Computer Science-Project

BIRLA VIDYA NIKETAN

(Sarala Birla Group of Schools)

CERTIFICATE

THIS IS TO CERTIFY THAT ADITYA SHARMA, ANSHUMAN KAR AND RITAM RAJ OF CLASS XII-C SUCCESSFULLY COMPLETED THE PROJECT WORK UNDER THE GUIDANCE OF MRS. DEEPIKA PAREEK(SUBJECT TEACHER) DURING THE YEAR 2021-22
IN PARTIAL FULFILMENT OF COMPUTER SCIENCE PRACTICAL EXAMINATION OF CENTRAL BOARD OF SECONDARY EDUCATION(CBSE)

MRS. DEEPIKA PAREEK

COMPUTER SCIENCE HOD

ACKNOWLEDGMENT

We would like to thank our school “Birla Vidya Niketan” and teacher Mrs. Deepika Pareek who gave us this opportunity to work on this wonderful project. Thank you for your guidance.

INTRODUCTION

Find and eat the energy balls to gain a score! You will be spawned in a black screen and you have to find the energy ball hidden in the screen, your cursor will start to change colors once near the energy ball. You can press space to reveal the location of the energy ball. (applicable only upto two times per game)

INSPIRATION

Our inspiration was drawn when we first heard about the module *pygame* and saw how students of the previous batch had made amazing and fun games using it. We did not come up with this idea on the first try, we slowly progressed from making a snake-like game to something which looks similar to our current game.

MODULES USED

1) *pygame :*

pygame is a free and open-source cross-platform library for the development of multimedia applications like video games using python, here we have used it as the main framework for our game.

2) *random :*

random implements pseudo-random number generations, here it has been used to generate random coordinates and colors.

3) *mysql.connector :*

mysql.connector enables python programs to access MySQL databases, using an API, here it has been used to bridge a connection between our game and a local MySQL database.

4) *time :*

time provides various time-related functions, here it has been used to keep a track of time before closing a window.

SOURCE CODE

Imports used for making the game

```
1   #------------IMPORTS--------------
  1 import pygame
  2 import random
  3 import mysql.connector
  4 import time
  5 #---------------------------------
  6
```

Variables for the game.

```
 6
 7 #------------VARIABLES------------
 8 s = 0
 9 x, y = 150, 150
10 tx, ty = random.randrange(20, 281, 10), random.randrange(20, 281, 10)
11 velocity = 10
12 radius = 10
13 run = True
14 swidth = 300
15 sheight = 300
16 scoreSum = 0
17 count = 0
18 r, g, b = 255, 255, 0
19 tr, tg, tb = 0, 0, 0
20 can = 0
21 #---------------------------------
```

While making the game we kept in mind that the person using this might not be as technologically literate so we made sure that the person does not have to edit the source

code themselves, i.e our game asks for the MySQL password of the local machine and automatically sees if the local machine has the correct database/table already present or not and makes and/or drops the database/table accordingly. In short the person playing this game does not have to edit the MySQL connectivity code in the python file themselves the code handles that. We were able to achieve this using try and except blocks.

Asking for the root password

```
27
26 #--------ASKING_PASSWORD----------
25 password=input("Please enter the password for the root user: ")
24 passgame=password
23 #---------------------------------
22
```

Automatic database handling

```
29
28 #-----------SQL_CONNECTION-------
27
26 # trying to connect to an existing database named "game" for an existing user
25 try:
24     db = mysql.connector.connect(host="localhost",
23             user="root",
22             passwd='{}'.format(passgame,),
21             database="game"
20             )
19     cursor = db.cursor()
18     try: # tries to create a table score if not already present
17         cursor.execute("CREATE TABLE scores (id INT PRIMARY KEY AUTO_INCREMENT, score INT)")
16     except:
15         except_temp = 0
14
13 # if no database named "game" is present then it makes one for a new user
12 except:
11     db = mysql.connector.connect(host="localhost",
10             user="root",
 9             passwd="{}".format(passgame,)
 8             )
 7     cursor = db.cursor()
 6     cursor.execute("CREATE DATABASE game")
 5     cursor.execute("USE game")
 4     cursor.execute("CREATE TABLE scores (id INT PRIMARY KEY AUTO_INCREMENT, score INT)")
 3 #---------------------------------
 2
```

Our main use of connectivity with a MySQL server was to store the scores of previous game sessions and take the average of them to predict the expected score.

