Computer Investigatory Project

**Class 12**

Premankur Chakraborty

Niroop Karera

Abhinav Kumar



CERTIFICATE

**NAME OF THE STUDENT : Premankur Chakraborty**

**GRADE : XII**

**ROLL No. : 23**

**INSTITUTION**: National Public School, Yelahanka.

**This is certified to be the bonafide project of the student in COMPUTER SCIENCE Laboratory during the academic year 2020-21.**

**He/She has completed the project titled *Computer Investigatory Project* successfully.**

**Teacher In-Charge**

**Examiner’s Signature Principal**

Date: Institution Stamp

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to our computer teacher, Suma ma’am, for providing me with the knowledge and opportunity to complete this project. I would like to thank the management and faculty at NPS Yelahanka for providing us with a well-equipped computer lab and for their immense support. Lastly, we would like to thank our principal, Latha ma’am, and our director, Rizwana ma’am, for the opportunity to do this project.

**INDEX**

* Project Synopsis - Pg. 5
* Theory - Pg. 6-8
* Files Used - Pg. 9-10
* Code - Pg. 11-27
* Output - Pg. 28-29

**SYNOPSIS**

Our project is a game in which the player must shoot as many enemies as

possible while ensuring that the enemies don’t come too close the player. If

the enemies are in the hitbox of the player then the player’s health gradually

decreases. As the game progresses the number of enemies increase making

the game more challenging. The game ends once the player has lost all of their

health. Health packs spawn in random locations in the field, which regenerates

the health of the player once the player walks over the health pack.

The controls for the game are: left arrow key to move left, right arrow key to

move right, up arrow key to move up, down arrow key to move down,

spacebar to shoot.

We have used the pygame module to create the background, characters,

bullets, music and animations. The 4 classes defined in the main program are

player, healthpack, projectile and enemy. These classes are responsible for the

animations and mechanics of the game. Other modules used include random,

mysql.connector and sqlmodule which is a user defined module responsible for

storing scores and opening the instructions text file. After the game ends the

player is asked to enter their username and their score along with their

username is saved.

**THEORY**

Our game uses a variety of files and features. These include the following:

Main.py

* Main.py is the main python file that contains most of the code we have created. It references all the other types of files that we have used, like the MP3, PNG, text, and user defined module files.
* This file serves 4 main functions-
* Firstly, it imports the various modules used. This will be covered in detail in the next section of this report. Apart from that, it defines a few important variables that are used later on in the program. These variables include, but are not limited to- window size and captioning, background image, sound effects and music, character images.
* Secondly, the main file also defines various classes which define the various attributes of all the aspects of our game, the protagonist, his enemies, the artillery and the powerups. We have also defined various functions which carry out specific functions for various scenarios in the game.
* Thirdly, the program includes the main loop. The main loop is the essence of the game. Everything is set into motion using this loop. It creates the content on the window every time it is run and it updates the whole screen in the next iteration by using a flip command.
* Lastly, the main file takes care of the back-end data storage and manipulation. This part includes a few function calls (these functions have been defined in a module) which allow the player to store his/her score into a database and also view other high scores. Further, the players are given the option to leave comments or ask questions about the game.

Sqlmodule.py

* This file is a user defined module that has two major parts. Firstly, is the definition and usage of functions which connect python to MySQL. Secondly, are a collection of functions which open text files to retrieve and store data like instructions, comments, queries, etc.
* This module uses the Python-MySQL interface to easily store data in the back end, and then display it in the front end. This is done by importing the mysql.connector module in python. MySQL Connector is a driver for connecting to a MySQL database server through the Open Database Connectivity (ODBC) application program interface (API), which is the standard means of connecting to any database.
* After creating a connection, a table is created (if it doesn’t already exist) and using this table, data is inserted, retrieved and displayed. The player is provided a variety of ways to look through previous scores by other players.
* Apart from this, the module also interacts with text files. This is used to print the game instructions which are stored in a simple text file. At the end of a game, the program also calls a function defined in the module which asks the user for any comments he/she has about the game. These are stored in a different file along with the player’s username.

