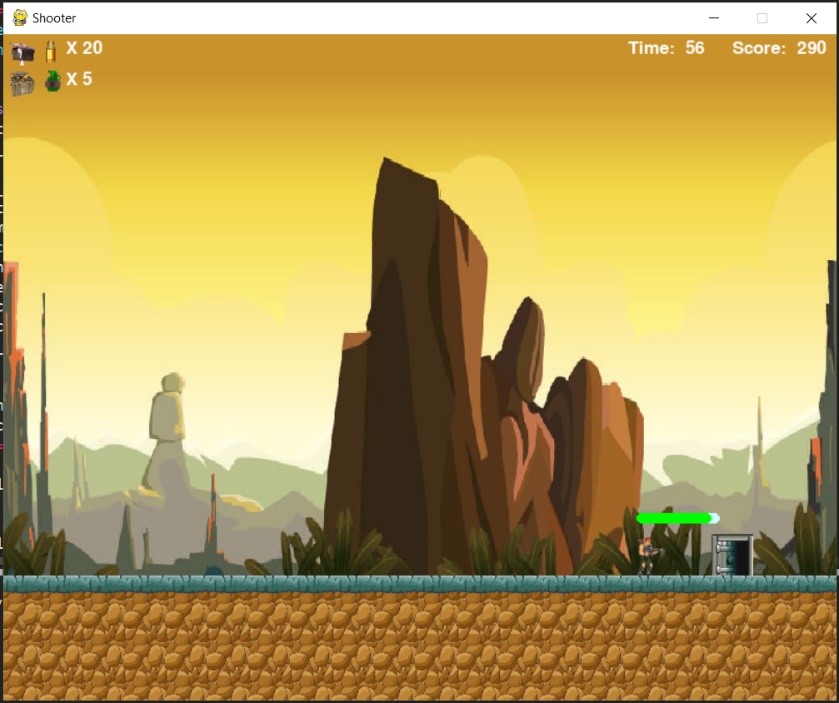
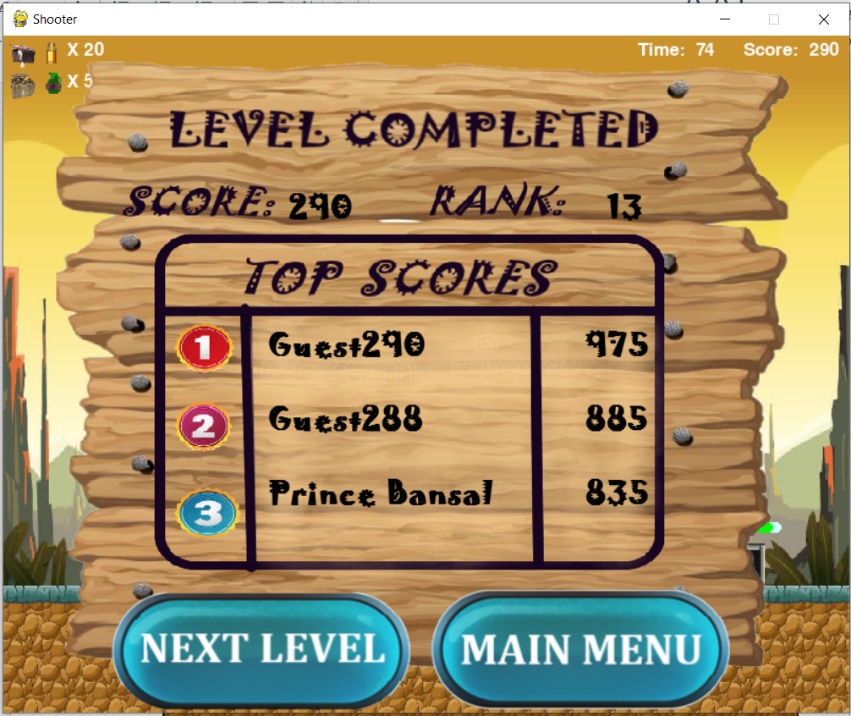
i+=70

pass

cursor.execute(f"select \* from users where Id={uid}")

record=cursor.fetchall()

Before Collision After Collision



(Player after collision with exit gate)

(Player before collision with exit gate)

Another important function that should be performed during the whole game is that the checking of health of player and enemies and then change in health bar size and colour when health decrease or increases corresponding to each enemy and player.

