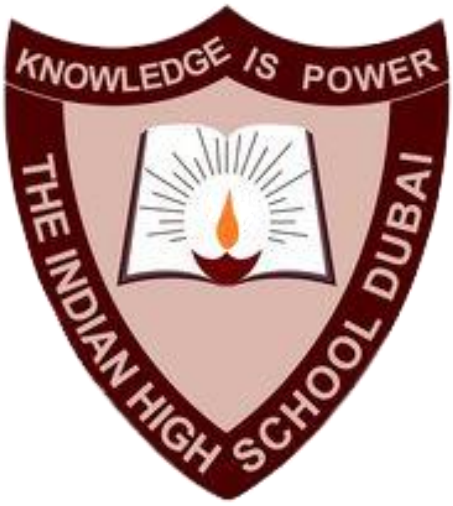


A

PROJECT ON

ARCADIA

SUBMITTED BY:

Dhanvin Sajith Roll :

Aldrin Dsouza Roll :

THE INDIAN HIGH SCHOOL

DUBAI

CERTIFICATE

This is to certify that the work in this project is the bonafide work of

Master _____

Class _____ *Div*_____ *Roll No.*_____

recorded in the school lab during the academic year

2022-2023

Date: _____

_____

Teacher in – Charge

External Examiner: _____ **Internal Examiner:**

_____

Proud winner of the Sheikh Hamdan bin Rashid Al Maktoum Award for Distinguished School & School Administration, 2002 & 2005

ACKNOWLEDGEMENT

I would like to take this opportunity to thank the Central Board of Secondary Education (CBSE) and our beloved The Indian High School, Dubai (IHS), for granting me the opportunity to expand the depth of my knowledge in my favorite subject, computer science.

I would also like to thank my teacher Mrs.Swapnil Verma for guiding me and sharing her wide variety of knowledge. We are honored to have this opportunity to showcase our skills and talents.

Index

Introduction

Arcadia is a user profile-based application filled with six classic minigames: Flappy bird, endless runner, snake, pong, rock-paper-scissors and tictactoe.

The premise for our project was to build an application that can deliver the provision for multiple games, which reduces the dependency of using separate apps for each game.

Arcadia also serves the purpose of being a stress buster through its simple and minimalistic art style.

The upcoming models as well as any upcoming entrance exam preparation is what compelled us to come up with Arcadia.

Arcadia is the culmination of all our hard work and dedication as well as the fruit of our passion and labour.

Files

Files are those key components which helped with the smooth functioning of any application. The following are the files used in Arcadia:-

For the main program file, a .py file was used.

For the purpose of integrating images along with the graphics a .png file was utilized. This was necessary for creating many important in-game graphics which elevated the overall gaming experience.

A .ico file was used for creating the application icon. This helps to uniquely identify Arcadia on your device and adds a much-needed brightness to your desktop.

Functions

Our program contains a myriad of functions, but there remain a few that stand above the rest with regards to importance. They include: -

login() – lets the user create a new account or sign in.

mainMenu() – this function is responsible for the assembly of the main menu.

gamesScreen() – this function is responsible for the screen displaying list of games available to the player.

updateFirstDatabase() - updates the highscore data after every run in every score based game.

scoreboardScreen() - displays the highscores on each score based game on a new screen.

flappyBird() – as the name suggests this function runs the code related to the famous game flappy bird.

pong() – the code for a game of pong lies well hidden within this unsuspecting function.

snake() – remember snake? Its been ages but still feels like yesterday thanks to this function.

endlessRunner() – this function takes a twist on most endless runner games and adds that calm relaxed 8 bit-esque feel to it.

rockPaperScissors() – this function recreates the all-time classic game, except digitally for all you sophisticated folks.

ticTacToe() – ever wanted to play a good old game of ticktacktoe but are out of paper? Well, no

look no further because this function has got you covered.

Python MySQL Connectivity

As promised, we followed through with adding the leader boards.

This was handled my MySQL Connectivity

Here’s how it works; the database collects and stores the top records for showcase whenever needed.

```
#updates SQL database
def updateFirstDatabase(gcode, name, score, userid, scoreDat):
    fbSc, erSc, sSc = scoreDat['FlappyBird']['score'], scoreDat['EndlessRunner']['score'], scoreDat['Snake']['score']
    cursor.execute("USE arcadia")
    cursor.execute("CREATE TABLE IF NOT EXISTS highscores(GameCode int PRIMARY KEY,GameName VARCHAR(20), Highscore int, User VARCHAR(20))")
    if fbSc == -1 and erSc == -1 and sSc == -1:
        cursor.execute("insert into highscores values (1, 'FlappyBird', -1, NULL), (2, 'EndlessRunner', -1, NULL), (3, 'Snake', -1, NULL)")
    print('1')
    if gcode == 1: cursor.execute(f"UPDATE highscores SET GameName = '%s', Highscore = %s, User = '%s' WHERE GameCode = 1" % (name, score, userid,))
    elif gcode == 2: cursor.execute(f"UPDATE highscores SET GameName = '%s', Highscore = %s, User = '%s' WHERE GameCode = 2" % (name, score, userid,))
    elif gcode == 3: cursor.execute(f"UPDATE highscores SET GameName = '%s', Highscore = %s, User = '%s' WHERE GameCode = 3" % (name, score, userid,))
    arcadiaDB.commit()
```

```
mysql> use arcadia
Database changed
mysql> select * from highscores;
+----------+---------------+-----------+-----------+
| GameCode | GameName      | Highscore | User      |
+----------+---------------+-----------+-----------+
|        1 | FlappyBird    |         4 | userone   |
|        2 | EndlessRunner |       193 | usertwo   |
|        3 | Snake         |         8 | userthree |
+----------+---------------+-----------+-----------+
3 rows in set (0.00 sec)

mysql>
```