Text Files

* Text files or .txt files are a very simple, yet useful type of file that allow efficient and easy interaction with python. These are used to store data and later retrieve it. All data is stored as ASCII or UNICODE, as text, as opposed to binary files which store information in binary code.
* Text files are used in python with an open statement which opens the file as a file handler in the mode (read, write or append) specified in the open statement. After this, the file can be read from, or written into using various functions.
* Our program includes two text files. One of them, Instructions.txt, stores the instructions on how to play the game. These are displayed at the start of every game. The UDF used to open and read from this file is defined in the sqlmodule.py and it also interacts with a different function within it to display the game’s top score from the SQL database.
* The other text file used is comments.txt. This file stores any comments or questions that a player has by writing into the file. This too is done with a UDF that is defined in the sqlmodule.py file and is called in main.py

PNG Files

* Image files are very important for our game’s presentation. Every kind of motion that a character can exhibit is stored as a separate image. These images, when displayed properly, create a sensation of motion that looks very convincing. Images are used in our project for the background image, protagonist movement, and enemy movement. There are separate images for left and right motion, which make the character appear to be walking.
* These images, used for motion, are loaded and stored into a list (one for each entity) and they are used when required.

MP3 Files

* MP3 files are used to store music and sound effects that are used in the game. Firstly, there is the background music that truly sets the intense mood for the game. Sound effects include the sound of a bullet being shot and that of an enemy being killed.
* These are used in the program by using the pygame.mixer module. In order to play music/audio files in pygame, pygame.mixer is used (pygame module for loading and playing sounds). This module contains classes for loading sound objects and controlling playback.
* The mixer is first initialized and then, the audio is loaded and played. Volume and other features can be controlled. For sound effects, the file which stores the sound is defined into a variable and the sound is played when required using the sound.play() code.

**FILES/MODULES USED**

* Sqlmodule.py: This is a user defined module. It connects to the back end and stores data in a table. It also interacts with text files. It contains the following functions:
  + tablecreate() -creates table if it doesn’t already exist
  + recordinsert(score) -inserts a player’s name and score into the table
  + displayhighscores(n) -displays ‘n’ of the highest scores
  + displayspecificuser(username) -displays score of a specific player
  + mediadriven() -runs a program to select database actions
  + gamestartinstructions() -prints game instructions in the beginning
  + suggestions() -asks users for comments/questions.
* Instructions.txt: This file contains the instructions printed in the beginning of the game.
* Comments.txt: This file contains the comments given by the players of the game.
* MP3 files:
  + Bg\_music.mp3 -background music
  + enemy\_dying.mp3 -sound effect
  + shoot\_sound.mp3 -sound effect
* PNG files:
  + A collection of about 35 images used for characters and the background.
* Classes used in the main program:
  + player() -defines the protagonist, controls his motion, health, and collisions.
  + healthpack() -defines the powerup that increases the player’s health.
  + projectile() -defines bullets shot by the protagonist
  + enemy() - defines the enemy, controls his motion and collisions.
* Inbuilt modules used:
  + Pygame- Pygame is the main contributor to our game. It allows basically all the major aspects of our game. It draws the windows, updates them, draws moving characters, checks for collisions, and does much more.
  + Random- This is used to spawn the enemies and the health powerup randomly within a range across the map. This is also used to create enemies that move with different speeds. This non-uniformity makes the game much more challenging and enjoyable to play.
  + Time- This module is used to keep track of the time in the game. It defines a clock, and is used to control the frame rate of the game.
  + Mysql.connector- MySQL and python are connected using this inbuilt module. It is used in the sqlmodule.py file.

**CODE**

For this file we have used two user defined modules – main.py and sqlmodule.py

The code for each is as follows:

Main.py

#change sql password

#fix sql database creation

import pygame

import random

import time

from sqlmodule import \*

tablecreate()

print('Instructions: ')

gamestartinstructions()

pygame.init() #initializing pygame window

pygame.mixer.init() #intializing pygame audio

#windowsize

winwidth=640 #The width of the window

winheight=360 #The height of the windo

win = pygame.display.set\_mode((winwidth, winheight)) #creating the window.