Formula used to calculate health bar’s width is:

**health\_bar\_width=(self.health/self.max\_health)\*MAX\_HEIGHT**

And the criteria used for the colour of health bar is that if health is greater than 75, then its colour will be “Green” and if health is greater than 50, then its colour will be “Orange” and if its health is less than 50, then its colour will be “Red” and if health becomes 0, then player or enemy will dies. Function used to draw health bar is:

def health\_bar(self):

pygame.draw.rect(screen,(204, 255, 255),pygame.Rect(self.rect.left,self.rect.top-20,80,10),0,10)

health\_bar=(self.health/self.max\_health)\*80;

if(self.health>75):

color=GREEN

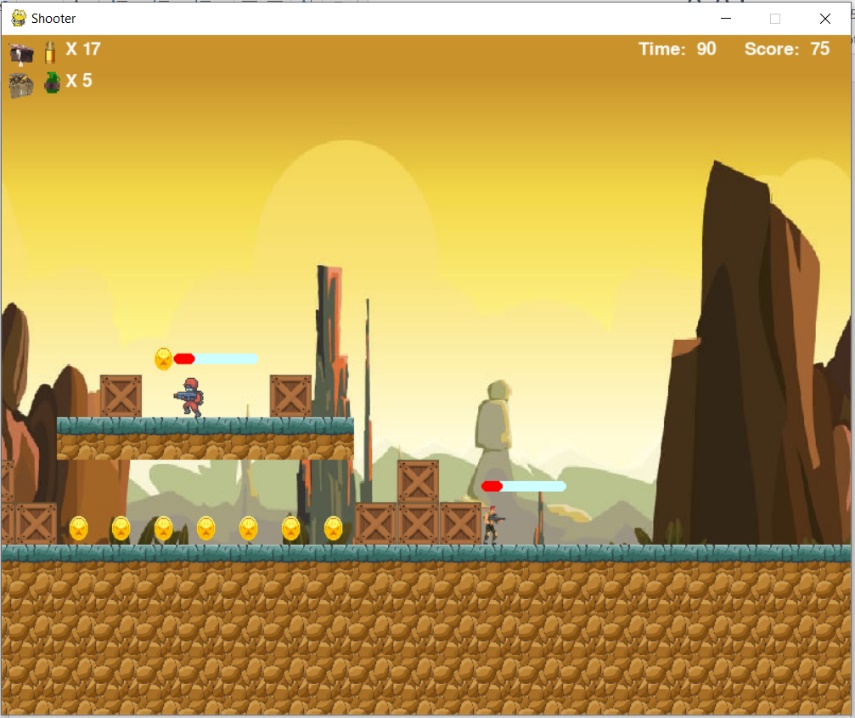
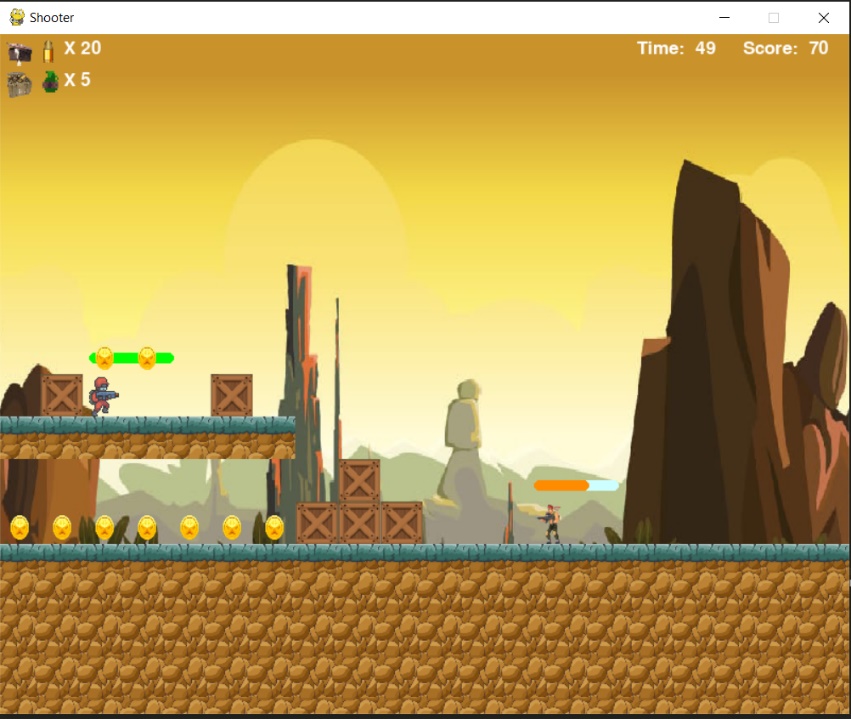
elif (self.health>50):