Code

```
import pygame
from pygame.math import Vector2
from pygame.locals import *
import random
import sys
import time
from sys import exit
import json
import pwinput
import mysql.connector

#environment initialization
directory = __file__.replace('\\', '/').strip('/main.py')
print(directory)
pygame.init()
pygame.display.set_caption('Arcadia')
pygame.display.set_icon(pygame.image.load(f'{directory}/media/
appIcon.png'))
clock = pygame.time.Clock()

#database initialisation
arcadiaDB = mysql.connector.connect(
    host="localhost",
```

```
    port=3306,
    user="root",
password="dangerfluid2005",)
cursor = arcadiaDB.cursor()
cursor.execute("CREATE DATABASE IF NOT EXISTS arcadia")

#login credentials
def login():
    check, loop = True, True
    with open(f'{directory}/login.json', 'r') as f:
        loginDetails = json.load(f)
    while loop:
        while check:
            ch = input("Create Account?(y/n) ")
            if ch in 'yYnN': check = False
            else: print('Please enter a valid input.')
        if ch in 'yY':
            username = input("Username: ")
            if len(username)>20:
                print("Username must be under 20 characters!")
                continue
            password = pwinput.pwinput()
            loginDetails[f'{username}'] = password
            with open(f'{directory}/login.json', 'w') as f:
                json.dump(loginDetails, f)
        elif ch in 'nN':
            username = input("Username: ")
            usernames = list(loginDetails.keys())
            if username not in usernames:
                print("User I.D not found!")
                check = True
                continue
            password = pwinput.pwinput()
            if password != loginDetails[f'{username}']:
                print('Incorrect username or password!')
                check = True
                continue
        loop = False

    return username, pygame.display.set_mode((1280, 720))

#updates SQL database
def updateFirstDatabase(gcode, name, score, userid, scoreDat):
    fbSc, erSc, sSc = scoreDat['FlappyBird']['score'],
scoreDat['EndlessRunner']['score'], scoreDat['Snake']['score']
```

```
    cursor.execute("USE arcadia")
    cursor.execute("CREATE TABLE IF NOT EXISTS
highscores(GameCode int PRIMARY KEY,GameName VARCHAR(20),
Highscore int, User VARCHAR(20))")
    if fbSc == -1 and erSc == -1 and sSc == -1:
        cursor.execute("insert into highscores values (1,
'FlappyBird', -1, NULL), (2, 'EndlessRunner', -1, NULL), (3,
'Snake', -1, NULL)")
    print('1')
    if gcode == 1: cursor.execute(f"UPDATE highscores SET
GameName = '%s', Highscore = %s, User = '%s' WHERE GameCode =
1" % (name, score, userid,))
    elif gcode == 2: cursor.execute(f"UPDATE highscores SET
GameName = '%s', Highscore = %s, User = '%s' WHERE GameCode =
2" % (name, score, userid,))
    elif gcode == 3: cursor.execute(f"UPDATE highscores SET
GameName = '%s', Highscore = %s, User = '%s' WHERE GameCode =
3" % (name, score, userid,))
    arcadiaDB.commit()

#Main Menu State
def mainMenu():
    #fonts
    regularFont =
pygame.font.Font(f'{directory}/font/dogica.ttf', 100)
    fontUnspaced =
pygame.font.Font(f'{directory}/font/dogicapixel.ttf', 60)
    creditsFont =
pygame.font.Font(f'{directory}/font/dogicapixel.ttf', 25)

    #objects
    mainBgSurface =
pygame.image.load(f'{directory}/media/mainMenuBG.png').convert
()
    mainMenuButton =
pygame.image.load(f'{directory}/media/mainMenuButton.png').con
vert_alpha()
    mainMenuButtonHover =
pygame.image.load(f'{directory}/media/mainMenuButtonHover.png'
).convert_alpha()

    #texts
    mainTitle = regularFont.render('Arcadia', False, (140, 3,
252))
    mainTitleRect = mainTitle.get_rect(center = (640, 180))
```

```
    playButtonText = fontUnspaced.render('Play', False,
'#170030')
    playButtonTextHover = fontUnspaced.render('Play', False,
'#4d018c')
    playButtonTextRect = playButtonText.get_rect(center =
(450, 380))
    exitButtonText = fontUnspaced.render('Exit', False,
'#170030')
    exitButtonTextHover = fontUnspaced.render('Exit', False,
'#4d018c')
    exitButtonTextRect = exitButtonText.get_rect(center =
(830, 380))
    credits = creditsFont.render('A game by Dhanvin Sajith and
Aldrin Dsouza', False, (140, 3, 252))
    creditsRect = credits.get_rect(center = (640, 565))

    #object rects
    playButtonRect = mainMenuButton.get_rect(center = (450,
380))
    playButtonRectHover = mainMenuButtonHover.get_rect(center
= (450, 380))
    exitButtonRect = mainMenuButton.get_rect(center = (830,
380))
    exitButtonRectHover = mainMenuButtonHover.get_rect(center
= (830, 380))

    #game loop
    while True:
        #event checker
        for event in pygame.event.get():
            #quit on clicking close window button
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
            #loading game list screen
            if event.type == pygame.MOUSEBUTTONUP:
                if playButtonRect.collidepoint(event.pos):
gamesScreen()
            #quit game on exit button
            if event.type == pygame.MOUSEBUTTONUP and
exitButtonRect.collidepoint(event.pos):
                pygame.quit()
                exit()

        #background
```

```
        screen.blit(mainBgSurface, (0, 0))

        #buttons
        if
playButtonRect.collidepoint(pygame.mouse.get_pos()):
            screen.blit(mainMenuButtonHover,
playButtonRectHover)
            screen.blit(playButtonTextHover,
playButtonTextRect)
        else:
            screen.blit(mainMenuButton, playButtonRect)
            screen.blit(playButtonText, playButtonTextRect)
        if
exitButtonRect.collidepoint(pygame.mouse.get_pos()):
            screen.blit(mainMenuButtonHover,
exitButtonRectHover)
            screen.blit(exitButtonTextHover,
exitButtonTextRect)
        else:
            screen.blit(mainMenuButton, exitButtonRect)
            screen.blit(exitButtonText, exitButtonTextRect)

        #text
        screen.blit(mainTitle, mainTitleRect)
        screen.blit(credits, creditsRect)


        #update frames and cap framerate
        pygame.display.update()
        clock.tick(60)

#Choose game to play
def gamesScreen():
    #reset screen size on returning
    screen = pygame.display.set_mode((1280, 720))

    #offsets on scrolling
    yOffset = 0
    scrollBarPos = 200

    #variable to check if scrolling
    scrolling = False

    #fonts
```

```
    regularFont =
pygame.font.Font(f'{directory}/font/dogica.ttf', 60)
    nameFont =
pygame.font.Font(f'{directory}/font/dogicapixel.ttf', 30)

    #objects
    mainBgSurface =
pygame.image.load(f'{directory}/media/mainMenuBG.png').convert
()
    trophyImage =
pygame.image.load(f'{directory}/media/trophy.png').convert_alp
ha()
    flappyBirdThumbnailImage =
pygame.image.load(f'{directory}/media/flappyBird/flappyBirdThu
mbnail.png').convert_alpha()
    pongThumbnailImage =
pygame.image.load(f'{directory}/media/pong/pongThumbnail.png')
.convert_alpha()
    snakeThumbnailImage =
pygame.image.load(f'{directory}/media/snake/snakeThumbnail.png
').convert_alpha()
    endlessRunnerThumbnailImage =
pygame.image.load(f'{directory}/media/endlessRunner/endlessRun
nerThumbnail.png').convert_alpha()
    tictactoeThumbnailImage =
pygame.image.load(f'{directory}/media/tictactoe/tictactoeThumb
nail.png').convert_alpha()
    rpsThumbnailImage =
pygame.image.load(f'{directory}/media/rps/rpsThumbnail.png').c
onvert_alpha()

    #scaling
    trophyImageMod = pygame.transform.scale(trophyImage,
(trophyImage.get_width()/5.5, trophyImage.get_height()/5.5))
    flappyBirdThumbnailImageMod =
pygame.transform.scale(flappyBirdThumbnailImage,
(flappyBirdThumbnailImage.get_width()/3.5,
flappyBirdThumbnailImage.get_height()/3.5))
    pongThumbnailImageMod =
pygame.transform.scale(pongThumbnailImage,
(pongThumbnailImage.get_width()/3.5,
pongThumbnailImage.get_height()/3.5))
    snakeThumbnailImageMod =
pygame.transform.scale(snakeThumbnailImage,
(snakeThumbnailImage.get_width()/3.5,
snakeThumbnailImage.get_height()/3.5))
```

```
    endlessRunnerThumbnailImageMod =
pygame.transform.scale(endlessRunnerThumbnailImage,
(endlessRunnerThumbnailImage.get_width()/3.5,
endlessRunnerThumbnailImage.get_height()/3.5))
    tictactoeThumbnailImageMod =
pygame.transform.scale(tictactoeThumbnailImage,
(tictactoeThumbnailImage.get_width()/3.5,
tictactoeThumbnailImage.get_height()/3.5))
    rpsThumbnailImageMod =
pygame.transform.scale(rpsThumbnailImage,
(rpsThumbnailImage.get_width()/3.5,
rpsThumbnailImage.get_height()/3.5))

    #rect
    trophyRect = pygame.Rect(1150, yOffset+50,
trophyImageMod.get_width(), trophyImageMod.get_height())

    #decoy rect
    flappyBirdThumbnailRect = pygame.Rect(-25, -25, 1, 1)
    pongThumbnailRect = pygame.Rect(-25, -25, 1, 1)
    endlessRunnerThumbnailRect = pygame.Rect(-25, -25, 1, 1)
    snakeThumbnailRect = pygame.Rect(-25, -25, 1, 1)
    tictactoeThumbnailRect = pygame.Rect(-25, -25, 1, 1)
    rpsThumbnailRect = pygame.Rect(-25, -25, 1, 1)

    #texts
    arcadeTitle = regularFont.render('The Arcade', False,
(140, 3, 252))
    flappyBirdText = nameFont.render('Flappy Bird', False,
(140, 3, 252))
    pongText = nameFont.render('Pong', False, (140, 3, 252))
    endlessRunnerText = nameFont.render('Endless Runner',
False, (140, 3, 252))
    snakeText = nameFont.render('Snake', False, (140, 3, 252))
    rpsText = nameFont.render('Rock-Paper-Scissors', False,
(140, 3, 252))
    tictactoeText = nameFont.render('Tic-Tac-Toe', False,
(140, 3, 252))

    #function to draw thumbnails
    def drawThumbnail(thumbnailImageMod, thumbnailRect, text,
offset, screen, posx, posy):
        thumbnailRect = pygame.Rect(posx+182.5-
(thumbnailImageMod.get_width()/2), posy+offset+102.5-
```

```
(thumbnailImageMod.get_height()/2),
thumbnailImageMod.get_width(), thumbnailImageMod.get_height())
        thumbnail = screen.blit(thumbnailImageMod,
thumbnailRect)
        textRect = text.get_rect(center = (thumbnail.centerx,
posy+240+offset))
        screen.blit(text, textRect)
        if thumbnailRect != None: return thumbnailRect

    #game loop
    while True:
        #text rect
        arcadeTitleRect = arcadeTitle.get_rect(center = (640,
100+yOffset))

        #event checker
        for event in pygame.event.get():
            #quit on clicking close window button
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()

            #back to main menu
            if event.type == pygame.KEYDOWN and event.key ==
pygame.K_ESCAPE:
                mainMenu()

            #trophy button
            if event.type == pygame.MOUSEBUTTONDOWN and
trophyRect.collidepoint(event.pos):
                scoreboardScreen()

            #scroll bar functioning
            try:
                if event.type == pygame.MOUSEBUTTONDOWN :
                    if scrollBar.collidepoint(event.pos):
                        scrolling = True
                if pygame.mouse.get_pressed()[0] and scrolling
== True:
                    scrollBarPos = pygame.mouse.get_pos()[1] -
50
                    yOffset = 200 - scrollBarPos
                if event.type == pygame.MOUSEBUTTONUP:
                    scrolling = False
            except:
```

```
                pass
            #loading the games on clicking thumbnail
            if event.type == pygame.MOUSEBUTTONDOWN and
flappyBirdThumbnailRect.collidepoint(event.pos): flappyBird()
            if event.type == pygame.MOUSEBUTTONDOWN and
pongThumbnailRect.collidepoint(event.pos): pong()
            if event.type == pygame.MOUSEBUTTONDOWN and
rpsThumbnailRect.collidepoint(event.pos): rockPaperScissors()
            if event.type == pygame.MOUSEBUTTONDOWN and
endlessRunnerThumbnailRect.collidepoint(event.pos):
endlessRunner()
            if event.type == pygame.MOUSEBUTTONDOWN and
snakeThumbnailRect.collidepoint(event.pos): snake()
            if event.type == pygame.MOUSEBUTTONDOWN and
tictactoeThumbnailRect.collidepoint(event.pos): tictactoe()

        #display bg
        screen.blit(mainBgSurface, (0, 0))

        #limiting scrolling
        if scrollBarPos < 200: scrollBarPos = 200
        if scrollBarPos > 550: scrollBarPos = 550
        if yOffset > 0: yOffset = 0
        if yOffset < -350: yOffset = -350

        #scroll bar
        scrollBar = pygame.draw.rect(screen, (140, 3, 252),
pygame.Rect(610, scrollBarPos, 60, 100), 0)

        #flappy bird thumbnail
        if
flappyBirdThumbnailRect.collidepoint(pygame.mouse.get_pos()):
            flappyBirdThumbnailImageMod =
pygame.transform.scale(flappyBirdThumbnailImage,
(flappyBirdThumbnailImage.get_width()/3.5,
flappyBirdThumbnailImage.get_height()/3.5))
        else:
            flappyBirdThumbnailImageMod =
pygame.transform.scale(flappyBirdThumbnailImage,
(flappyBirdThumbnailImage.get_width()/4,
flappyBirdThumbnailImage.get_height()/4))
        flappyBirdThumbnailRect =
drawThumbnail(flappyBirdThumbnailImageMod,
flappyBirdThumbnailRect, flappyBirdText, yOffset, screen, 100,
190)
```

```
        #pong thumbnail
        if
pongThumbnailRect.collidepoint(pygame.mouse.get_pos()):
            pongThumbnailImageMod =
pygame.transform.scale(pongThumbnailImage,
(pongThumbnailImage.get_width()/3.5,
pongThumbnailImage.get_height()/3.5))
        else:
            pongThumbnailImageMod =
pygame.transform.scale(pongThumbnailImage,
(pongThumbnailImage.get_width()/4,
pongThumbnailImage.get_height()/4))
        pongThumbnailRect =
drawThumbnail(pongThumbnailImageMod, pongThumbnailRect,
pongText, yOffset, screen, 100, 500)

        #endless runner thumbnail
        if
endlessRunnerThumbnailRect.collidepoint(pygame.mouse.get_pos()
):
            endlessRunnerThumbnailImageMod =
pygame.transform.scale(endlessRunnerThumbnailImage,
(endlessRunnerThumbnailImage.get_width()/3.5,
endlessRunnerThumbnailImage.get_height()/3.5))
        else:
            endlessRunnerThumbnailImageMod =
pygame.transform.scale(endlessRunnerThumbnailImage,
(endlessRunnerThumbnailImage.get_width()/4,
endlessRunnerThumbnailImage.get_height()/4))
        endlessRunnerThumbnailRect =
drawThumbnail(endlessRunnerThumbnailImageMod,
endlessRunnerThumbnailRect, endlessRunnerText, yOffset,
screen, 810, 190)

        #snake thumbnail
        if
snakeThumbnailRect.collidepoint(pygame.mouse.get_pos()):
            snakeThumbnailImageMod =
pygame.transform.scale(snakeThumbnailImage,
(snakeThumbnailImage.get_width()/3.5,
snakeThumbnailImage.get_height()/3.5))
        else:
            snakeThumbnailImageMod =
pygame.transform.scale(snakeThumbnailImage,
```

```
(snakeThumbnailImage.get_width()/4,
snakeThumbnailImage.get_height()/4))
        snakeThumbnailRect =
drawThumbnail(snakeThumbnailImageMod, snakeThumbnailRect,
snakeText, yOffset, screen, 810, 500)

        #tictactoe thumbnail
        if
tictactoeThumbnailRect.collidepoint(pygame.mouse.get_pos()):
            tictactoeThumbnailImageMod =
pygame.transform.scale(tictactoeThumbnailImage,
(tictactoeThumbnailImage.get_width()/3.5,
tictactoeThumbnailImage.get_height()/3.5))
        else:
            tictactoeThumbnailImageMod =
pygame.transform.scale(tictactoeThumbnailImage,
(tictactoeThumbnailImage.get_width()/4,
tictactoeThumbnailImage.get_height()/4))
        tictactoeThumbnailRect =
drawThumbnail(tictactoeThumbnailImageMod,
tictactoeThumbnailRect, tictactoeText, yOffset, screen, 810,
800)

        #rps thumbnail
        if
rpsThumbnailRect.collidepoint(pygame.mouse.get_pos()):
            rpsThumbnailImageMod =
pygame.transform.scale(rpsThumbnailImage,
(rpsThumbnailImage.get_width()/3.5,
rpsThumbnailImage.get_height()/3.5))
        else:
            rpsThumbnailImageMod =
pygame.transform.scale(rpsThumbnailImage,
(rpsThumbnailImage.get_width()/4,
rpsThumbnailImage.get_height()/4))
        rpsThumbnailRect = drawThumbnail(rpsThumbnailImageMod,
rpsThumbnailRect, rpsText, yOffset, screen, 100, 800)

        #trophy
        trophyRect = pygame.Rect(1150, yOffset+50,
trophyImageMod.get_width(), trophyImageMod.get_height())
        screen.blit(trophyImageMod, trophyRect)

        #text
        screen.blit(arcadeTitle, arcadeTitleRect)
```

```
        #update frames and cap framerate
        pygame.display.update()
        clock.tick(60)

#Show highscorers
def scoreboardScreen():
    #getting the data
    with open(f'{directory}/scores.json', 'r') as scoreFile:
        scoreData = json.load(scoreFile)
    fbUser, fbScore = scoreData["FlappyBird"]["user"],
scoreData["FlappyBird"]["score"]
    erUser, erScore = scoreData["EndlessRunner"]["user"],
scoreData["EndlessRunner"]["score"]
    sUser, sScore = scoreData["Snake"]["user"],
scoreData["Snake"]["score"]

    #font
    regularFont =
pygame.font.Font(f'{directory}/font/dogica.ttf', 60)
    titleFont =
pygame.font.Font(f'{directory}/font/dogica.ttf', 25)
    tableFont =
pygame.font.Font(f'{directory}/font/dogica.ttf', 35)

    #image
    crownImage =
pygame.image.load(f'{directory}/media/crown.png').convert_alph
a()
    crownImageMod = pygame.transform.scale(crownImage,
(crownImage.get_width()/5.5, crownImage.get_height()/5.5))
    crownoneRect = pygame.Rect(980, 35,
crownImageMod.get_width(), crownImageMod.get_height())
    crowntwoRect = pygame.Rect(200, 35,
crownImageMod.get_width(), crownImageMod.get_height())

    #objects
    mainBgSurface =
pygame.image.load(f'{directory}/media/mainMenuBG.png').convert
()

    #texts
    arcadeTitle = regularFont.render('Leaderboard', False,
(140, 3, 252))
    flappybirdTitle = titleFont.render('Flappy Bird -', False,
(140, 3, 252))
```

```
    endlessrunnerTitle = titleFont.render('Endless Runner -',
False, (140, 3, 252))
    snakeTitle = titleFont.render('Snake -', False, (140, 3,
252))
    gameText = tableFont.render('GAME', False, (140, 3, 252))
    leaderText = tableFont.render('LEADER', False, (140, 3,
252))
    flappybirdScoreText =
titleFont.render(f'{fbUser}({fbScore})', False, (140, 3, 252))
    endlessrunnerScoreText =
titleFont.render(f'{erUser}({erScore})', False, (140, 3, 252))
    snakeScoreText = titleFont.render(f'{sUser}({sScore})',
False, (140, 3, 252))

    #game loop
    while True:
        #event checker
        for event in pygame.event.get():
            #quit on clicking close window button
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
            #back to games screen
            if event.type == pygame.KEYDOWN and event.key ==
pygame.K_ESCAPE:
                gamesScreen()

        #display bg
        screen.blit(mainBgSurface, (0, 0))

        #main title text
        arcadeTitleRect = arcadeTitle.get_rect(center = (640,
80))
        screen.blit(arcadeTitle, arcadeTitleRect)

        #displaying crowns
        screen.blit(crownImageMod, crownoneRect)
        screen.blit(crownImageMod, crowntwoRect)

        #displaying titles
        flappybirdTitleRect =
flappybirdTitle.get_rect(topright = (625, 350))
        endlessrunnerTitleRect =
endlessrunnerTitle.get_rect(topright = (620, 450))
```

```
        snakeTitleRect = snakeTitle.get_rect(topright = (620,
550))
        gameTextRect = gameText.get_rect(topright = (540,
250))
        leaderTextRect = leaderText.get_rect(topleft = (740,
250))
        screen.blit(flappybirdTitle, flappybirdTitleRect)
        screen.blit(endlessrunnerTitle,
endlessrunnerTitleRect)
        screen.blit(snakeTitle, snakeTitleRect)
        screen.blit(gameText, gameTextRect)
        screen.blit(leaderText, leaderTextRect)

        #displaying scores
        flappybirdScoreTextRect =
flappybirdScoreText.get_rect(topleft = (715, 350))
        endlessrunnerScoreTextRect =
endlessrunnerScoreText.get_rect(topleft = (715, 450))
        snakeScoreTextRect = snakeScoreText.get_rect(topleft =
(715, 550))
        screen.blit(flappybirdScoreText,
flappybirdScoreTextRect)
        screen.blit(endlessrunnerScoreText,
endlessrunnerScoreTextRect)
        screen.blit(snakeScoreText, snakeScoreTextRect)

        #update frames
        pygame.display.update()
        clock.tick(60)

#starts flappy bird
def flappyBird():
    #helper variables
    started, gameOver = False, False
    idleAnim = True
    downTubes, upTubes = [], []
    gravity = 0
    tubePos = [-460]
    score = 0
    restartButton = pygame.draw.rect(screen, (84, 48, 0),
pygame.Rect(-500, -500, 280, 110), 0)
    exitButton = pygame.draw.rect(screen, (84, 48, 0),
pygame.Rect(-500, -500, 280, 110), 0)

    #fonts
```

```
    scoreFont = pygame.font.Font(f'{directory}/font/flappy-
bird-font.ttf', 100)
    textFont =
pygame.font.Font(f'{directory}/font/FlappyBirdy.ttf', 125)
    buttonFont =
pygame.font.Font(f'{directory}/font/dogicapixel.ttf', 35)
    scorePanelFont =
pygame.font.Font(f'{directory}/font/dogicapixel.ttf', 25)
    finalDetailsFont =
pygame.font.Font(f'{directory}/font/dogicapixel.ttf', 50)

    #objects
    mainBgSurface =
pygame.image.load(f'{directory}/media/flappyBird/flappyBirdBG.
png').convert()
    groundSurface =
pygame.image.load(f'{directory}/media/flappyBird/flappyBirdGro
und.png').convert()
    groundSurface2 =
pygame.image.load(f'{directory}/media/flappyBird/flappyBirdGro
und2.png').convert()
    tubeDown =
pygame.image.load(f'{directory}/media/flappyBird/tubeDown.png'
).convert_alpha()
    tubeUp =
pygame.image.load(f'{directory}/media/flappyBird/tubeUp.png').
convert_alpha()
    playerBird =
pygame.image.load(f'{directory}/media/flappyBird/playerBirdMid
dle.png').convert_alpha()
    scorePanel =
pygame.image.load(f'{directory}/media/flappyBird/scorePanel.pn
g').convert_alpha()

    #text
    scoreText = scoreFont.render(f'{score}', False, (0, 0, 0))
    scoreTextRect = scoreText.get_rect(center = (640, 125))
    startMessage = textFont.render('Click to start', False,
(0, 0, 0))
    startMessageRect = startMessage.get_rect(center = (640,
250))
    restart = buttonFont.render('RESTART', False, '#ffffff')
    restartRect = restart.get_rect(center = (390, 432.5))
    back = buttonFont.render('EXIT', False, '#ffffff')
    backRect = restart.get_rect(center = (940, 432.5))
```

```
    finalScoreText = scorePanelFont.render('SCORE', False,
(232, 97, 1))
    finalScoreTextRect = finalScoreText.get_rect(center =
(640, 110))
    bestScoreText = scorePanelFont.render('BEST', False, (232,
97, 1))
    bestScoreTextRect = bestScoreText.get_rect(center = (640,
220))

    #scaling
    mainBgSurface = pygame.transform.scale(mainBgSurface,
(mainBgSurface.get_width()*10, mainBgSurface.get_height()*10))
    groundSurface = pygame.transform.scale(groundSurface,
(groundSurface.get_width()*10, groundSurface.get_height()*10))
    groundSurface2 = pygame.transform.scale(groundSurface2,
(groundSurface2.get_width()*10,
groundSurface2.get_height()*10))
    tubeDown = pygame.transform.scale(tubeDown,
(tubeDown.get_width()*10, tubeDown.get_height()*10))
    tubeUp = pygame.transform.scale(tubeUp,
(tubeUp.get_width()*10, tubeUp.get_height()*10))
    playerBird = pygame.transform.scale(playerBird,
((playerBird.get_width()*4.5), playerBird.get_height()*4.5))
    scorePanel = pygame.transform.scale(scorePanel,
(scorePanel.get_width()*10, scorePanel.get_height()*10))

    #multiple bgs for cycling
    g1 = 0
    g2 = groundSurface.get_width()

    #object rects
    playerBirdRect = playerBird.get_rect(center = (320, 360))
    for pos in tubePos:
        downTubes.append(tubeDown.get_rect(topleft = (1500,
pos)))
        upTubes.append(tubeUp.get_rect(topleft = (1500,
pos+940)))
    scorePanelRect = scorePanel.get_rect(center = (640, 200))

    #function to add new tube
    def tubeAdder():
        newPos = random.randint(-610, -409)
        tubePos.append(newPos)
        downTubes.append(tubeDown.get_rect(topleft = (1500,
newPos)))
```

```
        upTubes.append(tubeUp.get_rect(topleft = (1500,
newPos+940)))

    #function to display score
    def updateScore(score):
        scoreText = scoreFont.render(f'{score}', False, (0, 0,
0))
        screen.blit(scoreText, scoreTextRect)

    #displaying final results on game over screen
    def scorePanelUpdate(score):
        with open(f'{directory}/scores.json', 'r') as
scoreFileRead:
            scoreData = json.load(scoreFileRead)
        highScore = scoreData["FlappyBird"]["score"]
        finalScore = finalDetailsFont.render(f'{score}',
False, (0, 0, 0))
        finalScoreRect = finalScore.get_rect(center = (640,
155))
        bestScore = finalDetailsFont.render(f'{highScore}',
False, (0, 0, 0))
        bestScoreRect = bestScore.get_rect(center = (640,
265))
        screen.blit(finalScore, finalScoreRect)
        screen.blit(bestScore, bestScoreRect)

    #checks to update highscore
    def highScoreUpdate(score):
        with open(f'{directory}/scores.json', 'r') as
scoreFileRead:
            scoreData = json.load(scoreFileRead)
        with open(f'{directory}/scores.json', 'w') as
scoreFileWrite:
            if scoreData['FlappyBird']['score'] < int(score):
                updateFirstDatabase(1, 'FlappyBird',
int(score), username, scoreData)
                scoreData['FlappyBird']['score'] = score
                scoreData['FlappyBird']['user'] = username
            json.dump(scoreData, scoreFileWrite)
        return score

    #game loop
    while True:
        #event checker
        for event in pygame.event.get():
```

```
            #quit on clicking close window button
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
            #starting game on mouseclick
            if event.type == pygame.MOUSEBUTTONDOWN or
event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
                if started == False:
                    started = True
                    gameOver = False
            #jumping
            if event.type == pygame.MOUSEBUTTONDOWN or
event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
                if started == True and gameOver == False:
gravity = -18
            #restarting
            if event.type == pygame.MOUSEBUTTONDOWN and
restartButton.collidepoint(event.pos):
                if started == True and gameOver == True:
                    started, gameOver = False, False
                    gravity, score = 0, 0
                    idleAnim = True
                    downTubes, upTubes = [], []
                    playerBirdRect.x, playerBirdRect.y = 282,
360
                    tubePos = [-460]
                    for pos in tubePos:

downTubes.append(tubeDown.get_rect(topleft = (1500, pos)))
                        upTubes.append(tubeUp.get_rect(topleft
= (1500, pos+940)))
            #exit game with button
            if event.type == pygame.MOUSEBUTTONDOWN and
exitButton.collidepoint(event.pos):
                gamesScreen()
            #exit game with escape
            if not started and not gameOver:
                if event.type == pygame.KEYDOWN and event.key
== pygame.K_ESCAPE:
                    gamesScreen()

        #idle animation before game starts
        if started == False and gameOver == False:
                        if idleAnim == True:  playerBirdRect.y
-= 1
```

```
            if idleAnim == False: playerBirdRect.y += 1
            if playerBirdRect.y < 350:
                playerBirdRect.y = 350
                idleAnim = False
            if playerBirdRect.y > 370:
                playerBirdRect.y = 370
                idleAnim = True

        #background
        screen.blit(mainBgSurface, (0, 0))

        #started game
        if started == True:
            for pos in tubePos:
                #getting index number of pos in tubePos to
find respective tubes in downTubes and upTubes
                commonIndex = tubePos.index(pos)
                #deleting tubes that are out of the screen
                if downTubes[commonIndex].x < -200:
                    del tubePos[commonIndex],
downTubes[commonIndex], upTubes[commonIndex]
                #displaying tubes
                screen.blit(tubeDown, downTubes[commonIndex])
                screen.blit(tubeUp, upTubes[commonIndex])
        if started == True and gameOver == False:
            #creating new tube on old tube reaching sufficient
distance
            if downTubes[-1].x <= 805: tubeAdder()
            #moving tube every frame
            for downTube in downTubes: downTube.x -= 5
            for upTube in upTubes: upTube.x -= 5
            for tube in downTubes:
                if tube.center[0] == 320:
                    score += 1
        #displaying score
        if not gameOver:
            updateScore(score)

        #ending game
        if started == True:
            for downTube in downTubes:
                if downTube.colliderect(playerBirdRect) or
playerBirdRect.bottom >= 620:
                    started = True
                    gameOver = True
```

```
            for upTube in upTubes:
                if upTube.colliderect(playerBirdRect) or
playerBirdRect.bottom >= 620:
                    started = True
                    gameOver = True

        #calling once to prevent overlapping
        restartButton = pygame.draw.rect(screen, (84, 48, 0),
pygame.Rect(-500, -500, 280, 110), 0)
        exitButton = pygame.draw.rect(screen, (84, 48, 0),
pygame.Rect(-500, -500, 280, 110), 0)

        #game over screen
        if gameOver and started:
            #score panel
            screen.blit(scorePanel, scorePanelRect)
            screen.blit(finalScoreText, finalScoreTextRect)
            screen.blit(bestScoreText, bestScoreTextRect)
            score = highScoreUpdate(score)
            scorePanelUpdate(score)
            #restart button
            restartButton = pygame.draw.rect(screen, (84, 48,
0), pygame.Rect(250, 380, 280, 110), 0)
            pygame.draw.rect(screen, '#ffffff',
pygame.Rect(257.5, 387.5, 265, 90), 0)
            pygame.draw.rect(screen, (232, 97, 1),
pygame.Rect(265, 397.5, 250, 70), 0)
            screen.blit(restart, restartRect)
            #exit button
            exitButton = pygame.draw.rect(screen, (84, 48, 0),
pygame.Rect(750, 380, 280, 110), 0)
            pygame.draw.rect(screen, '#ffffff',
pygame.Rect(757.5, 387.5, 265, 90), 0)
            pygame.draw.rect(screen, (232, 97, 1),
pygame.Rect(765, 397.5, 250, 70), 0)
            screen.blit(back, backRect)

        #ground
        if not gameOver:
            g1 -= 5
            g2 -= 5
            if g1 <= -1280: g1 = 1280
            if g2 <= -1280: g2 = 1280
        screen.blit(groundSurface, (g1, 720-
groundSurface.get_height()))
```

```
        screen.blit(groundSurface2, (g2-10, 720-
groundSurface.get_height()))

        #text
        if not started:
            screen.blit(startMessage, startMessageRect)

        #player
        screen.blit(playerBird, playerBirdRect)

        #falling when game over
        if started == True and gameOver == True:
            gravity = 15

        #increasing gravity every frame
        if started == True and gameOver == False:
            gravity += 1.35
        playerBirdRect.y += gravity

        #y level limitations to player
        if playerBirdRect.bottom >= 620: playerBirdRect.bottom
= 620
        if playerBirdRect.bottom <= -30: playerBirdRect.bottom
= -30

        #update frame and cap framerate
        pygame.display.update()
        clock.tick(60)

#starts pong
def pong():
    #variables
    isSingle = True
    player2Score, player1Score = 0, 0
    player1Pos, player2Pos = 0, 0
    player1MoveUp, player1MoveDown = False, False
    player2MoveUp, player2MoveDown = False, False
    ballSpeedX, ballSpeedY = 6, random.randint(6, 8)
    ball = pygame.draw.rect(screen, '#dcdcdc',
pygame.Rect(625, 345, 25, 25))
    started, gameOver = False, False
    color1, color2 = 'white', 'white'

    #font
```

```
    scoreFont = pygame.font.Font(f'{directory}/font/pong-
score.ttf', 115)
    titleFont =
pygame.font.Font(f'{directory}/font/slkscreb.ttf', 200)
    optionsFont =
pygame.font.Font(f'{directory}/font/pzim3x5b.ttf', 50)
    winFont =
pygame.font.Font(f'{directory}/font/dogicabold.ttf', 50)
    finalFont =
pygame.font.Font(f'{directory}/font/dogicabold.ttf', 20)

    #text
    titleText = titleFont.render('pong', False, '#ffffff')
    titleTextRect = titleText.get_rect(center = (640, 250))
    singlePlayerText = optionsFont.render('Single Player',
False, '#ffffff')
    singlePlayerTextHover = optionsFont.render('Single
Player', False, (0, 0, 0))
    singlePlayerTextRect = singlePlayerText.get_rect(center =
(380, 475))
    multiPlayerText = optionsFont.render('Multi Player',
False, '#ffffff')
    multiPlayerTextHover = optionsFont.render('Multi Player',
False, (0, 0, 0))
    multiPlayerTextRect = multiPlayerText.get_rect(center =
(900, 475))

    #function to display score
    def updateScore(player1, player2):
        player1Display = scoreFont.render(f'{player1}', False,
'#dcdcdc')
        player1Rect = player1Display.get_rect(topright = (595,
100))
        screen.blit(player1Display, player1Rect)
        player2Display = scoreFont.render(f'{player2}', False,
'#dcdcdc')
        player2Rect = player2Display.get_rect(topleft = (740,
100))
        screen.blit(player2Display, player2Rect)

    #function to display win message and final options
    def finalScoreDisplay(score1, score2, playAgainColor,
titleScreenColor):
        if score2 == 3: xPos = 340
        elif score1 == 3: xPos = 930
```

```
        if playAgainColor == 'black': playAgainSecColor = 0
        elif playAgainColor == 'white': playAgainSecColor = 3
        if titleScreenColor == 'black': titleScreenSecColor =
0
        elif titleScreenColor == 'white': titleScreenSecColor
= 3
        finalScoreText = winFont.render(f'Wins!', False,
'#ffffff')
        finalScoreTextRect = finalScoreText.get_rect(center =
(xPos, 350))
        playAgainText = finalFont.render('Play Again', False,
playAgainColor)
        playAgainTextRect = playAgainText.get_rect(center =
(xPos-135, 500))
        titleScreenText = finalFont.render('Main Screen',
False, titleScreenColor)
        titleScreenTextRect = titleScreenText.get_rect(center
= (xPos+135, 500))
        mainRect = pygame.draw.rect(screen, '#ffffff',
pygame.Rect(playAgainTextRect.left-15, finalScoreTextRect.top-
25, titleScreenTextRect.right-playAgainTextRect.left+30,
titleScreenTextRect.bottom-finalScoreTextRect.top+50), 3)
        if playAgainSecColor == 3:
            pygame.draw.rect(screen, '#000000',
pygame.Rect(playAgainTextRect.left-15,
titleScreenTextRect.top-25, mainRect.width/2,
titleScreenTextRect.height+50), 0)
        playAgainButton = pygame.draw.rect(screen, '#ffffff',
pygame.Rect(playAgainTextRect.left-15,
titleScreenTextRect.top-25, mainRect.width/2,
titleScreenTextRect.height+50), playAgainSecColor)
        if titleScreenSecColor == 3:
            pygame.draw.rect(screen, '#000000',
pygame.Rect(mainRect.left+(mainRect.width/2),
titleScreenTextRect.top-25, mainRect.width/2,
titleScreenTextRect.height+50), 0)
        titleScreenButton = pygame.draw.rect(screen,
'#ffffff', pygame.Rect(mainRect.left+(mainRect.width/2),
titleScreenTextRect.top-25, mainRect.width/2,
titleScreenTextRect.height+50), titleScreenSecColor)
        screen.blit(finalScoreText, finalScoreTextRect)
        screen.blit(playAgainText, playAgainTextRect)
        screen.blit(titleScreenText, titleScreenTextRect)
        return playAgainButton, titleScreenButton

    #game loop
```

```
    while True:
        #event checker
        for event in pygame.event.get():
            #quit on clicking close window button
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
            #menu events
            if not started and not gameOver:
                #loading game on button click
                if event.type == pygame.MOUSEBUTTONDOWN:
                    if
singlePlayerButton.collidepoint(event.pos) or
multiPlayerButton.collidepoint(event.pos): started, gameOver =
True, False
                    if
singlePlayerButton.collidepoint(event.pos): isSingle = True
                    elif
multiPlayerButton.collidepoint(event.pos): isSingle = False
            #events when game starts
            if started and not gameOver:
                #paddle movement for player 1
                if event.type == pygame.KEYDOWN and event.key
== pygame.K_UP: player1MoveUp, player1MoveDown = True, False
                if event.type == pygame.KEYDOWN and event.key
== pygame.K_DOWN: player1MoveUp, player1MoveDown = False, True
                if event.type == pygame.KEYUP and event.key ==
pygame.K_DOWN: player1MoveDown = False
                if event.type == pygame.KEYUP and event.key ==
pygame.K_UP: player1MoveUp = False
                if not isSingle:
                    #paddle movement for player 2
                    if event.type == pygame.KEYDOWN and
event.key == pygame.K_w: player2MoveUp, player2MoveDown =
True, False
                    if event.type == pygame.KEYDOWN and
event.key == pygame.K_s: player2MoveUp, player2MoveDown =
False, True
                    if event.type == pygame.KEYUP and
event.key == pygame.K_s: player2MoveDown = False
                    if event.type == pygame.KEYUP and
event.key == pygame.K_w: player2MoveUp = False
            #escape key uses
            if event.type == pygame.KEYDOWN and event.key ==
pygame.K_ESCAPE:
```

```
                player2Score, player1Score = 0, 0
                player2Pos, player1Pos = 0, 0
                player1MoveUp, player1MoveDown = False, False
                player2MoveUp, player2MoveDown = False, False
                ballSpeedX, ballSpeedY = 6, random.randint(6,
8)
                ball = pygame.draw.rect(screen, '#dcdcdc',
pygame.Rect(625, 345, 25, 25))
                if not started and not gameOver: gamesScreen()
                elif started and not gameOver or started and
gameOver: started, gameOver = False, False
            #events when game over
            if started and gameOver:
                #common button functions
                if event.type == pygame.MOUSEBUTTONDOWN:
                    if playAgainButton.collidepoint(event.pos)
or titleScreenButton.collidepoint(event.pos):
                        player2Score, player1Score = 0, 0
                        player1Pos, player2Pos = 0, 0
                        player1MoveUp, player1MoveDown =
False, False
                        player2MoveUp, player2MoveDown =
False, False
                        ballSpeedX, ballSpeedY = 6,
random.randint(6, 8)
                        ball = pygame.draw.rect(screen,
'#dcdcdc', pygame.Rect(625, 345, 25, 25))
                #restarting
                if event.type == pygame.MOUSEBUTTONDOWN and
playAgainButton.collidepoint(event.pos): started, gameOver =
True, False
                #back to menu
                if event.type == pygame.MOUSEBUTTONDOWN and
titleScreenButton.collidepoint(event.pos): started, gameOver =
False, False

        #code block for when game starts
        if started and not gameOver:
            #background
            screen.fill('#dcdcdc')
            pygame.draw.rect(screen, 'black', pygame.Rect(15,
15, 1250, 690))

            #moving player 1
            if player1MoveUp == True:
```

```
                if player1Paddle.top > 20: player1Pos -= 9.5
            elif player1MoveDown == True:
                if player1Paddle.bottom < 700: player1Pos +=
9.5
            #moving player2
            if player2MoveUp == True:
                if player2Paddle.top > 20: player2Pos -= 9.5
            elif player2MoveDown == True:
                if player2Paddle.bottom < 700: player2Pos +=
9.5

            #player paddles
            player2Paddle = pygame.draw.rect(screen,
'#dcdcdc', pygame.Rect(50, 310+player2Pos, 25, 100))
            player1Paddle = pygame.draw.rect(screen,
'#dcdcdc', pygame.Rect(1205, 310+player1Pos, 25, 100))

            #centre split
            for i in range(30, 660, 60):
                pygame.draw.rect(screen, '#dcdcdc',
pygame.Rect(630, 15+i, 20, 30))

            #moving ball
            ballPosX = ball.x
            ballPosY = ball.y
            ball = pygame.draw.rect(screen, '#dcdcdc',
pygame.Rect(ballPosX+ballSpeedX, ballPosY+ballSpeedY, 25, 25))

            #ball physics
            if ball.bottom >= 705 or ball.top <= 15:
ballSpeedY *= -1
            if player1Paddle.colliderect(ball) or
player2Paddle.colliderect(ball):
                if ballSpeedX < 0: ballSpeedX =
random.randint(-12, -9)
                elif ballSpeedX > 0: ballSpeedX =
random.randint(9, 12)
                if ballSpeedY < 0: ballSpeedY =
random.randint(-10, -4)
                elif ballSpeedY >= 0: ballSpeedY =
random.randint(4, 10)
            if player1Paddle.colliderect(ball) or
player2Paddle.colliderect(ball): ballSpeedX *= -1
            if player1Paddle.colliderect(ball): ball.right =
player1Paddle.left
```

```
            if player2Paddle.colliderect(ball): ball.left =
player2Paddle.right

            #singleplayer AI
            if isSingle:
                if ball.bottom < player2Paddle.top+8:
player2Pos -= 9.5
                elif ball.top > player2Paddle.bottom-8:
player2Pos += 9.5

            #scoring
            if ball.right >= 1280: player2Score += 1
            if ball.left <= 0: player1Score += 1
            if ball.right >= 1280 or ball.left <= 0:
                ballSpeedY = random.randint(-8, 8)
                if ballSpeedX < 0: ballSpeedX = 6
                elif ballSpeedX > 0: ballSpeedX = -6
                ball = pygame.draw.rect(screen, '#dcdcdc',
pygame.Rect(625, 345, 25, 25))

            #display score
            updateScore(player2Score, player1Score)

            #ending game on victory
            if player1Score == 3 or player2Score == 3:
started, gameOver = True, True

        #code block for menu screens
        elif not started and not gameOver:
            #background
            screen.fill('#dcdcdc')
            pygame.draw.rect(screen, 'black', pygame.Rect(15,
15, 1250, 690))
            #title text
            screen.blit(titleText, titleTextRect)

            #buttons
            singlePlayerButton = pygame.draw.rect(screen,
'#ffffff', pygame.Rect(singlePlayerTextRect.left-15,
singlePlayerTextRect.top-12, singlePlayerText.get_width()+20,
singlePlayerText.get_height()+20), 3)
            multiPlayerButton = pygame.draw.rect(screen,
'#ffffff', pygame.Rect(multiPlayerTextRect.left-15,
multiPlayerTextRect.top-12, multiPlayerText.get_width()+20,
multiPlayerText.get_height()+20), 3)
```

```
            if
singlePlayerButton.collidepoint(pygame.mouse.get_pos()):
                singlePlayerButton = pygame.draw.rect(screen,
'#ffffff', pygame.Rect(singlePlayerTextRect.left-15,
singlePlayerTextRect.top-12, singlePlayerText.get_width()+20,
singlePlayerText.get_height()+20))
                screen.blit(singlePlayerTextHover,
singlePlayerTextRect)
            else:
                screen.blit(singlePlayerText,
singlePlayerTextRect)
            if
multiPlayerButton.collidepoint(pygame.mouse.get_pos()):
                multiPlayerButton = pygame.draw.rect(screen,
'#ffffff', pygame.Rect(multiPlayerTextRect.left-15,
multiPlayerTextRect.top-12, multiPlayerText.get_width()+20,
multiPlayerText.get_height()+20))
                screen.blit(multiPlayerTextHover,
multiPlayerTextRect)
            else:
                screen.blit(multiPlayerText,
multiPlayerTextRect)

            #decor paddles
            pygame.draw.rect(screen, '#ffffff',
pygame.Rect(50, 310, 25, 100))
            pygame.draw.rect(screen, '#ffffff',
pygame.Rect(1205, 310, 25, 100))

        #code block for game over screen
        elif started and gameOver:
            playAgainButton, titleScreenButton =
finalScoreDisplay(player1Score, player2Score, color1, color2)
            if
playAgainButton.collidepoint(pygame.mouse.get_pos()): color1 =
'black'
            else: color1 = 'white'
            if
titleScreenButton.collidepoint(pygame.mouse.get_pos()): color2
= 'black'
            else: color2 = 'white'

        #update frame and cap framerate
        pygame.display.update()
        clock.tick(60)
```

```
#starts snake
def snake():
    isGameOver = False
    restartPressed = True
    dsConstant = 0
    moveConstant = 0

    class SNAKE:
        def __init__(self):
            self.body =
[Vector2(5,10),Vector2(4,10),Vector2(3,10)]
            self.direction = Vector2(0,0)
            self.new_block = False

            self.head_up =
pygame.image.load(f'{directory}/media/snake/head_up.png').conv
ert_alpha()
            self.head_down =
pygame.image.load(f'{directory}/media/snake/head_down.png').co
nvert_alpha()
            self.head_right =
pygame.image.load(f'{directory}/media/snake/head_right.png').c
onvert_alpha()
            self.head_left =
pygame.image.load(f'{directory}/media/snake/head_left.png').co
nvert_alpha()

            self.tail_up =
pygame.image.load(f'{directory}/media/snake/tail_up.png').conv
ert_alpha()
            self.tail_down =
pygame.image.load(f'{directory}/media/snake/tail_down.png').co
nvert_alpha()
            self.tail_right =
pygame.image.load(f'{directory}/media/snake/tail_right.png').c
onvert_alpha()
            self.tail_left =
pygame.image.load(f'{directory}/media/snake/tail_left.png').co
nvert_alpha()

            self.body_vertical =
pygame.image.load(f'{directory}/media/snake/body_vertical.png'
).convert_alpha()
```

```
            self.body_horizontal =
pygame.image.load(f'{directory}/media/snake/body_horizontal.pn
g').convert_alpha()

            self.body_tr =
pygame.image.load(f'{directory}/media/snake/body_tr.png').conv
ert_alpha()
            self.body_tl =
pygame.image.load(f'{directory}/media/snake/body_tl.png').conv
ert_alpha()
            self.body_br =
pygame.image.load(f'{directory}/media/snake/body_br.png').conv
ert_alpha()
            self.body_bl =
pygame.image.load(f'{directory}/media/snake/body_bl.png').conv
ert_alpha()

        def draw_snake(self):
            self.update_head_graphics()
            self.update_tail_graphics()

            for index,block in enumerate(self.body):
                x_pos = int(block.x * cell_size)
                y_pos = int(block.y * cell_size)
                block_rect =
pygame.Rect(x_pos,y_pos,cell_size,cell_size)

                if index == 0:
                    screen.blit(self.head,block_rect)
                elif index == len(self.body) - 1:
                    screen.blit(self.tail,block_rect)
                else:
                    previous_block = self.body[index + 1] -
block
                    next_block = self.body[index - 1] - block
                    if previous_block.x == next_block.x:

screen.blit(self.body_vertical,block_rect)
                    elif previous_block.y == next_block.y:

screen.blit(self.body_horizontal,block_rect)
                    else:
                        if previous_block.x == -1 and
next_block.y == -1 or previous_block.y == -1 and next_block.x
== -1:
```

```
screen.blit(self.body_tl,block_rect)
                        elif previous_block.x == -1 and
next_block.y == 1 or previous_block.y == 1 and next_block.x ==
-1:

screen.blit(self.body_bl,block_rect)
                        elif previous_block.x == 1 and
next_block.y == -1 or previous_block.y == -1 and next_block.x
== 1:

screen.blit(self.body_tr,block_rect)
                        elif previous_block.x == 1 and
next_block.y == 1 or previous_block.y == 1 and next_block.x ==
1:

screen.blit(self.body_br,block_rect)

        def update_head_graphics(self):
            head_relation = self.body[1] - self.body[0]
            if head_relation == Vector2(1,0): self.head =
self.head_left
            elif head_relation == Vector2(-1,0): self.head =
self.head_right
            elif head_relation == Vector2(0,1): self.head =
self.head_up
            elif head_relation == Vector2(0,-1): self.head =
self.head_down

        def update_tail_graphics(self):
            tail_relation = self.body[-2] - self.body[-1]
            if tail_relation == Vector2(1,0): self.tail =
self.tail_left
            elif tail_relation == Vector2(-1,0): self.tail =
self.tail_right
            elif tail_relation == Vector2(0,1): self.tail =
self.tail_up
            elif tail_relation == Vector2(0,-1): self.tail =
self.tail_down

        def move_snake(self):
            nonlocal moveConstant
            if moveConstant == 1:
                if self.new_block == True:
                    body_copy = self.body[:]
```

```
                    body_copy.insert(0,body_copy[0] +
self.direction)
                    self.body = body_copy[:]
                    self.new_block = False
                else:
                    body_copy = self.body[:-1]
                    body_copy.insert(0,body_copy[0] +
self.direction)
                    self.body = body_copy[:]

        def add_block(self):
            self.new_block = True

        def reset(self):
            nonlocal isGameOver, dsConstant, moveConstant
            isGameOver = False
            dsConstant = 0
            moveConstant = 0
            self.body =
[Vector2(5,10),Vector2(4,10),Vector2(3,10)]
            self.direction = Vector2(0,0)


    class FRUIT:
        def __init__(self):
            self.randomize()

        def draw_fruit(self):
            fruit_rect = pygame.Rect(int(self.pos.x *
cell_size),int(self.pos.y * cell_size),cell_size,cell_size)
            screen.blit(apple,fruit_rect)
            #pygame.draw.rect(screen,(126,166,114),fruit_rect)

        def randomize(self):
            self.x = random.randint(0,32 - 1)
            self.y = random.randint(0,18 - 1)
            self.pos = Vector2(self.x,self.y)

    class MAIN:
        def __init__(self):
            self.snake = SNAKE()
            self.fruit = FRUIT()

        def update(self):
            self.snake.move_snake()
```

```
            self.check_collision()
            self.check_fail()

        def draw_elements(self):
            self.draw_grass()
            self.fruit.draw_fruit()
            self.snake.draw_snake()
            self.draw_score()

        def check_collision(self):
            if self.fruit.pos == self.snake.body[0]:
                self.fruit.randomize()
                self.snake.add_block()
                # self.snake.play_crunch_sound()

            for block in self.snake.body[1:]:
                if block == self.fruit.pos:
                    self.fruit.randomize()

        def check_fail(self):
            nonlocal restartPressed

            if not 0 <= self.snake.body[0].x < 32 or not 0 <=
self.snake.body[0].y < 18:
                if restartPressed:
                    restartPressed = False
                    self.game_over()

            for block in self.snake.body[1:]:
                if block == self.snake.body[0]:
                    if int(len(self.snake.body)-3) > 0:
                        restartPressed = False
                    else:
                        restartPressed = True
                    if restartPressed == False:
                        self.game_over()

        def game_over(self):
            nonlocal isGameOver, restartPressed, dsConstant
            isGameOver = True
            if dsConstant == 0:
                self.highScoreUpdate(int(len(self.snake.body)
- 3))
```

```
                s = pygame.Surface((1280,720))  # the size of
your rect
                s.set_alpha(128)                # alpha level
                s.fill((0,0,0))           # this fills the
entire surface
                screen.blit(s, (0,0))

                font =
pygame.font.Font(f'{directory}/font/dogicapixel.ttf', 35)
                text = font.render('Click to try again', True,
(0,0,0))
                textRect = text.get_rect()
                textRect.center = (640, 360)
                screen.blit(text, textRect)

                dsConstant = 1

            if restartPressed: self.restart()

        def restart(self):
            self.snake.reset()

        def draw_grass(self):
            grass_color = (167,209,61)
            for row in range(18):
                if row % 2 == 0:
                    for col in range(32):
                        if col % 2 == 0:
                            grass_rect = pygame.Rect(col *
cell_size,row * cell_size,cell_size,cell_size)

pygame.draw.rect(screen,grass_color,grass_rect)
                else:
                    for col in range(32):
                        if col % 2 != 0:
                            grass_rect = pygame.Rect(col *
cell_size,row * cell_size,cell_size,cell_size)

pygame.draw.rect(screen,grass_color,grass_rect)

        def draw_score(self):
            score_text = str(len(self.snake.body) - 3)
            score_surface =
game_font.render(score_text,True,(56,74,12))
            score_x = int(cell_size * 32 - 60)
```

```
            score_y = int(cell_size * 18 - 40)
            score_rect = score_surface.get_rect(center =
(score_x,score_y))
            apple_rect = apple.get_rect(midright =
(score_rect.left,score_rect.centery))
            bg_rect =
pygame.Rect(apple_rect.left,apple_rect.top,apple_rect.width +
score_rect.width + 6,apple_rect.height)

            pygame.draw.rect(screen,(167,209,61),bg_rect)
            screen.blit(score_surface,score_rect)
            screen.blit(apple,apple_rect)
            pygame.draw.rect(screen,(56,74,12),bg_rect,2)

        #checks to update highscore
        def highScoreUpdate(self, score):
            with open(f'{directory}/scores.json', 'r') as
scoreFileRead:
                scoreData = json.load(scoreFileRead)
            with open(f'{directory}/scores.json', 'w') as
scoreFileWrite:
                if scoreData['Snake']['score'] < int(score):
                    updateFirstDatabase(3, 'Snake',
int(score), username, scoreData)
                    scoreData['Snake']['score'] = score
                    scoreData['Snake']['user'] = username
                json.dump(scoreData, scoreFileWrite)
            return score

    pygame.mixer.pre_init(44100,-16,2,512)
    cell_size = 40
    screen = pygame.display.set_mode((32 * cell_size, 18 *
cell_size))
    apple =
pygame.image.load(f'{directory}/media/snake/apple.png').conver
t_alpha()
    game_font =
pygame.font.Font(f'{directory}/font/PoetsenOne-Regular.ttf',
25)

    SCREEN_UPDATE = pygame.USEREVENT
    pygame.time.set_timer(SCREEN_UPDATE,150)

    main_game = MAIN()
```

```
    while True:
        print(isGameOver, restartPressed)
        for event in pygame.event.get():
            # print(isGameOver)
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if isGameOver == True:
                if event.type == pygame.KEYDOWN and event.key
== pygame.K_ESCAPE:
                    gamesScreen()
            if event.type == pygame.MOUSEBUTTONDOWN and
isGameOver:
                restartPressed = True
            if event.type == SCREEN_UPDATE:
                if isGameOver:
                    main_game.game_over()
                else:
                    main_game.update()
            if event.type == pygame.KEYDOWN and restartPressed
== True:
                moveConstant = 1
                if event.key == pygame.K_UP:
                    if main_game.snake.direction.y != 1:
                        main_game.snake.direction =
Vector2(0,-1)
                if event.key == pygame.K_RIGHT:
                    if main_game.snake.direction.x != -1:
                        main_game.snake.direction =
Vector2(1,0)
                if event.key == pygame.K_DOWN:
                    if main_game.snake.direction.y != -1:
                        main_game.snake.direction =
Vector2(0,1)
                if event.key == pygame.K_LEFT:
                    if main_game.snake.direction.x != 1:
                        main_game.snake.direction = Vector2(-
1,0)

        if not isGameOver:
            screen.fill((175,215,70))
            main_game.draw_elements()
        pygame.display.update()
        clock.tick(60)
```

```
#starts endless runner
def endlessRunner():
    #update score in game
    def score_update():
        current_score = int((pygame.time.get_ticks() -
start_time)/40)
        score_surface = test_font.render(f'Score:
{current_score}', False, (64, 64, 64))
        score_rect = score_surface.get_rect(center = (640,
100))
        screen.blit(score_surface, score_rect)
        return current_score

    #checks to update highscore
    def highScoreUpdate(score):
        with open(f'{directory}/scores.json', 'r') as
scoreFileRead:
            scoreData = json.load(scoreFileRead)
        with open(f'{directory}/scores.json', 'w') as
scoreFileWrite:
            if scoreData['EndlessRunner']['score'] <
int(score):
                updateFirstDatabase(2, 'EndlessRunner',
int(score), username, scoreData)
                scoreData['EndlessRunner']['score'] = score
                scoreData['EndlessRunner']['user'] = username
            json.dump(scoreData, scoreFileWrite)
        return score

    #moves obstacles
    def obstacle_movement(obst_rect_list):
        if obst_rect_list:
            for obst_rect in obst_rect_list:
                obst_rect.x -= 14
                if obst_rect.bottom == 600:
screen.blit(snail_surface, obst_rect)
                else: screen.blit(fly_surface, obst_rect)
            obst_rect_list = [obst for obst in obst_rect_list
if obst.right > 0]
            return obst_rect_list
        else: return []

    #detects collisions
    def collisions(player, obstacles):
        if obstacles:
```

```
            for obstacle_rect in obstacles:
                if player.colliderect(obstacle_rect): return
False
        return True

    #animates player
    def player_animation():
        nonlocal player_surface, player_index

        if player_rect.bottom < 600:
            player_surface = player_jump
        else:
            player_index += 0.1
            if player_index >= len(player_walk): player_index
= 0
            player_surface = player_walk[int(player_index)]

    #helper variables
    game_active = False
    start_time = 0
    screen = pygame.display.set_mode((1280, 720))
    score = 0
    player_index = 0

    #font
    test_font =
pygame.font.Font(f'{directory}/font/Pixeltype.ttf', 75)

    #import images
    sky_surface =
pygame.image.load(f'{directory}/media/endlessRunner/Sky.png').
convert()
    ground_surface =
pygame.image.load(f'{directory}/media/endlessRunner/ground.png
').convert()
    snail_surface =
pygame.image.load(f'{directory}/media/endlessRunner/snail/snai
l1.png').convert_alpha()
    fly_surface =
pygame.image.load(f'{directory}/media/endlessRunner/Fly/Fly1.p
ng').convert_alpha()
    player_walk_1 =
pygame.image.load(f'{directory}/media/endlessRunner/player/pla
yer_walk_1.png').convert_alpha()
```

```
    player_walk_2 =
pygame.image.load(f'{directory}/media/endlessRunner/player/pla
yer_walk_2.png').convert_alpha()
    player_jump =
pygame.image.load(f'{directory}/media/endlessRunner/player/jum
p.png').convert_alpha()
    player_stand =
pygame.image.load(f'{directory}/media/endlessRunner/player/pla
yer_stand.png').convert_alpha()

    #scale images
    sky_surface = pygame.transform.scale2x(sky_surface)
    ground_surface = pygame.transform.scale2x(ground_surface)
    snail_surface = pygame.transform.scale2x(snail_surface)
    fly_surface = pygame.transform.scale2x(fly_surface)
    player_walk_1 = pygame.transform.scale2x(player_walk_1)
    player_walk_2 = pygame.transform.scale2x(player_walk_2)
    player_walk = [player_walk_1, player_walk_2]
    player_jump = pygame.transform.scale2x(player_jump)
    player_surface = player_walk[player_index]
    player_stand = pygame.transform.scale(player_stand,
(player_stand.get_width() * 4, player_stand.get_height() * 4))

    #store obstacles in a list
    obst_list = []

    #player rect
    player_rect = player_surface.get_rect(midbottom = (160,
600))
    player_stand_rect = player_stand.get_rect(center = (640,
360))

    #text
    game_name = test_font.render('Endless Runner', False,
(111, 196, 169))
    game_name_rect = game_name.get_rect(center = (640, 144))
    game_message = test_font.render('Press space to run',
False, (111, 196, 169))
    game_message_rect = game_message.get_rect(center = (640,
594))

    #timer
    obst_timer = pygame.USEREVENT+1
    pygame.time.set_timer(obst_timer, 1600)
```

```
    #gravity variable
    player_gravity = 0

    #game loop
    while True:
        for event in pygame.event.get():
            #exit on close button
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()

            if game_active:
                #jump
                if event.type == pygame.MOUSEBUTTONDOWN:
                    if player_rect.collidepoint(event.pos) and
player_rect.bottom >= 600:
                        player_gravity = -33
                #jump
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_SPACE and
player_rect.bottom >= 600:
                        player_gravity = -33

                #what happens when obstacle timer  triggered
                if event.type == obst_timer:
                    if random.randint(0,2):
obst_list.append(snail_surface.get_rect(bottomright =
(random.randint(1800, 2200), 600)))
                    else:
obst_list.append(fly_surface.get_rect(bottomright =
(random.randint(1800, 2200), 420)))
            else:
                #restart game
                if event.type == pygame.KEYDOWN and event.key
== pygame.K_SPACE:
                    game_active = True
                    start_time = pygame.time.get_ticks()
                #leave game
                if event.type == pygame.KEYDOWN and event.key
== pygame.K_ESCAPE:
                    gamesScreen()

        if game_active:
            #sky and ground
            screen.blit(sky_surface, (0, 0))
```

```
            screen.blit(ground_surface, (0, 600))
            score = score_update()

            #obstacle movement
            obst_list = obstacle_movement(obst_list)

            #collisions
            game_active = collisions(player_rect, obst_list)

            #accelaration due to gravity and player animation
            player_gravity += 2
            player_rect.y += player_gravity
            if player_rect.bottom >= 600: player_rect.bottom =
600
            player_animation()
            screen.blit(player_surface, player_rect)

        else:
            #game over bg
            screen.fill((94, 129, 162))

            #update highscore?
            highScoreUpdate(score)

            #display finalscore
            score_message = test_font.render(f'Your score:
{score}', False, (111, 196, 169))
            score_message_rect = score_message.get_rect(center
= (640, 594))

            #clear obstacle list
            obst_list.clear()

            player_rect.midbottom = (160, 600)
            player_gravity = 0

            screen.blit(game_name, game_name_rect)
            if score == 0:
                screen.blit(game_message, game_message_rect)
            else:
                screen.blit(score_message, score_message_rect)

            screen.blit(player_stand, player_stand_rect)

        #update frames
```

```
        pygame.display.update()
        clock.tick(60)

#starts rps
def rockPaperScissors():
    pygame.display.quit()

    plays = ['rock', 'paper', 'scissors']
    print('---------------------\n|ROCK-PAPER-SCISSORS|\n-----
----------------')
    while True:
        userPlay = input('Enter your play: ')
        if userPlay.lower() not in plays:
            print('not a valid option')
            continue
        else:
            aiPlay = random.choice(plays)
            print(f'AI played: {aiPlay}')
            aiPlay = plays.index(aiPlay)
            userPlay = plays.index(userPlay.lower())
            if userPlay == aiPlay:
                print('Draw.')
            elif [userPlay, aiPlay] in [[0, 2], [1, 0], [2,
1]]:
                print('You Won!')
            elif [userPlay, aiPlay] in [[0, 1], [1, 2], [2,
0]]:
                print('You Lost!')
        isRestart = input('Play again?(y/n): ')
        if isRestart.lower() in 'yY':
            continue
        else:
            break

    pygame.display.set_mode((1280, 720))
    gamesScreen()

#starts tictactoe
def tictactoe():
    #redefining display dimensions
    pygame.display.quit()
    screen = pygame.display.set_mode((400, 500), 0, 32)

    #variables
    XO = 'x'
```

```
    winner = None
    draw = None
    width = 400
    height = 400
    white = (255, 255, 255)
    line_color = (0, 0, 0)
    playing = True

    # setting up a 3X3 board
    board = [[None]*3, [None]*3, [None]*3]

    #images
    initiating_window =
pygame.image.load(f'{directory}/media/tictactoe/modified_cover
.png')
    x_img =
pygame.image.load(f'{directory}/media/tictactoe/X_modified.png
')
    y_img =
pygame.image.load(f'{directory}/media/tictactoe/o_modified.png
')

    #resizing
    initiating_window =
pygame.transform.scale(initiating_window, (width, height +
100))
    x_img = pygame.transform.scale(x_img, (80, 80))
    o_img = pygame.transform.scale(y_img, (80, 80))

    #function for loading screen
    def game_initiating_window():
        #displaying over the screen
        screen.blit(initiating_window, (0, 0))

        #updating the display
        pygame.display.update()
        time.sleep(0.5)
        screen.fill(white)

        #drawing vertical lines
        pygame.draw.line(screen, line_color, (width / 3, 0),
(width / 3, height), 7)
        pygame.draw.line(screen, line_color, (width / 3 * 2,
0), (width / 3 * 2, height), 7)
```

```
        #drawing horizontal lines
        pygame.draw.line(screen, line_color, (0, height / 3),
(width, height / 3), 7)
        pygame.draw.line(screen, line_color, (0, height / 3 *
2), (width, height / 3 * 2), 7)
        draw_status()

    #checks winner status
    def draw_status():
        nonlocal draw

        if winner is None: message = XO.upper() + "'s Turn"
        else: message = winner.upper() + " won!"
        if draw: message = "Game Draw!"

        #font object
        font = pygame.font.Font(None, 30)

        #message text
        text = font.render(message, 1, (255, 255, 255))

        #display message
        screen.fill ((0, 0, 0), (0, 400, 500, 100))
        text_rect = text.get_rect(center =(width / 2, 500-50))
        screen.blit(text, text_rect)
        pygame.display.update()

    def check_win():
        nonlocal board, winner, draw

        # checking for winning rows
        for row in range(0, 3):
            if((board[row][0] == board[row][1] ==
board[row][2]) and (board [row][0] is not None)):
                winner = board[row][0]
                pygame.draw.line(screen, (250, 0, 0), (0, (row
+ 1)*height / 3 -height / 6), (width, (row + 1)*height / 3 -
height / 6 ), 10)
                break

        # checking for winning columns
        for col in range(0, 3):
            if((board[0][col] == board[1][col] ==
board[2][col]) and (board[0][col] is not None)):
                winner = board[0][col]
```

```
                pygame.draw.line(screen, (250, 0, 0), ((col +
1)* width / 3 - width / 6, 0), ((col + 1)* width / 3 - width /
6, height), 10)
                break

        # check for diagonal winners
        if (board[0][0] == board[1][1] == board[2][2]) and
(board[0][0] is not None):
            #game won diagonally left to right
            winner = board[0][0]
            pygame.draw.line(screen, (250, 70, 70), (0, 0),
(400, 400), 10)

        if (board[0][2] == board[1][1] == board[2][0]) and
(board[0][2] is not None):
            #game won diagonally right to left
            winner = board[0][2]
            pygame.draw.line(screen, (250, 70, 70), (400, 0),
(0, 400), 10)

        if(all([all(row) for row in board]) and winner is None
): draw = True
        draw_status()

    def drawXO(row, col):
        nonlocal board, XO

        #row pos
        if row == 1: posx = 30
        if row == 2: posx = width / 3 + 30
        if row == 3: posx = width / 3 * 2 + 30

        #column pos
        if col == 1: posy = 30
        if col == 2: posy = height / 3 + 30
        if col == 3: posy = height / 3 * 2 + 30

        #board value to display
        board[row-1][col-1] = XO

        if(XO == 'x'):
            screen.blit(x_img, (posy, posx))
            XO = 'o'
        else:
            screen.blit(o_img, (posy, posx))
```

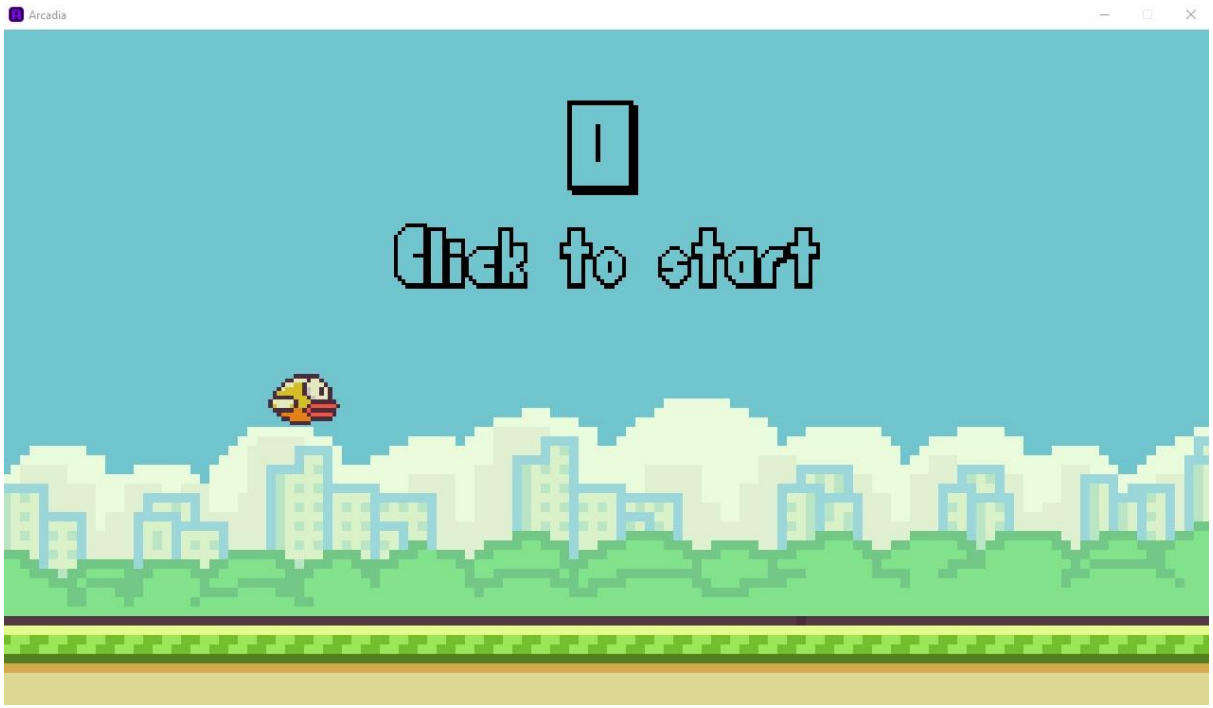
```
            XO = 'x'
        pygame.display.update()

    def user_click():
        nonlocal playing
        #get coordinates of mouse click
        x, y = pygame.mouse.get_pos()

        #column of mouse click
        if(x<width / 3): col = 1
        elif (x<width / 3 * 2): col = 2
        elif(x<width): col = 3
        else: col = None

        #row of mouse click
        if(y<height / 3): row = 1
        elif (y<height / 3 * 2): row = 2
        elif(y<height): row = 3
        else: row = None

        #draw the images at desired positions
        if(row and col and board[row-1][col-1] is None and
playing == True):
            nonlocal XO
            drawXO(row, col)
            check_win()

    #reset game
    def reset_game():
        nonlocal board, winner, XO, draw, playing
        time.sleep(0.5)
        XO = 'x'
        draw = False
        playing = True
        game_initiating_window()
        winner = None
        board = [[None]*3, [None]*3, [None]*3]

    game_initiating_window()

    #game loop
    while True:
        for event in pygame.event.get():
            if event.type == QUIT:
```

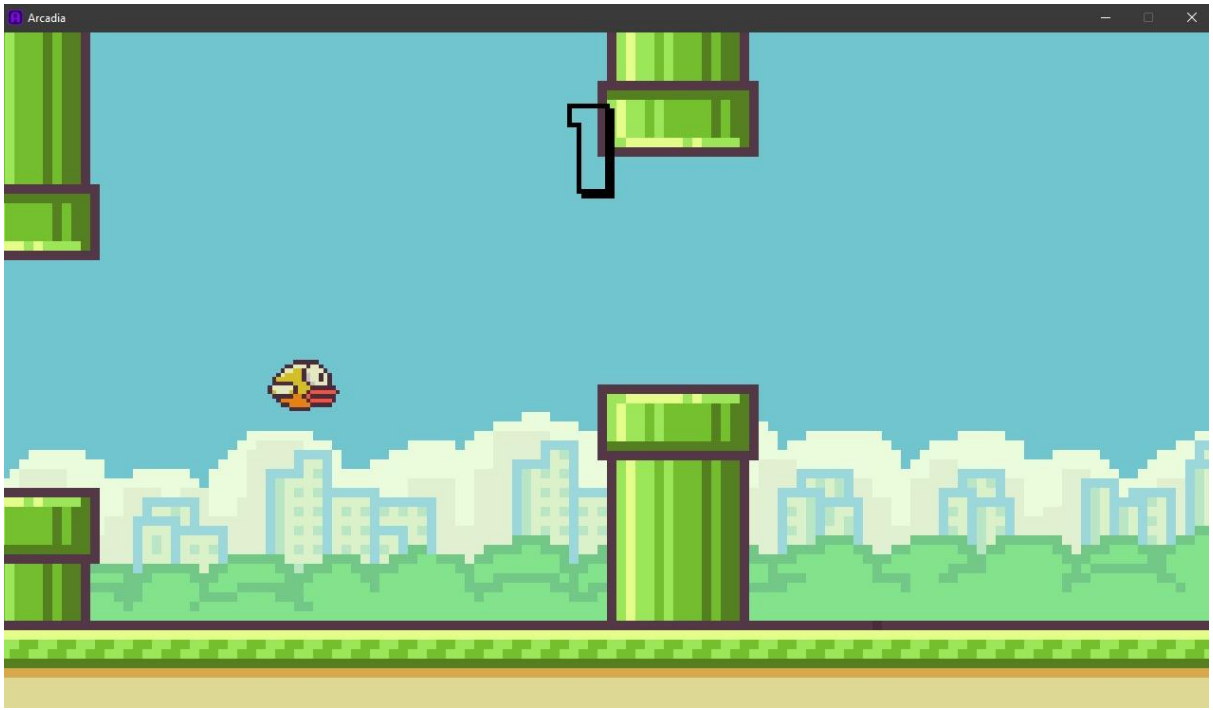
```
                pygame.quit()
                exit()
            elif event.type == pygame.KEYDOWN and event.key ==
pygame.K_ESCAPE:
                pygame.display.quit()
                pygame.display.set_mode((1280, 720))
                gamesScreen()
            elif event.type == pygame.MOUSEBUTTONDOWN:
                user_click()
                playing = True
                if(winner or draw):
                    playing = True
                    reset_game()

        #update frames
        pygame.display.update()
        clock.tick(60)


#initiate username and screen variables, then start game
username, screen = login()
mainMenu()
```

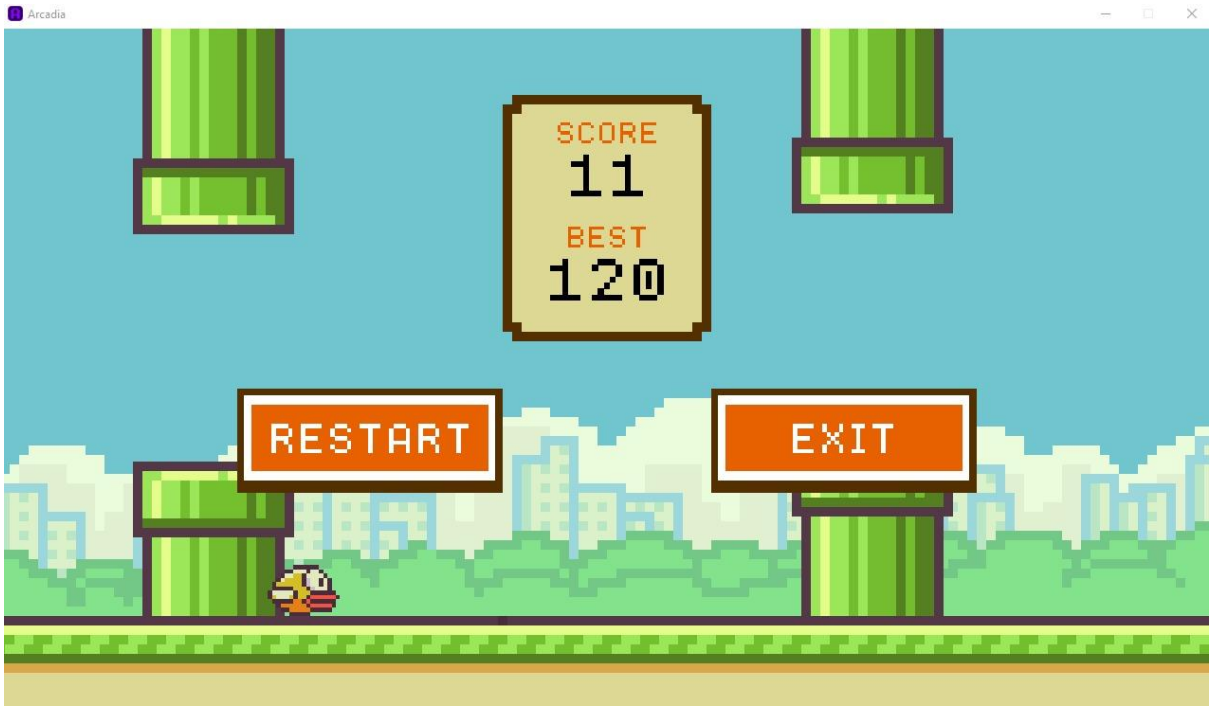
Output

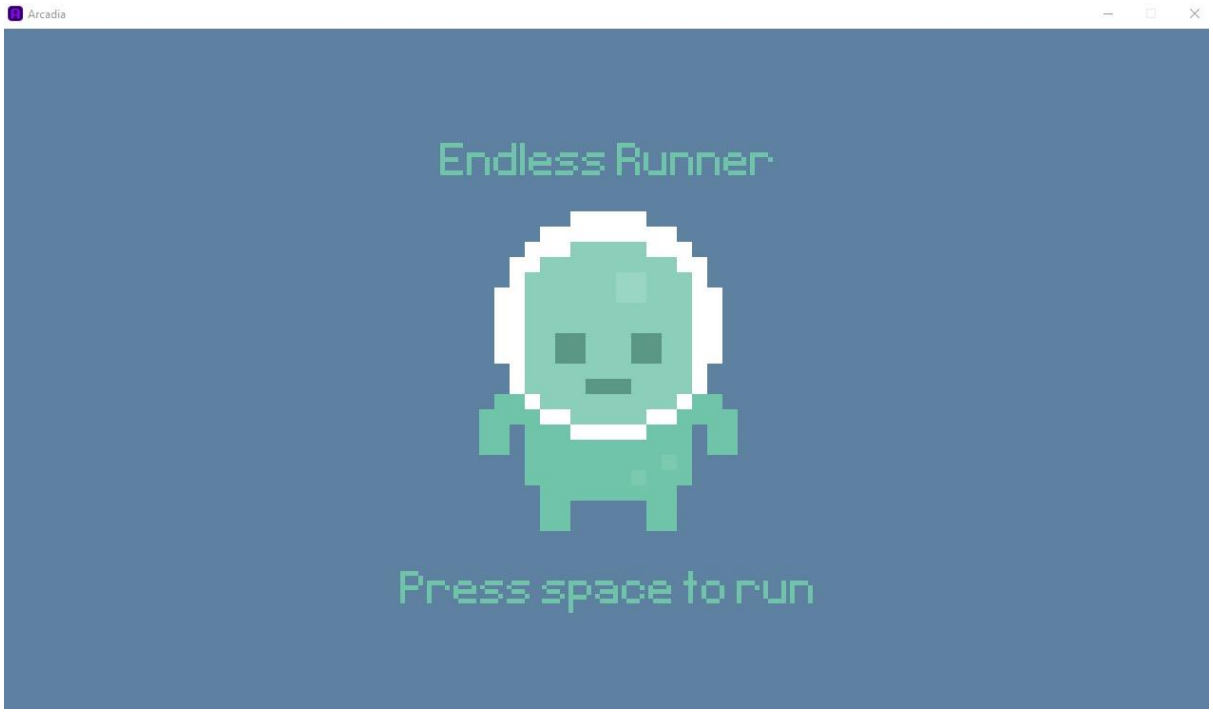

Arcadia
Arcadia
Play
Exit
A game by Dhanvin Sajith and Aldrin Dsouza

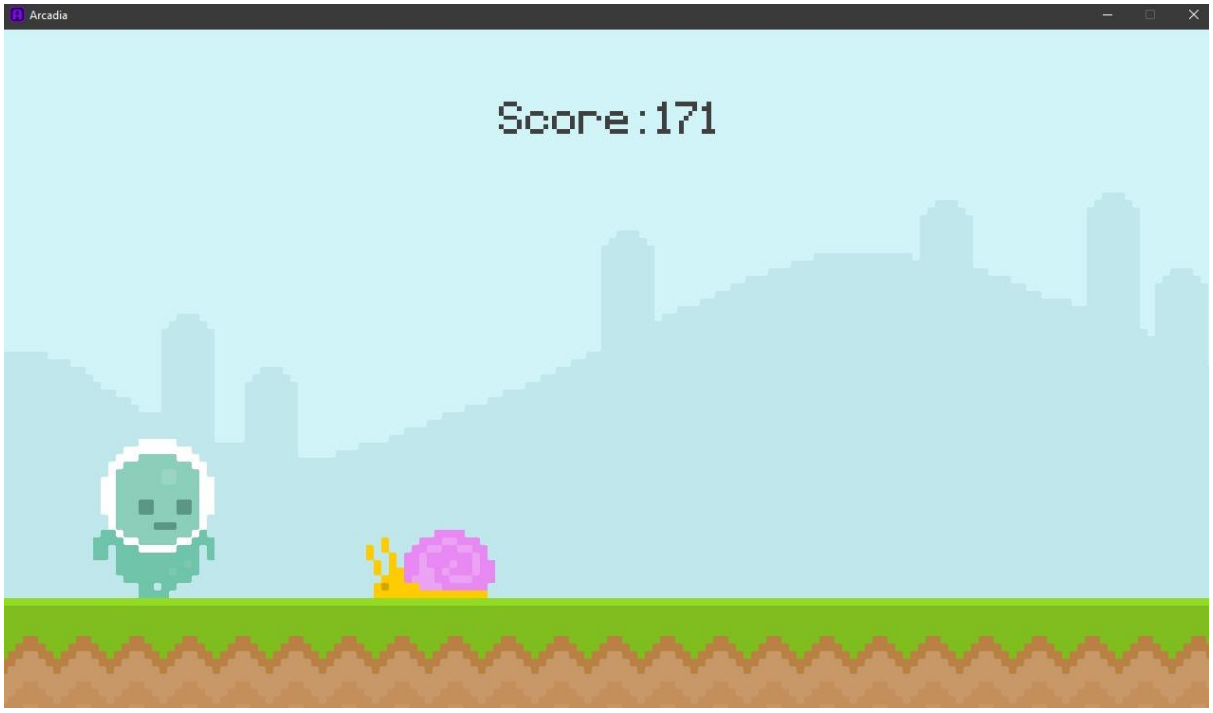

Arcadia
The Arcade
Flappy Bird
Endless Runner
Endless Runner
Press space to run
PONG
Pong
Snake

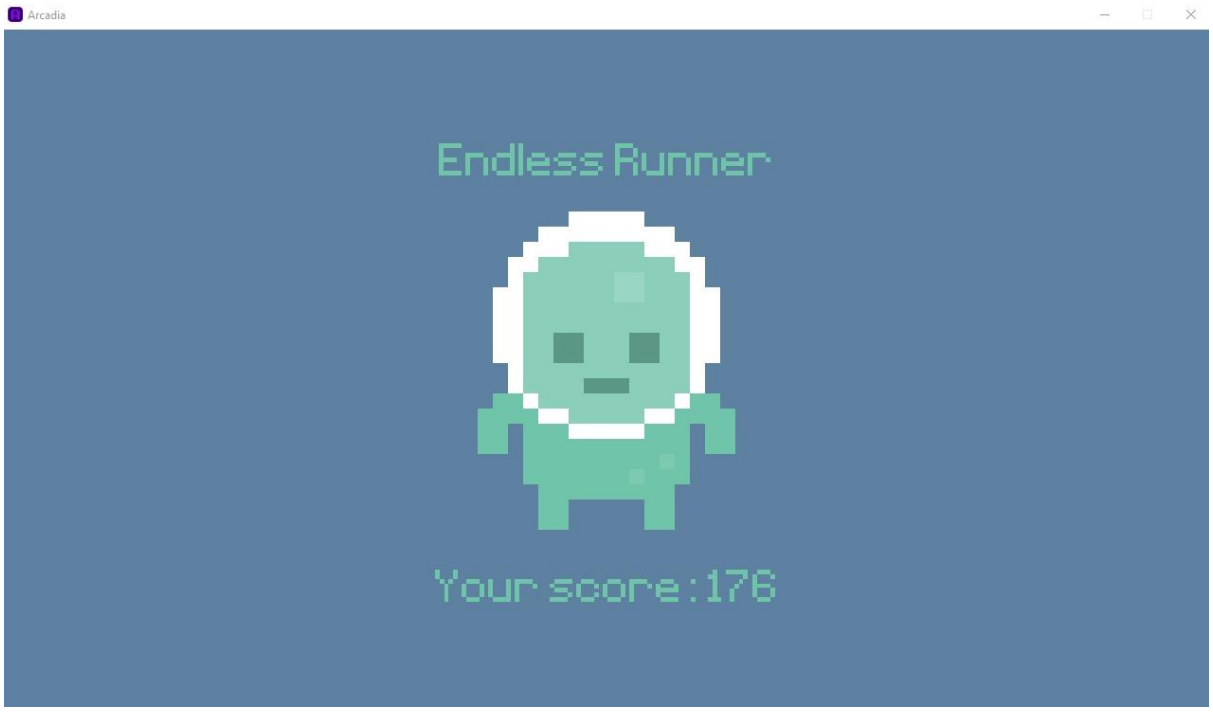

Arcadia
0
Click to start


Arcadia
1

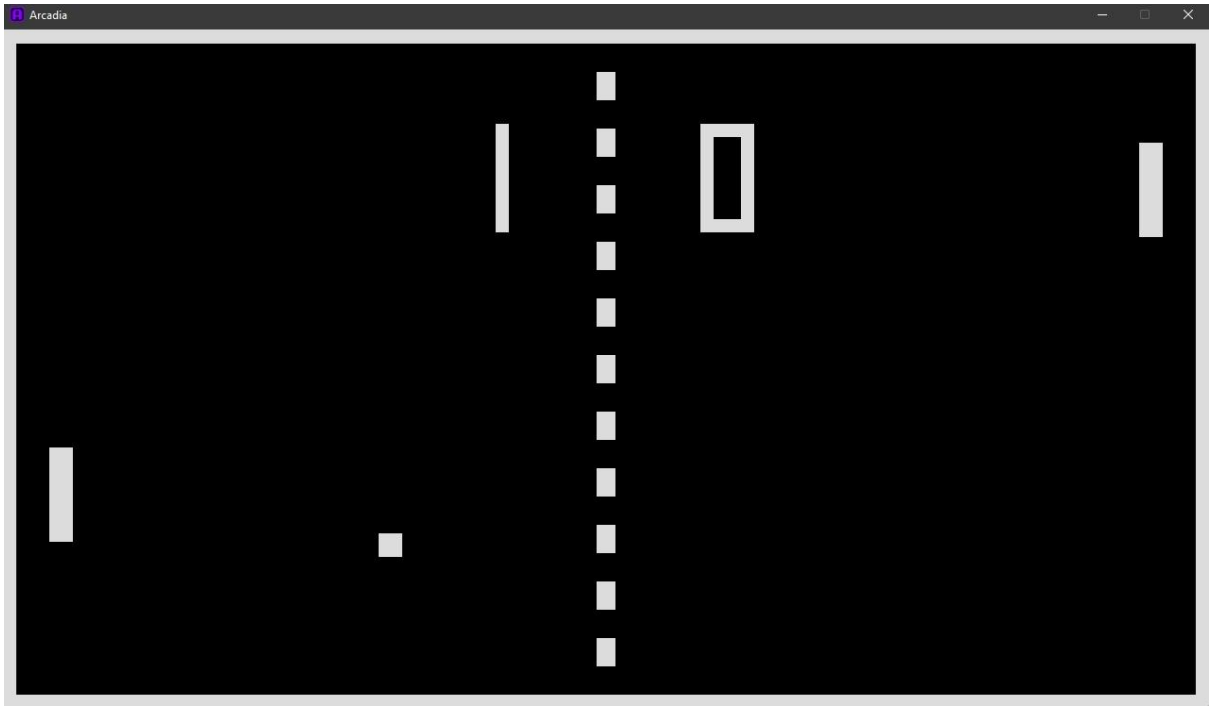

Arcadia
SCORE
11
BEST
120
RESTART
EXIT

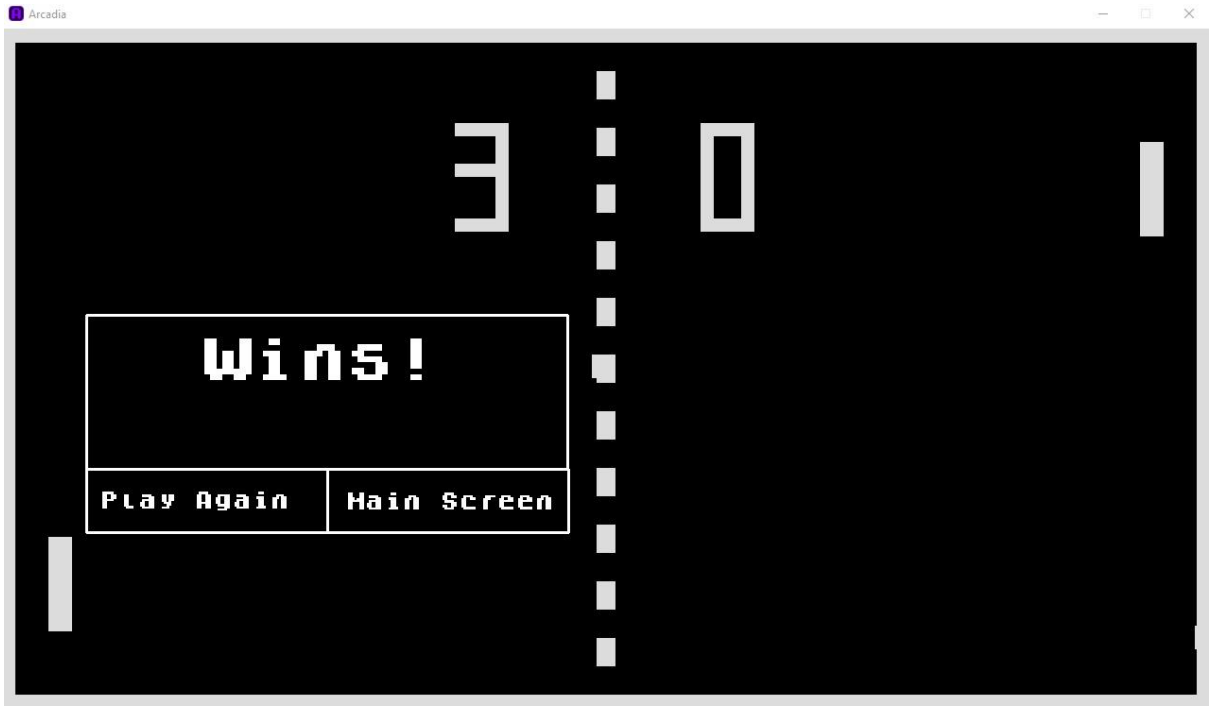

Arcadia
Endless Runner
Press space to run

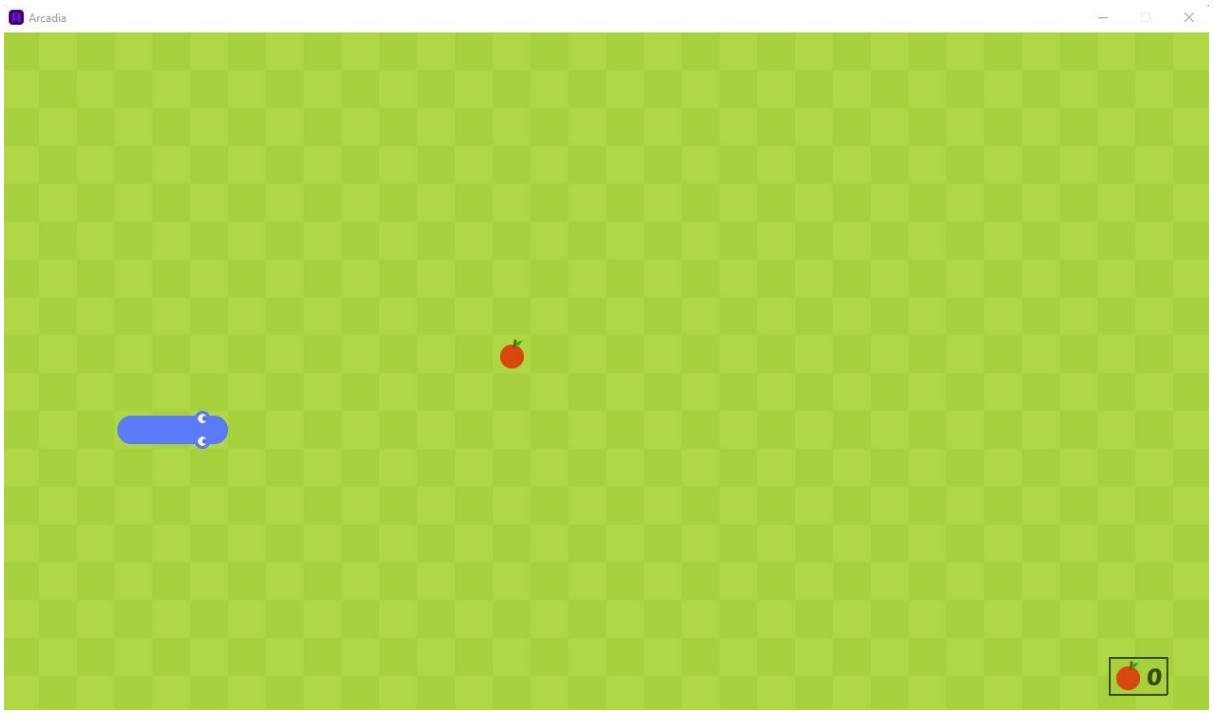

Arcadia
Score:171

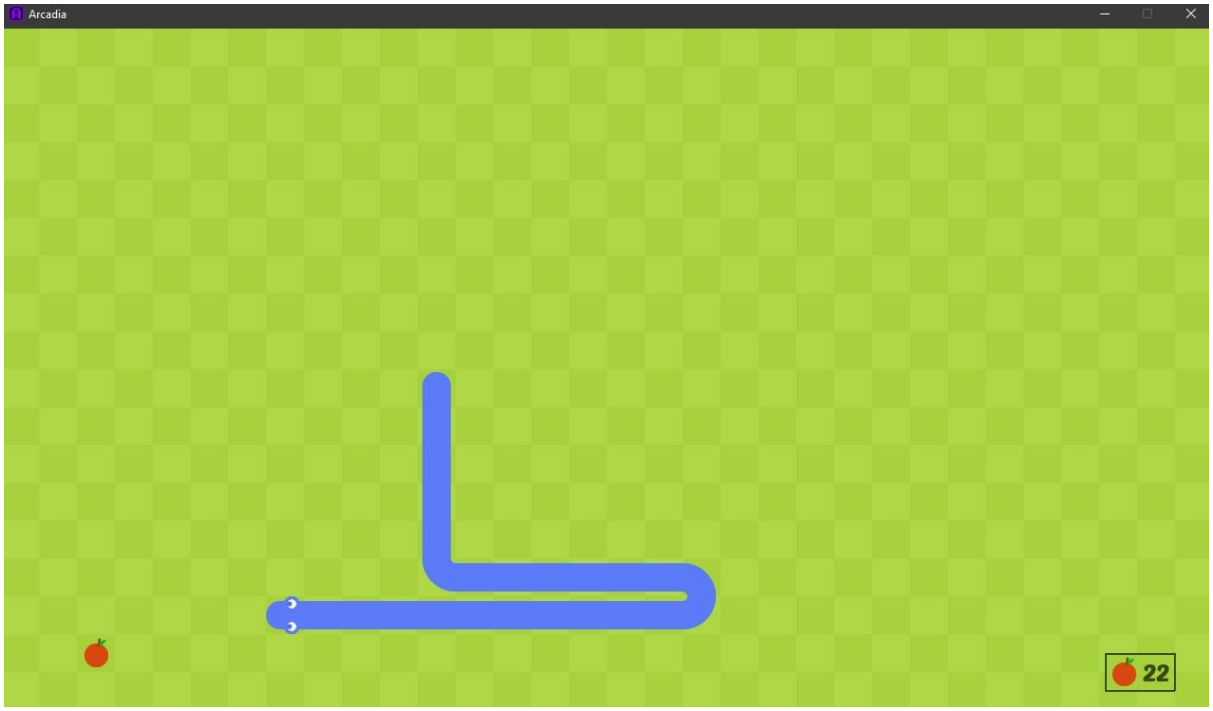

Arcadia
Endless Runner
Your score:176

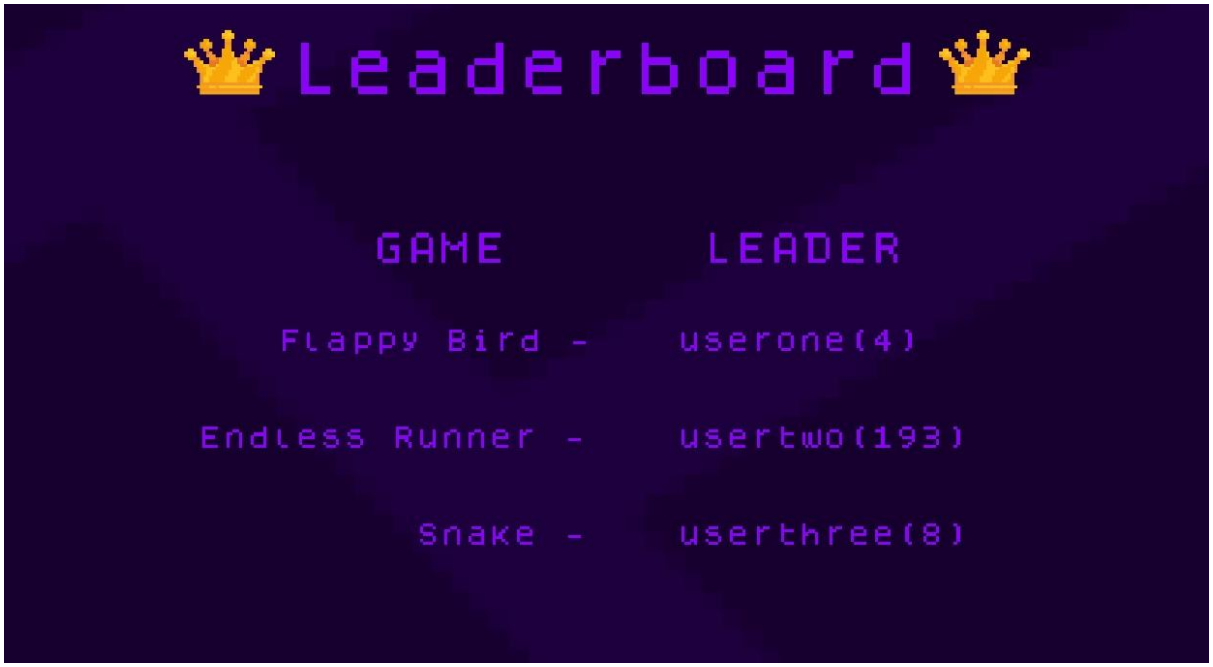

Arcadia
PONG
SINGLE PLAYER
MULTI PLAYER


Arcadia
0


Arcadia
3
0
Wins!
Play Again
Main Screen


Arcadia
0


Arcadia
22


Leaderboard
GAME
LEADER
Flappy Bird -
userone(4)
Endless Runner -
usertwo(193)
Snake -
userthree(8)

Bibliography

- Stack overflow
- Python documentation
- Pygame documentation
- Youtube