# win2=pygame.display.set\_mode((winwidth, winheight))

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

#sprites

walkLeft=[pygame.image.load('L1.png'),pygame.image.load('L2.png'),pygame.image.load('L3.png'),pygame.image.load('L4.png'),pygame.image.load('L5.png'),pygame.image.load('L6.png'),pygame.image.load('L7.png'),pygame.image.load('L8.png'),pygame.image.load('L9.png'),pygame.image.load('L1.png')]

walkRight=[pygame.image.load('R1.png'),pygame.image.load('R2.png'),pygame.image.load('R3.png'),pygame.image.load('R4.png'),pygame.image.load('R5.png'),pygame.image.load('R6.png'),pygame.image.load('R7.png'),pygame.image.load('R8.png'),pygame.image.load('R9.png'),pygame.image.load('R1.png')]

char=pygame.image.load('standing.png')

bg=pygame.image.load('background.png')

clock=pygame.time.Clock()

#Sounds

pygame.mixer.music.load('Bg\_music.mp3')

pygame.mixer.music.set\_volume(0.4)

shootSound = pygame.mixer.Sound('shoot\_sound.mp3')

enemyDyingSound = pygame.mixer.Sound('enemy\_dying.mp3')

class player():

def \_\_init\_\_(self,x,y,width,height):

self.x=x

self.y=y

self.width=width

self.height=height

self.vel=5

self.isJump=False

self.jumpCount=10

self.left= False

self.right= False

self.standing= True

self.walkCount=0

self.hitbox=(self.x+20,self.y,28,60)

self.health=100

self.score=0

def draw(self,win):

if self.walkCount+1>=27:

self.walkCount=0

if not(self.standing):

if self.left:

win.blit(walkLeft[self.walkCount//3],(self.x,self.y))

self.walkCount+=1

elif self.right:

win.blit(walkRight[self.walkCount//3],(self.x,self.y))

self.walkCount+=1

else:

#to keep character faced left or right

if self.right:

win.blit(walkRight[0],(self.x,self.y))

elif self.left:

win.blit(walkLeft[0],(self.x,self.y))

else:

#change this

win.blit(char,(self.x,self.y))

#win.blit(walkLeft[0],(self.x,self.y))

self.hitbox=(self.x+20,self.y,28,60)

pygame.draw.rect(win, (255,0,0), (self.hitbox[0], self.hitbox[1] -20,50,10))

#to decrease the health bar

pygame.draw.rect(win, (0,255,0), (self.hitbox[0], self.hitbox[1] -20,(50-0.5\*(100-self.health)),10))

#pygame.draw.rect(win,(255,0,0),self.hitbox,2)

def hit(self):

if self.health>0:

self.health-=1

def healthfunc(self):

if self.health>0 and self.health+20<=100:

self.health+=20

elif self.health+20>=100:

self.health=105

class healthpack():

def \_\_init\_\_(self,x,y,radius,color):

self.x=x

self.y=y

self.radius=radius

self.color=color

def draw(self,win):

pygame.draw.circle(win,self.color,(self.x,self.y),self.radius)

class projectile():

def \_\_init\_\_(self,x,y,radius,color,facing):

self.x=x

self.y=y

self.radius=radius

self.color=color

self.facing=facing

self.vel=8\*facing

def draw(self,win):

pygame.draw.circle(win,self.color,(self.x,self.y),self.radius)

class enemy():

walkLeft=[pygame.image.load('L1E.png'),pygame.image.load('L2E.png'),pygame.image.load('L3E.png'),pygame.image.load('L4E.png'),pygame.image.load('L5E.png'),pygame.image.load('L6E.png'),pygame.image.load('L7E.png'),pygame.image.load('L8E.png'),pygame.image.load('L9E.png'),pygame.image.load('L1E.png')]

walkRight=[pygame.image.load('R1E.png'),pygame.image.load('R2E.png'),pygame.image.load('R3E.png'),pygame.image.load('R4E.png'),pygame.image.load('R5E.png'),pygame.image.load('R6E.png'),pygame.image.load('R7E.png'),pygame.image.load('R8E.png'),pygame.image.load('R9E.png'),pygame.image.load('R1E.png')]

def \_\_init\_\_(self,x,y,width,height,end):

self.x=x

self.y=y

self.width=width

self.height=height

self.end=end

self.path=[self.x,self.end]

self.walkCount=0

self.vel=random.randint(1,3)

self.hitbox=(self.x+20,self.y,28,60)

def draw(self,win):