color=ORANGE

else :

color=RED

pygame.draw.rect(screen,color,pygame.Rect(self.rect.left,self.rect.top-20,health\_bar,10),0,10)



(Player & enemy with health<50)

(Player with 75>health>50)

1. **Evaluation of project**

At the end of each level , total score and rank of player will be evaluated on the basis of player’s score of all levels completed. Also, top 3 scores will be evaluated and displayed to the player.

To evaluate the total score, we first fetch scores of all levels and then add them, and then the total of scores of all levels will be displayed to the player and also stored in Total Score Column of database. Sql Query used to do this:

"select (1\_score+2\_score+3\_score+4\_score+5\_score+6\_score+7\_score+8\_score+9\_score),

(1\_time+2\_time+3\_time+4\_time+5\_time+6\_time+7\_time+8\_time+9\_time) from users where Id={uid}"

//Here uid is the User’s Id//

To evaluate rank of player, we first fetch total scores of all users in decreasing order and then perform searching in data and use a counter. When we found the user , counter value will be saved as rank. Code and SQL query used is:

#Query to fetch total scores of all users

cursor.execute("select Id,Totalscore from users order by Totalscore desc")

record=cursor.fetchall()

#counter will declared as 0

count=0

#perform searching

for rec in record:

#increase counter by 1

count+=1

if rec[0]==uid:

#value of count will be assigned to rank

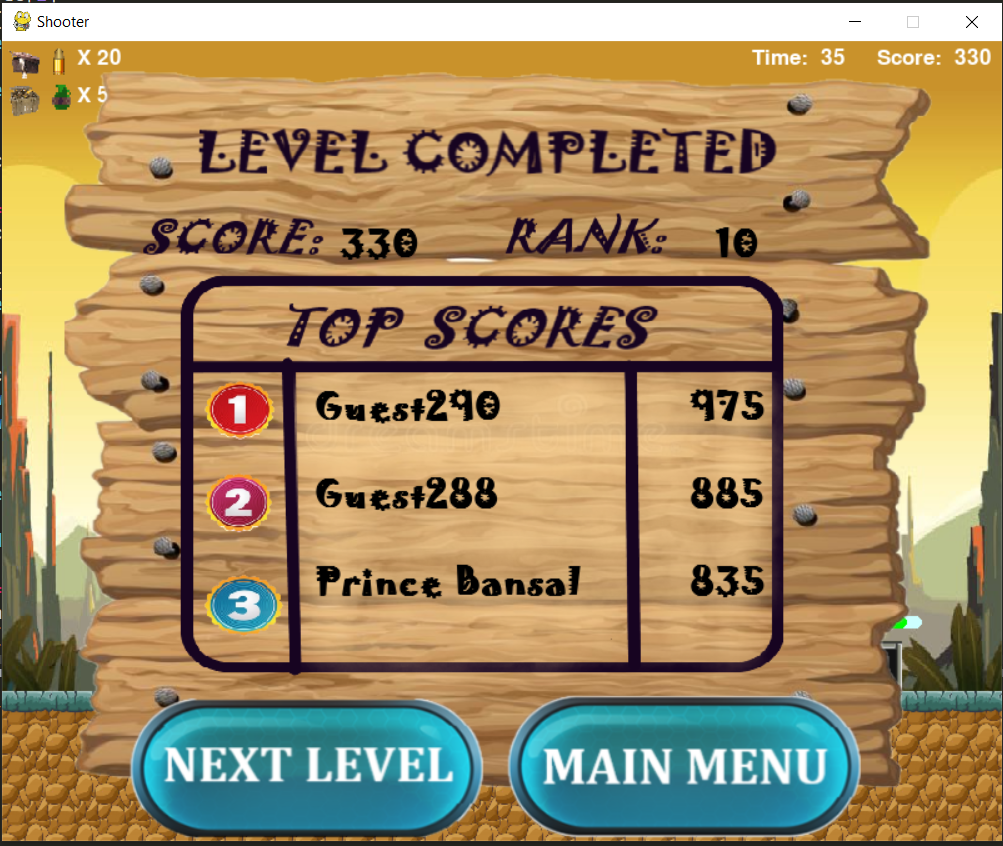
rank=count

pass

#rank will be displayed

label.draw(f'{rank}',(0,0,0),570,140,screen,28,'Joker Man')

At last we evaluate top 3 scores, to do this we use the following query:

cursor.execute("select Name,Totalscore from users order by Totalscore desc limit 3")

Total Score of all levels

Completed

Top 3 Scores

Rank of player

1. **Conclusion**

If I were to summarize the entire experience of the third year project in a word, it would be: adventure. Looking back to the past few months, I can truly assert it has been a journey that started slowly, without much progress due to the lack of technical knowledge required for this project, but took off while making progress with the research. I began the project with absolutely no idea on how graphics work, how a collision detection system works or even how to display an image on the screen and, by the end of the project, I got to a point where I could create my own or modify existing game specific algorithms to suit my needs. What is more, this was my first attempt to building a game and I believe that achieving to create a single player enemy shooter game with a smart power­up module and AI for NPCs is a good start if I decide to go down the game development path.

1. **Further Scope of project**

If Enemy Shooter Game was not a third year project and if the development was not constrained by a tight deadline, the player would have benefited from a number of extra features such as more levels, player selection, boss fights and a mobile version. Even though player selection was part of initial plan, it was not delivered due to the meticulosity other tasks were undertaken.

If Enemy Shooter were to be a real life project, the players would have enjoyed from a variety of level styles as well as boss fights to pass certain levels. To take the game one step further, it could have even be deployed on mobile. Despite Python is not the first programming language that comes to mind when one thinks of writing applications for either Android or iOS devices, tools such as Kivy can convert the Python code and create applications from its logic, following a Write Once, Deploy to Many concept.

1. **Bibliography**

* [**https://pypi.org/project/pygame/**](https://pypi.org/project/pygame/)
* [**https://en.wikipedia.org/wiki/Data-flow\_diagram**](https://en.wikipedia.org/wiki/Data-flow_diagram)
* [**https://www.dreamstime.com/illustration/posts-wooden.html**](https://www.dreamstime.com/illustration/posts-wooden.html)
* [**https://www.shutterstock.com/search/wood+ui**](https://www.shutterstock.com/search/wood+ui)
* [**https://craftpix.net/freebies/free-buttons-2d-game-objects/**](https://craftpix.net/freebies/free-buttons-2d-game-objects/)
* [**https://www.codester.com/items/**](https://www.codester.com/items/)
* [**https://www.youtube.com/watch?v=vR5utJvN4JY**](https://www.youtube.com/watch?v=vR5utJvN4JY)
* [**https://www.youtube.com/watch?v=FfWpgLFMI7w**](https://www.youtube.com/watch?v=FfWpgLFMI7w)