Taking average score

```
27
26 #------------TAKING_THE_AVG_SCORE----------------
25
24 su, c = 0, 0
23 try :
22         cursor.execute("SELECT SUM(score) FROM scores")
21         for i in cursor:
20                 scoreSum = i
19         cursor.execute("SELECT COUNT(score) from scores")
18         for i in cursor:
17                 count = i
16         for i in scoreSum:
15                 su = i
14         for i in count:
13                 c = i
12         avg = su // c
11 except :  # <--- If the table is empty
10     avg = 0
 9 #------------------------------------------------
 8
```

When using *pygame* we have to first initialise it by setting up the window size and the window title followed by a main game loop.

pygame initialisation

```
20
19 #------------PYGAME_INITIALISATION---------------
18 pygame.init()
17 window = pygame.display.set_mode((swidth, sheight))
16 pygame.display.set_caption("EAT EM' ALL")
15 #------------------------------------------------
14
```

The main code of the game lies in the game loop which has a structure of something like this:

```
while run:
    '''

    game code and logic

    '''
```

Since the game loop is the most important part of our code we have decided to break it down into different parts according to their functioning.

1) Setting up the time delay and the closing window functionality.

```

#------------PYGAME_LOOP-----------------------------------------------
while run:
        pygame.time.delay(55)
        for event in pygame.event.get():
                if event.type == pygame.QUIT:
                        run = False

```

The *pygame.time.delay(55)* means the frequency for the game, therefore the 55 here means that the game loop runs every 55 milliseconds.

pygame.event.get() gets an array of all the events happening with the game window. Here we check if the event type is of *pygame.QUIT*, i.e if the close button of the

window is pressed, it then updates the value of *run* to be *False* and thus quitting the game loop in the next iteration.

2) Setting up temporary r,g,b values and the display strings.

```
4
5           tr,tg,tb = 0, 0, 0
6           avgDisp = 'Average expected score: ' + str(avg)
7           scoreDisp = 'Score: ' + str(s)
8           myfont = pygame.font.SysFont('Comic Sans MS', 15)
9           Cscore = myfont.render(scoreDisp, False, (255, 255, 255))
10          Cavg = myfont.render(avgDisp, False, (255,255,255))
11
12          keys = pygame.key.get_pressed()
13
```

Here we set the *tr,tg,tb* values for the temporary energy blob and the display strings which will be shown on the screen. We also assign the variable *keys* a list of all the keypresses made by the user, this will help us to add functionality into the game according to a specific key.

3) Using the *keys* for detection and movement

```
        '''
        Movement/Detection keys
        '''
        if keys[pygame.K_RIGHT] and x < swidth - radius:
                x += velocity
        if keys[pygame.K_LEFT] and x >= velocity:
                x -= velocity
        if keys[pygame.K_DOWN] and y < sheight - radius:
                y += velocity
        if keys[pygame.K_UP] and y >= velocity:
                y -= velocity
        if keys[pygame.K_SPACE]:
                if can < 2:
                        tr, tg, tb = 255, 0, 0
                        can += 1
                        time.sleep(1)
                        pygame.draw.circle(window, (tr, tg, tb), (tx, ty), 5)

```

We change the *X* & *Y* coordinates of the main cursor according to the velocity and the keys pressed. It must be noted that while going up the *Y* coordinate is reduced instead of being increased, this is because the coordinates in *pygame* start from the top left corner.

The color of the hidden blob is changed from *balck (0,0,0)* to *red (255,0,0)* when the spacebar is pressed, also note the condition *'if can < 2'* this makes sure that the special power is not used more than twice per game. It also waits for a second before revealing the location of the blob.

4) Drawing contents of the main window.

```

        window.fill((0,0,0))
        pygame.draw.circle(window, (tr, tg, tb), (tx, ty), 5) # <--- Drawing the small random blob
        pygame.draw.circle(window, (r, g, b), (x, y), radius) # <--- Drawing the main blob
        window.blit(Cscore,(0,0)) # Displaying the current score at the top left
        window.blit(Cavg,(100, 0)) # Displaying the average expected score at the top right
        pygame.display.update()

```

Here we fill the window with a black overlay and draw the cursor and the hidden blob on the screen. We also blit the current score and the average score on the screen at appropriate locations, at the end we update the display for the effects to actually take place.

5) Changing colors and updating the score.

```
23
22          # Checking if the main blob is in the vicinity and changing colours.
21          if ((ty == y) or (ty > y and ty < y + radius + 20) or (ty < y and ty > y - (radius + 20))) and ((tx == x) or (tx > x and t
    x < x + radius + 20) or (tx < x and tx > x - (radius +20))):
20                  r, g, b = random.randrange(0, 256),random.randrange(0, 256),random.randrange(0, 256)
19
18          # Checking if the small blob has been consumed or not
17          if ((ty == y) or (ty > y and ty < y + radius) or (ty < y and ty > y - radius)) and ((tx == x) or (tx > x and tx < x + radi
    us) or (tx < x and tx > x - radius)):
16                  s += 10
15                  tx, ty = random.randrange(20, 181, 10), random.randrange(20, 181, 10) # Generating random coordinates for the smal
    l blob
14                  pygame.display.update()
13
```

The color of the cursor changes when it is in the vicinity of the hidden blob. We achieved this by adding some basic mathematical conditions to the code.

Firstly we check if the *(x,y)* coordinates are equal to the *(tx,ty)* coordinates, we also checked if the *(x,y)* coordinated are in the vicinity or not by adding/subtracting from them to check the nearing area, if *(tx,ty)* were in that area then we would produce new values for *r, g, b* using the *random* module. The new colors will take effect upon the next iteration of the game loop.

Secondly we also checked if the coordinates of the hidden blob *(tx,ty)* were in the area of the cursor *(x,y)*, if so then we would remove the blob and generate new values for *(tx,ty)* and increase the score as well, all of this will take effect in the next iteration of the game loop, i.e 55

milliseconds later. We achieved this by adding/subtracting the radius of the cursor from its *x* and *y* coordinates and checking if tx and ty were in that range or not.

Note
All the random values being generated for *(tx,ty)* are in the range of the width and height of the screen, this is done to prevent the spawning of hidden blobs outside the playable area.

6) Checking if the player has touched the boundary.

```
13
12          # Checking if the player has touched the boundary
11          if (y <= radius or y  >= sheight - radius) or (x <= radius or x >= swidth- radius):
10                  run = False
 9                  myfont = pygame.font.SysFont('Comic Sans MS', 30)
 8                  Cscore = myfont.render(scoreDisp, False, (255, 255, 255))
 7                  window.blit(Cscore,(100,130)) # <--- Displaying the current score at the middle of the screen
 6                  pygame.display.update()
 5                  time.sleep(2) # <--- Waiting for 2 seconds before closing the window
 4
 3
 2 pygame.quit()
 1 #--------------------------------------------------------------------
```

To check if the cursor of the player has touched the boundaries of the window we added a condition to check if the *(x,y)* coordinates of the cursor are equal to or exceed the width, height respectively, if so then then we set the value of *run* to be *False* and the game stops in the next iteration but before that the current score is shown on the screen for 2 seconds.

The *pygame.quit()* is written after the game loop so that the game window closes once the game's over.

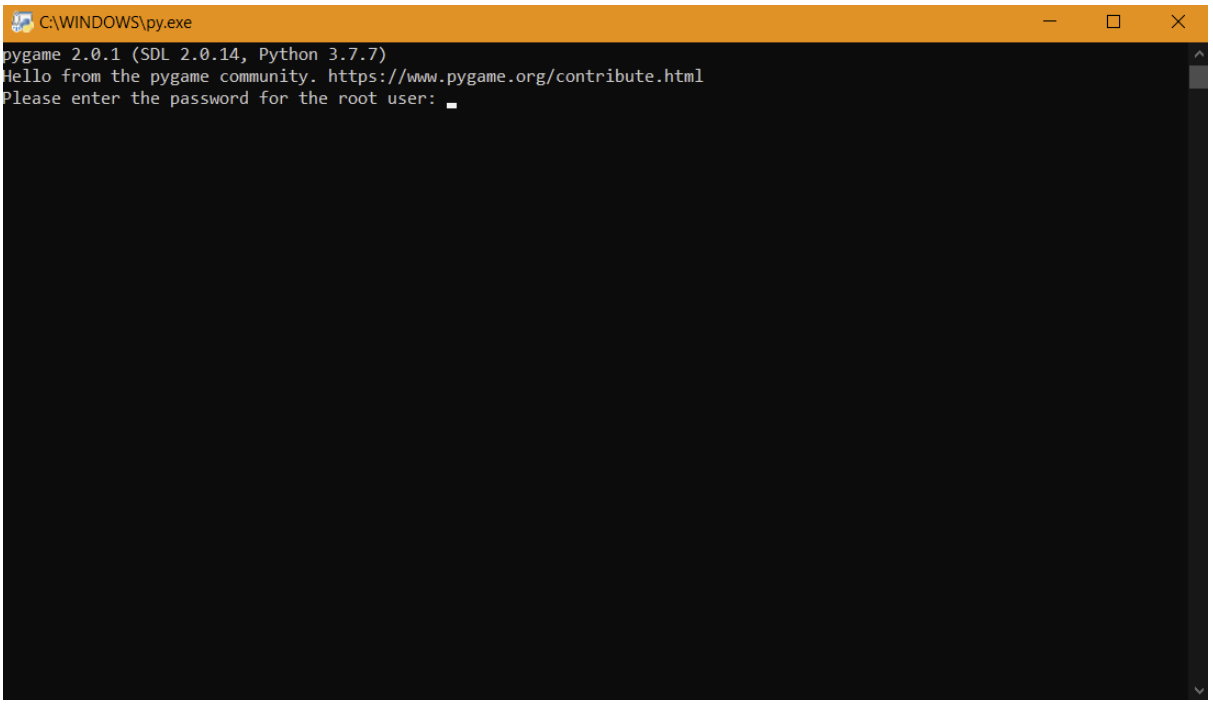
7) Saving in MySQL database.

```
7 #-------------------------------------------------------------------
6
5 #------------SAVING_IN_DATABASE-------------------
4 cursor.execute("INSERT INTO scores(score) VALUES(%s)", (s,))
3 db.commit()
2 #-------------------------------------------------
1
```

After the game finishes the only thing left to do is to save the current score in the MySQL database to which the connection was established at the start of the code. This score will be used in calculating the average the next time.

OUTPUT

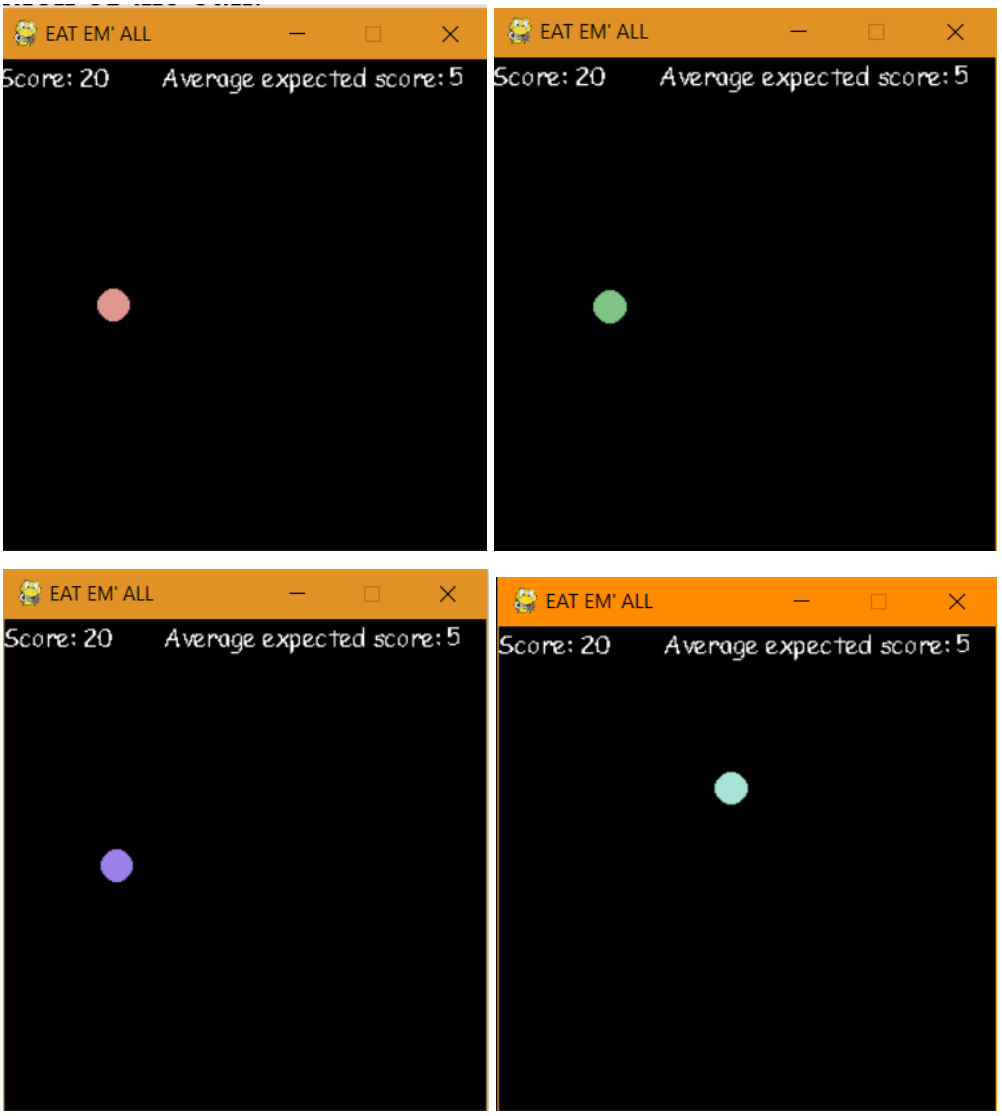
When we run the program python asks for our root password for MySQL.



After entering the password a new pygame window pops up and the pointer is in the center of the screen.

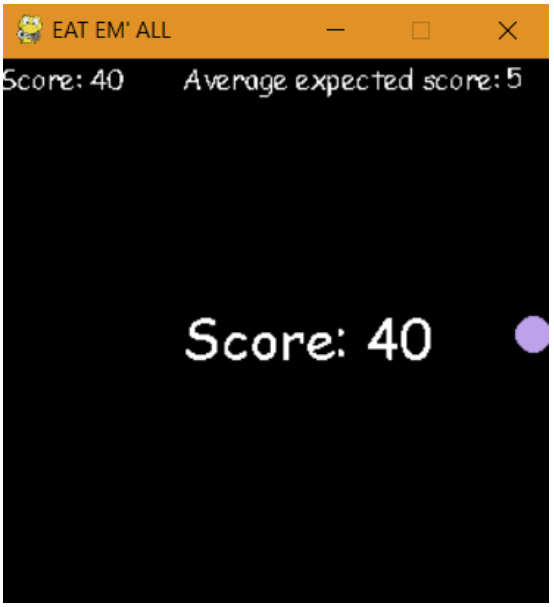
When the spacebar is pressed a red ball shows up for less than a second indicating the position of the ball.

When the cursor is near the ball it starts to change colors indicating that we are in the vicinity of the ball.



The score represents the number of points the character has collected so far in which one ball accounts for 10 points, while the average expected score is calculated by taking the average of scores made in all the previous games that is fetched from the SQL server .

If the character hits any of the four walls the game ends and the score is displayed at the centre of the window and thereafter the window closes.



Thank you !

If you wish to download the game file and test it for yourselves then visit our website at:

https://adityasharma223.github.io/EatEmAll

MADE BY -

1. Aditya Sharma (12-C)
2. Anshuman Kar (12-C)
3. Ritam Raj (12-C)