#self.move()

self.move1()

if self.walkCount+1>=27:

self.walkCount=0

#if enemy is left of player it will face right (towards the player)

if self.x<man.x:

win.blit(self.walkRight[self.walkCount//3],(self.x,self.y))

self.walkCount+=1

else:

win.blit(self.walkLeft[self.walkCount//3],(self.x,self.y))

self.walkCount+=1

self.hitbox=(self.x+20,self.y,28,60)

#pygame.draw.rect(win,(255,0,0),(self.x,self.y,self.width,self.height),2)

#used for movement of enemy(makes enemy follow player

def move(self):

if self.vel>0:

if self.x+self.vel<self.path[1]:

self.x+=self.vel

else:

self.vel=self.vel\*-1

self.walkCount=0

else:

if self.x-self.vel>self.path[0]:

self.x+=self.vel

else:

self.vel=self.vel\*-1

self.walkCount=0

def move1(self):

if man.x>self.x:

self.x+=self.vel\*.925

elif man.x<self.x:

self.x-=self.vel\*.925

if man.y<self.y:

self.y-=self.vel\*.925

elif man.y>self.y:

self.y+=self.vel\*.925

def hit(self):

enemyDyingSound.play()

#print('hit')

man.score+=1

#print(man.score)

def redrawGameWindow():

if man.health>0:

win.blit(bg,(0,0))

#pygame.draw.rect(win, (200,0,0),(x,y,width,height))

man.draw(win)

#enemy1.draw(win)

for i in enemies:

i.draw(win)

for i in bullets:

i.draw(win)

for i in healthpacks:

i.draw(win)

pygame.display.update()

font\_name = pygame.font.match\_font('arial')

def draw\_text(surf, text, size, x, y):

font = pygame.font.Font(font\_name, size)

text\_surface = font.render(text, True, (255,255,255))

text\_rect = text\_surface.get\_rect()

text\_rect.midtop = (x, y)

surf.blit(text\_surface, text\_rect)

running = True

def show\_go\_screen():

global running

draw\_text(win, 'GAME', 60, winwidth/2, winheight/4 )

draw\_text(win, 'Use Arrow keys to move and space bar to shoot', 34, winwidth/2, winheight/2)

draw\_text(win, 'Press any key to continue', 15, winwidth/2, winheight \* 3/4)

pygame.display.flip()

waiting = True

while waiting == True and running == True:

clock.tick(27)

for event in pygame.event.get():

if event.type==pygame.QUIT:

running = False

if event.type == pygame.KEYUP:

waiting = False

game\_over = True

#enemies=[enemy(random.randint(30,60),200,64,64,200),enemy(random.randint(30,60),100,64,64,200)]

#enemy1=enemy(60,110,64,64,200)

#delay before the game starts

pygame.time.delay(7000)

# count=0

pygame.mixer.music.play(loops=-1)

while running==True:

print('outer:',running, game\_over)

# count+=1

if game\_over == True:

print('game over 1:',running, game\_over)

show\_go\_screen()

print('Game over 2:', running, game\_over)

if running == True:

man=player(70,100,64,64)

enemies=[]

bullets=[]

healthpacks=[]

game\_over = False

print('game over 3:',running, game\_over)

draw\_text(win, f'Score: {man.score}', 18, winwidth/2, 7)

pygame.display.flip()

print(man.score)

# print(running, game\_over, count)

# print(running)

#controls frame rate (how fast the loop runs)

clock.tick(27)

for event in pygame.event.get():

if event.type==pygame.QUIT:

running=False

# break

# print(running)

if man.score<3:

countofenemies=2

elif man.score<7:

countofenemies=3

elif man.score<15:

countofenemies=4

else:

countofenemies=5

#colourrand=random.choice([(0,0,100),(100,0,0),(0,100,0),(50,50,0),(0,50,50),(50,0,50)])

while len(healthpacks)<1:

healthpacks.append(healthpack(random.randint(30,winwidth-50),random.randint(120,winheight-50),8,(0,0,100)))

while len(enemies)<countofenemies:

enemies.append(enemy(random.randint(30,winwidth),random.randint(120,winheight),64,64,200))

for x in enemies:

if man.hitbox[1]<x.hitbox[1]+x.hitbox[3] and man.hitbox[1]+man.hitbox[3]>x.hitbox[1]:

if man.hitbox[0] +man.hitbox[2]>x.hitbox[0] and man.hitbox[0]<x.hitbox[0]+x.hitbox[2]:

man.hit()

for i in healthpacks:

if i.y+i.radius<man.hitbox[1]+man.hitbox[3] and i.y+i.radius>man.hitbox[1]:

if i.x +i.radius>man.hitbox[0] and i.x-i.radius<man.hitbox[0]+man.hitbox[2]:

man.healthfunc()

#print("health")

healthpacks.pop(healthpacks.index(i))

for i in bullets:

for x in enemies:

#checks for collision within hitbox

if i.y-i.radius<x.hitbox[1]+x.hitbox[3] and i.y+i.radius>x.hitbox[1]:

if i.x +i.radius>x.hitbox[0] and i.x-i.radius<x.hitbox[0]+x.hitbox[2]:

x.hit()

enemies.pop(enemies.index(x))

if i in bullets:

bullets.pop(bullets.index(i))

if i.x<winwidth and i.x>0:

i.x+=i.vel

else:

bullets.pop(bullets.index(i))

keys=pygame.key.get\_pressed()

if keys[pygame.K\_SPACE]:

shootSound.play()

if man.left:

facing=-1

#tomoveleft

else:

facing=1

#shootright

#restricts n.o of bullets

if len(bullets)<1:

bullets.append(projectile(round(man.x+man.width//2),round(man.y+man.height//2),6,(100,0,0),facing))

#we use x+y/2 to project bullet from center of man

if keys[pygame.K\_LEFT] and man.x >man.vel:

man.x-=man.vel

man.left=True

man.right=False

man.standing=False

elif keys[pygame.K\_RIGHT] and man.x<winwidth-man.width-man.vel:

man.x+=man.vel

man.right=True

man.left=False

man.standing=False

else:

man.standing=True

man.walkCount=0

if not(man.isJump):

if keys[pygame.K\_UP] and man.y>man.vel:

man.y-=man.vel

man.right=False

man.left=False

man.standing=True

if keys[pygame.K\_DOWN] and man.y<winheight-man.height-man.vel:

man.y+=man.vel

man.standing=True

man.right=False

man.left=False

if man.health <=0:

game\_over = True

# print(running, game\_over, count)

redrawGameWindow()

pygame.quit()

print('\n'\*3)

print("Your score is ",man.score,"!!!")

recordinsert(man.score)

mediadriven()

suggestions()

sqlmodule.py

#Project SQL

#This program will act as a module that can be used to keep track of and display high scores

# from main import \*

import mysql.connector

mysqlPassword = input('Enter mysql password: ')

# p=input("Enter the password.") #We will replace this with the school pc's password later.

mycon=mysql.connector.connect(host='localhost',user='root',passwd=mysqlPassword)

curs=mycon.cursor()

def tablecreate():

try:

curs.execute("create database if not exists Project")

curs.execute("use Project")

curs.execute("create table SCORES(username char(20) primary key,timeofplay datetime,score int(10))") #adjust the range

#of score

except:

return

# this obviously is used only once. I left it as a function so we dont need to create

#for every new pc we use. the except makes sure the program doesnt stop when the table already exists.

def recordinsert(score):

try:

username=input("Enter your username.")

print()

curs.execute("insert into scores values('{}',now(),'{}')".format(username,score))

print("Your score has been saved in the database!!")

print()

except:

print("Your inputted username has already been used. ")

print()

recordinsert(score)

mycon.commit()

def displayhighscores(n):

#n is the number of highscores displayed, you can decide a value later. Or the user can be allowed to choose.

curs.execute("select \* from scores order by score desc")

res=curs.fetchall()

print("RANK",'\t',"NAME",'\t'\*3,"DATE OF GAME",'\t'\*3,"SCORE")

print()

for i in range(0,n):

try:

a,b,c=res[i]

print(i+1,'\t',a,'\t'\*3,b,'\t'\*2,c)

print()

except:

print()

print("Those are all the available records so far.")

print()

break

def displayspecificuser(username): #allows user to find their data specifically (obviously)

try:

curs.execute("select \* from scores where username='{}'".format(username))

res=curs.fetchall()

i=0

print("RANK",'\t',"NAME",'\t'\*3,"DATE OF GAME",'\t'\*3,"SCORE")

print()

a,b,c=res[i]

print(i+1,'\t',a,'\t'\*3,b,'\t'\*2,c)

print()

except:

print("Please try again. The given username does not exist.")

print()

username=input("Enter username. Enter 0 to quit.")

print()

if username=='0':

return

else:

displayspecificuser(username)

#media driven program is what I'm going for here, if y'all have a better idea, tell me , I'll switch it.

def mediadriven():

while True:

print("You can view the highscores, your score, and other users. You can:")

print()

print("1.View highscores.")

print("2.View a specific username's score.")

print("3.Exit")

print("\n"\*3)

choice=int(input("Enter your choice."))

print("\n"\*3)

if choice==3:

print()

print("Hope you enjoyed the game!! Thank you for playing.")

break

elif choice==1:

print()

n=int(input("How many highscores would you like to see?"))

print()

displayhighscores(n)

elif choice==2:

print()

u=input("Enter the username you would like to view.")

print()

displayspecificuser(u)

else:

print("INVALID INPUT")

#text file portion

def gamestartinstructions():

f=open('instructions.txt','r')

for i in f:

print(i)

displayhighscores(1)

def suggestions():

while True:

#asks users if they have any queries/suggestions/criticism about the game

inp=input("Do you have any questions, suggestions, or comments about our game? (Y/N)")

if inp.lower()=='y' or inp.lower()=='yes':

f=open('comments.txt',"a")

line=input('Enter your comments/questions.')

f.write(line+'\n')

print("Thank you for the feedback.")

return

elif inp.lower()=='n' or inp.lower()=='no':

print('Ok. Thank you for playing the game.')

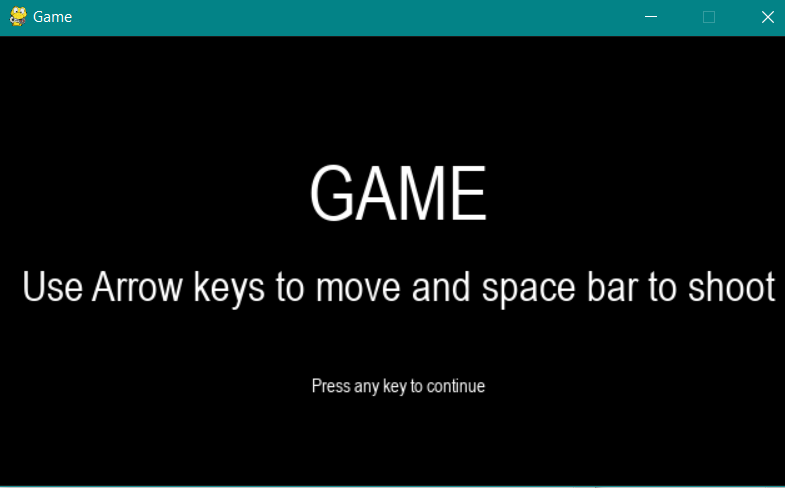
return

else:

print('Enter a valid input.')

**OUTPUT SCREENS**

Output Screen 1:

****

Output Screen 2:

Output Screen 3:

