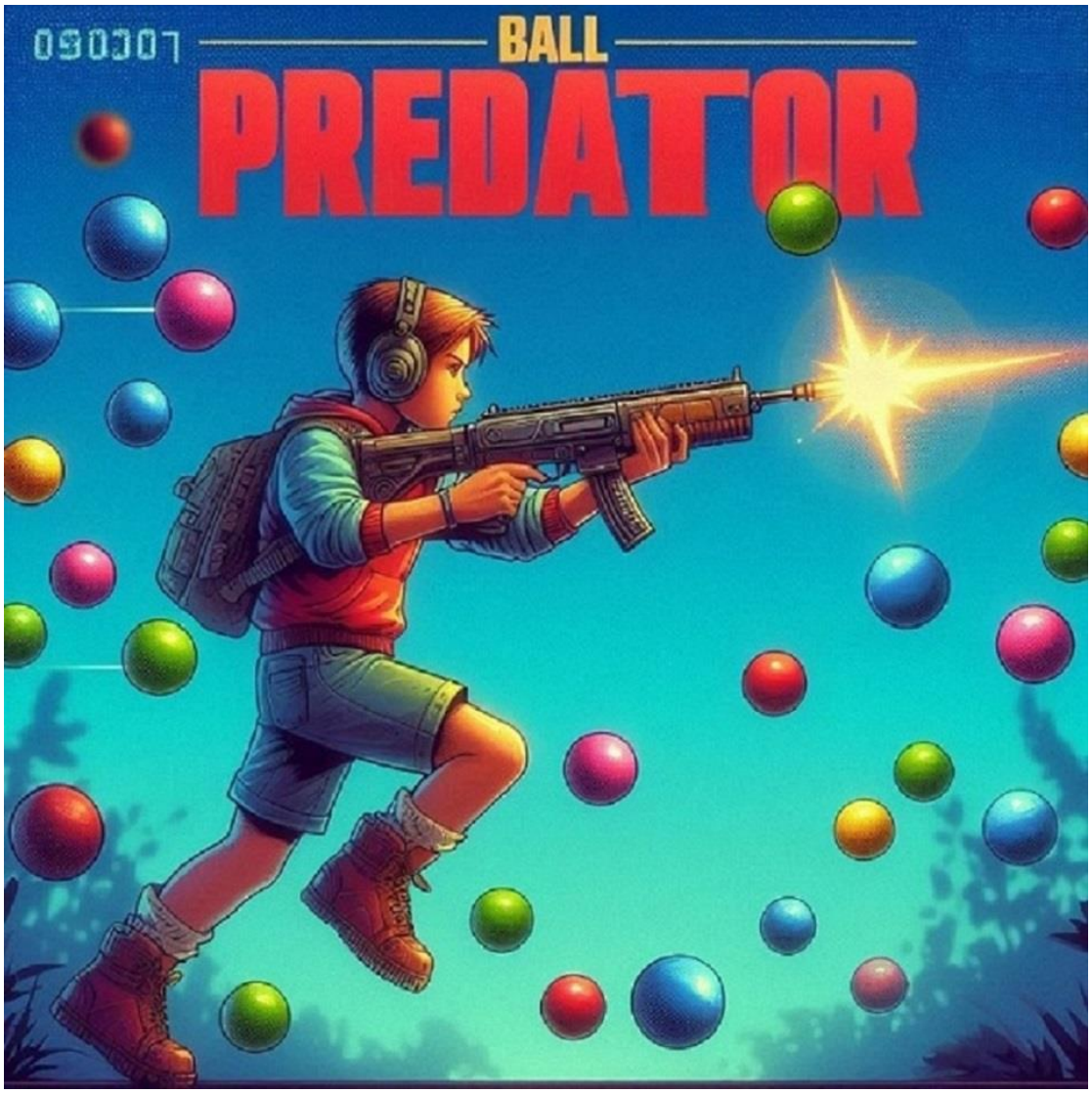
Ball Predator



Περιγραφή της λειτουργίας του παιχνιδιού Ball Predator

Το Ball Predator είναι ένα κλασικό Shooter Game. Είναι ένα απλό παιχνίδι όσον αφορά το game play, με απλά γραφικά, είναι για έναν παίκτη και είναι μικρής διάρκειας. Σκοπός του παιχνιδιού είναι ο παίκτης να μαζέψει όσο περισσότερους πόντους μπορεί.

Πιο συγκεκριμένα, όταν ανοίγει το παράθυρο του παιχνιδιού, το πρώτο πράγμα που φαίνεται είναι το αρχικό μενού με το εξώφυλλο του τίτλου του παιχνιδιού. Κάτω από το εξώφυλλο φαίνεται το μεγαλύτερο σκορ που έχει επιτευχθεί και ακριβώς κάτω από το σκορ φαίνεται η οδηγία που προτρέπει τον παίκτη να πατήσει το πλήκτρο space ώστε να παίξει το παιχνίδι. Εάν ο χρήστης πατήσει το πλήκτρο space, τότε το παιχνίδι ξεκινάει.

Όταν το παιχνίδι ξεκινήσει, η αρχική σκηνή αποτελείται από μία εικόνα κτιρίων σε μία πόλη, κατά το ξημέρωμα. Κάτω από τη σκηνή φαίνεται η λεζάντα, πάνω στην οποία αναγράφεται το σκορ, ο αριθμός των ζωών που απομένουν, καθώς και ο αριθμός του επιπέδου στο οποίο βρίσκεται ο παίκτης. Στο πάνω και δεξιά μέρος της σκηνής αναγράφεται ο χρόνος που απομένει για να ολοκληρωθεί το παιχνίδι. Το παιχνίδι είναι διάρκειας 120 δευτερολέπτων και περιλαμβάνει 4 διαφορετικά επίπεδα-σκηνές. Κάθε επίπεδο δείχνει το πέρασμα του χρόνου κι έτσι ξεκινώντας από το ξημέρωμα στο πρώτο επίπεδο, συνεχίζει με την ανατολή του ηλίου στο δεύτερο, στη συνέχεια γίνεται απόγευμα και στο τελευταίο επίπεδο γίνεται νύχτα.

Μέσα στη σκηνή, αιωρούνται ιπτάμενες μπάλες διαφορετικών χρωμάτων. Σκοπός του παίκτη, είναι να στοχεύει τις μπάλες και να τις πυροβολεί, κερδίζοντας με αυτό τον τρόπο πόντους. Η στόχευση γίνεται μέσω ενός γραφικού στοχάστρου το οποίο έχει τη θέση του κέρσορα του ποντικιού. Ο παίκτης τον κινεί χρησιμοποιώντας το ποντίκι του υπολογιστή του. Όλα τα χρώματα των ιπτάμενων μπαλών εκτός των λευκών και των μαύρων, δίνουν στον παίκτη 1 πόντο. Οι λευκές μπάλες δίνουν 10 πόντους. Οι μαύρες μπάλες δεν δίνουν πόντους, αντιθέτως, χάνει μία ζωή εάν κατά λάθος ο παίκτης πυροβολήσει κάποια από αυτές. Ο παίκτης έχει συνολικά 2 επιπλέον ζωές. Εάν δηλαδή πυροβολήσει 3 μαύρες ιπτάμενες μπάλες, το παιχνίδι τελειώνει, δείχνοντάς μία διεπιφάνεια «GAME OVER», το σκορ που πέτυχε, καθώς και την προτροπή να επιστρέψει στην αρχική διεπιφάνεια ή να κάνει τερματισμό του παιχνιδιού, πατώντας τα αντίστοιχα πλήκτρα, R ή Q . Όσο περνάει ο χρόνος, οι ιπτάμενες μπάλες κινούνται όλο και πιο γρήγορα. Εάν ο παίκτης καταφέρει να περάσει όλα τα επίπεδα και να φτάσει από το ξημέρωμα έως το τέλος της νύχτας, εμφανίζεται η διεπιφάνεια που αναγράφει πως η ώρα πέρασε, δείχνοντάς έτσι ότι ολοκληρώθηκε ο κύκλος του παιχνιδιού. Αναγράφεται επίσης το σκορ που επιτεύχθηκε, καθώς και οι ίδιες με τις παραπάνω προτροπές επιστροφής στο αρχικό μενού ή τερματισμού. Κατά τη διάρκεια του παιχνιδιού υπάρχει μουσική, ήχοι αλληλεπίδρασης με τις ιπτάμενες μπάλες, ήχοι σκορ, προειδοποιητικός ήχος ρολογιού λίγο πριν τη λήξη του χρόνου, ήχος «GAME OVER» και ήχος τερματισμού του παιχνιδιού.

Τα γραφικά, ο κώδικας και η υλοποίηση του Ball Predator από την αρχή

Για την υλοποίηση του «Ball Predator», χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python και η βιβλιοθήκη pygame. Το εξώφυλλο του τίτλου του παιχνιδιού, δημιουργήθηκε με AI μέσω του site Copilot. Τα υπόλοιπα μη διανυσματικά γραφικά, αποτελούνται από εικόνες κατεβασμένες από το διαδίκτυο, ελεύθερες από πνευματικά δικαιώματα, με μία επιπλέον μικρή επεξεργασία στο Photoshop και τυπική βελτίωση ανάλυσης σε online HD converters. Τα ηχητικά εφέ και η μουσική που επιλέχθηκαν είναι επίσης ελεύθερα δικαιωμάτων, κατεβασμένα από το site «pixabay».

Αφού έγινε εισαγωγή των απαραίτητων βιβλιοθηκών(pygame, μαθηματικές συναρτήσεις, συναρτήσεις τυχαιότητας και η sqlite3 για τη βάση δεδομένων) στο πρόγραμμα, ακολουθεί προσπάθεια για άνοιγμα σύνδεσης με τη βάση δεδομένων, μέσω ενός cursor. Στη συνέχεια η συνάρτηση **create_table() ,** υλοποιεί τη δημιουργία του πίνακα με τα σκορ. Η συνάρτηση **data_entry(v1,v2)**, υλοποιεί την εισαγωγή σκορ στη βάση δεδομένων. Έπειτα η **best_score_with_data_base(),** υλοποιεί την επιστροφή του μεγαλύτερου σκορ και τέλος καλείται η **create_table()** για να δημιουργηθεί ο πίνακας της βάσης δεδομένων με τα σκορ.

```
import pygame
import sys
import random
from math import *
import sqlite3
from sqlite3 import Error




try:
 conn = sqlite3.connect('pythonDB.db')
 c = conn.cursor()
 print("Connection to Python.db successfully")

except Error as e:
        print(f"The error '{e}' occured, no connection ith Python.db")
```

```
def create_table():
   try:

    c.execute('CREATE TABLE IF NOT EXISTS Scores (id INTEGER PRIMARY
KEY AUTOINCREMENT, score INTEGER)')

    c.execute("INSERT INTO Scores (id, score) VALUES(?, ?)", (0,0))

    print("Scores table created successfully")

   except Error as e:
        print(f"The error '{e}' occured, table Scores not created")

   conn.commit()


def data_entry(v1,v2):
  try:
   cu = conn.cursor()
   print("Connection cursor successfully")

  except Error as e:
        print(f"The error '{e}' occured")

  result=cu.execute("SELECT MAX(id) AS maximum FROM Scores")
  result = cu.fetchall()

  for i in result:
    maximum = (max(i) )
    v1=maximum+1
    try:
     c.execute("INSERT INTO Scores (id, score) VALUES(?, ?)", (v1,v2))

     print("Score registered successfully in python.db")

    except Error as e:
        print(f"The error '{e}' occured, score not register")

    conn.commit()



def best_score_with_data_base():

    conn = sqlite3.connect('pythonDB.db')
    c = conn.cursor()

    c.execute("SELECT MAX(score) AS maximum FROM Scores")
```

```
    result = c.fetchall()




    for i in result:
     maximum = (i[0])
     return maximum


create_table()
```

Στη συνέχεια γίνονται οι απαραίτητες αρχικοποιήσεις της pygame, των διαστάσεων της οθόνης του παιχνιδιού και του caption του παραθύρου του. Φορτώνονται επίσης στο πρόγραμμα οι εικόνες και οι ήχοι που θα χρειαστούν και γίνονται οι αρχικοποιήσεις των μεταβλητών, όπως των διαστάσεων της οθόνης, των σκορ, των χρωμάτων και των γραμματοσειρών που θα χρειαστούμε στο παιχνίδι.

```
pygame.init()
win = pygame.display.set_mode((888,780))
pygame.display.set_caption("'Ball Predator' By Anestis Kourdis")


try:

 bg = pygame.image.load("br1.png")
 bg2 = pygame.image.load("br2.png")
 bg3 = pygame.image.load("br3.png")
 bg4 = pygame.image.load("br4.png")
 bg5 = pygame.image.load("predator.jpg")

 print("level images files loaded successfully")

except Error as e:
 print(f"The error '{e}' occured on load level image")


try:

  pygame.mixer.music.load("sound.mp3")
  pygame.mixer.music.play(800)


  blow_sound = pygame.mixer.Sound("blowsup.wav")
```

```
  beep_sound =pygame.mixer.Sound("beep.mp3")
  nextlevelSound =pygame.mixer.Sound("nextlevelSound.mp3")
  gameoverSound =pygame.mixer.Sound("gameoverSound.mp3")

  clock_sound = pygame.mixer.Sound("alarm.mp3")
  bonus_sound = pygame.mixer.Sound("bonus.mp3")

  print("sound files loaded successfully")

except Error as e:
        print(f"The error '{e}' occured, on load sound files")



margin = 70
lowerBound = 70

width = 940
height = 780

display = pygame.display.set_mode((width, height))
clock = pygame.time.Clock()


score = 0
txt=1
levelCounter=1
counter=120
blackBallCounter=2
game_over=False
ending=False
game_state = "start_menu"



best_scores=[]

# Colors
black=(0,0,0)
black2=(30,30,30)
white = (230, 230, 230)
red = (220, 68, 55)
red2=(255,82,70)
green = (35, 155, 86)
yellow = (244, 208, 63)
blue = (46, 134, 193)
purple = (155, 89, 182)
orange = (243, 156, 18)
lime=(0,255,0)
```

```
saddlebrown=(130,62,13)


font = pygame.font.SysFont("Snap ITC", 25)
font2 = pygame.font.SysFont("comicsansms", 30)
font3 = pygame.font.SysFont("Snap ITC", 45)

leveltext = font2.render("Level  " + str(txt), True, yellow)
display.blit(leveltext, (740, height - lowerBound + 10))
```

Κάθε ιπτάμενη μπάλα, αποτελεί ένα αντικείμενο της κλάσης Ball, με ιδιότητες διαστάσεων πλάτους-μήκους(a, b), συντεταγμένων x, y στην οθόνη, γωνία (angle) για την ανακατεύθυνση καθώς ακουμπούν τα άκρα σε συνδυασμό με τις τυχαίες τιμές της ιδιότητας probPool, ταχύτητα(speed) και χρώμα(color). Την αναπαράσταση και τη συμπεριφορά των ιπτάμενων μπαλών και του στοχάστρου, υλοποιούν συναρτήσεις δημιουργίας διανυσματικών γραφικών, καθώς και συναρτήσεις κίνησής τους, οι οποίες εκτός από την υλοποίηση της κίνησης ελέγχουν και τη θέση των μπαλών όταν ακουμπούν στα άκρα ή υλοποιούν την ανανέωσή τους, όταν αυτές φθάνουν στο πάνω ή κάτω μέρος της σκηνής :

```
# Ball Class
class Ball:
    def __init__(self, speed):
        self.a = 75
        self.b = 75
        self.x = random.randrange(margin, width - self.a - margin)
        self.y = height - lowerBound
        self.angle = 90
        self.speed = -speed
        self.probPool = [-1, -1, -1, 0, 0, 0, 0, 1, 1, 1]
        self.color = random.choice([lime, purple, orange, yellow,
blue,saddlebrown])
```

```
 # ball around the screen
    def move(self):
        direction = random.choice(self.probPool)

        if direction == -1:
            self.angle += -10
        elif direction == 0:
```

```
            self.angle += 0
        else:
            self.angle += 10

        self.y += self.speed*sin(radians(self.angle))
        self.x += self.speed*cos(radians(self.angle))

        if (self.x + self.a > width) or (self.x < 0):
            if self.y > height/5:
                self.x -= self.speed*cos(radians(self.angle))
            else:
                self.createBall()
        if self.y + self.b < 0 or self.y > height + 30:
            self.createBall()




# Draw the ball
    def show(self):

        pygame.draw.ellipse(display, self.color, (self.x, self.y,
self.a, self.b))


    # create/reset the Ball

    def createBall(self):
        self.a = 75
        self.b = 75
        self.x = random.randrange(margin, width - self.a - margin)
        self.y = height - lowerBound
        self.angle = 90
        self.speed -= 0.0002
        self.probPool = [-1, -1, -1, 0, 0, 0, 0, 1, 1, 1]
        self.color = random.choice([black2,lime, purple, orange, white,
blue,red])
```

Με τη χρήση μιας λίστας, δημιουργούνται οι πρώτες 30 ιπτάμενες μπάλες :

```
balls = []
```

```
noBalls = 30

for i in range(noBalls):
    obj = Ball(random.choice([1, 1, 2, 2, 2, 2, 3, 3, 3, 4]))

    obj.speed=-5
    balls.append(obj)
```

Εν συνεχεία ορίστηκαν οι ακόλουθες συναρτήσεις ελέγχου για να γίνεται έλεγχος κατά τη διάρκεια εκτέλεσης του loop του παιχνιδιού, εάν σύμφωνα με τις συντεταγμένες του κέρσορα οι μπάλες έχουν χτυπηθεί, ώστε να ανανεωθούν στη σκηνή, όπως επίσης ελέγχεται και το χρώμα της μπάλας, για να γίνει η αντίστοιχη προσμέτρηση σκορ , μείωση ζωών και αναπαραγωγής ήχου αλληλεπίδρασής με τις μπάλες :

```
# Check if Ball is bursted
    def burst(self):
        global score
        global blackBallCounter
        pos = pygame.mouse.get_pos()

        if shootOnBall(self.x, self.y, self.a, self.b, pos):

            if self.color==black2:
                blow_sound.play()
                beep_sound.play()
                blackBallCounter=blackBallCounter-1
                score -= 1
            if self.color==(white):
                blow_sound.play()
                bonus_sound.play()
                score += 14

            score += 1
            blow_sound.play()



            self.createBall()
```

```
def shootOnBall(x, y, a, b, pos):


       if (x < pos[0] < x + a) and (y < pos[1] < y + b):
        return True
       else:
        return False
```

Συνεχίζοντας, η συνάρτηση **Mouse_Point()**, δημιουργεί πάνω στις συντεταγμένες του κέρσορα του ποντικιού το γραφικό του στοχάστρου, το οποίο τον ακολουθεί όταν αυτός κινείται. Το γραφικό του στοχάστρου είναι μαύρο, εκτός εάν βρίσκεται στις ίδιες συντεταγμένες με κάποια μπάλα. Στην περίπτωση αυτή γίνεται κόκκινο :

```
def Mouse_Point():

    pos = pygame.mouse.get_pos()
    r = 25
    l = 20
    color = black
    for i in range(noBalls):
        if shootOnBall(balls[i].x, balls[i].y, balls[i].a, balls[i].b,
pos)  :
            color = red2


    pygame.draw.line(display, color, (pos[0], pos[1] - l/2), (pos[0],
pos[1] - l), 4)
    pygame.draw.line(display, color, (pos[0] + l/2, pos[1]), (pos[0] +
l, pos[1]), 4)
    pygame.draw.line(display, color, (pos[0], pos[1] + l/2), (pos[0],
pos[1] + l), 4)
    pygame.draw.line(display, color, (pos[0] - l/2, pos[1]), (pos[0] -
l, pos[1]), 4)
```

Η συνάρτηση **Caption_for_scores()**, δημιουργεί μέσω της βιβλιοθήκης pygame στο κάτω μέρος της σκηνής, ένα μαύρο τετράγωνο το οποίο αποτελεί τη λεζάντα πάνω στην οποία θα αναγράφεται το σκορ, οι ζωές και το επίπεδο στο ο οποίο ο παίκτης βρίσκεται :

```
def Caption_for_scores():

      pygame.draw.rect(display,black , (0, height - lowerBound, width,
lowerBound))
```

Η συνάρτηση **show_Score_Level_BlackBallsBursted()**, θα υλοποιεί την εμφάνιση του τρέχοντος σκορ, θα ελέγχει τον χρόνο αλλάζοντας σύμφωνα με προκαθορισμένες τιμές την αλλαγή της ένδειξης του επιπέδου και τέλος, τη μείωση των ζωών σύμφωνα με τον μετρητή **blackBallCounter**. Το εικονίδιο της κάθε ζωής δεν αποτελεί διανυσματικό ή ψηφιογραφικό γραφικό, αλλα σύνολο χαρακτήρων “Q” με γραμματοσειρά SNAP ITC **:**

```
def show_Score_Level_BlackBallsBursted():
      global levelCounter

      scoreText = font.render("SCORE : " + str(score), True, white)
      display.blit(scoreText, (40, height - lowerBound + 10))



      if counter>=90:

       leveltext = font2.render("Level 1", True, yellow)
       display.blit(leveltext, (740, height - lowerBound + 10))


      if counter<90 and counter>=60:

       leveltext = font2.render("Level 2", True, yellow)
       display.blit(leveltext, (740, height - lowerBound + 10))

      if counter<60 and counter>=30:

       leveltext = font2.render("Level 3", True, yellow)
       display.blit(leveltext, (740, height - lowerBound + 10))

      if counter<30 and counter>=1:

        leveltext = font2.render("Level 4", True, yellow)
        display.blit(leveltext, (740, height - lowerBound + 10))


      if blackBallCounter==2:
```

```
        blackBalls = font3.render("Lives : Q Q", True, red)
        display.blit(blackBalls, (350, height - lowerBound + 10))

    if blackBallCounter==1:

        blackBalls = font3.render("Lives : Q", True, red)
        display.blit(blackBalls, (350, height - lowerBound + 10))

    if blackBallCounter==0:

        blackBalls = font3.render("Lives :  ", True, red)
        display.blit(blackBalls, (350, height - lowerBound + 10))
```

Η συνάρτηση **level()**, ελέγχει τις τιμές του χρόνου και αλλάζει το ανάλογο γραφικό backround του επιπέδου, σύμφωνα με τον μετρητή **counter :**

```
def level():


    if counter>90 :

       display.blit(bg, (0, 0))

    if counter<90 and counter>=60:

       display.blit(bg2, (0, 0))

    if counter<60 and counter>=30:

        display.blit(bg3, (0, 0))

    if counter<30 and counter>=1:

        display.blit(bg4, (0, 0))
```

Οι επόμενες 2 συναρτήσεις χρησιμοποιούνται για το κλείσιμο της εφαρμογής του παιχνιδιού και αντίστοιχα για τον έλεγχο του χρόνου ώστε να αναπαραχθεί ο προειδοποιητικός ήχος λήξης του :

```
def close():
```

```
     pygame.quit()
     sys.exit()

def clock_():

 if counter<10 and counter>1:
    clock_sound.play(1)
```

Οι συναρτήσεις **draw_start_menu()**, **draw_game_over_screen()** και **draw_ending_screen()** υλοποιούν την αρχική διεπιφάνεια, την διεπιφάνεια του «GAME OVER , καθώς και την διεπιφάνεια τερματισμού του παιχνιδιού. Δημιουργούν αρχικά μαύρο φόντο, έπειτα γραμματοσειρές και στη συνέχεια απεικονίζουν σε υπολογισμένες αναλογικά συντεταγμένες της οθόνης, τις ενδείξεις μέσω κειμένου :

```
def draw_start_menu():
    display.fill((0, 0, 0))
    display.blit(bg5, (0, 0))


    font2 = pygame.font.SysFont('Arial', 28)
    font3 = pygame.font.SysFont('Snap ITC', 45)


    start_button = font2.render('Press SPACE to Start New Game', True,
(255, 255, 255))
    highsc=best_score_with_data_base()
    highscores_= font3.render("Best Score : "+str(highsc), True,
(yellow))


    display.blit(start_button, (width/1.8 - start_button.get_width()/2,
height/1.1 + start_button.get_height()/2))
    display.blit(highscores_, (width/2 - start_button.get_width()/2,
height/1.2 + start_button.get_height()/2))

    pygame.display.update()



def draw_game_over_screen():

   display.fill((0, 0, 0))
   font = pygame.font.SysFont('Snap ITC', 50)
   font2 = pygame.font.SysFont('Arial', 40)
   title = font.render('Game Over', True, (blue))
   restart_button = font2.render('R - Restart', True, (255, 255, 255))
```

```
   quit_button = font2.render('Q - Quit', True, (255, 255, 255))
   last_scor = font2.render('Your Score : '+str(last_score), True,
(185, 89, 152))

   display.blit(title, (width/2 - title.get_width()/2, height/2.2 -
title.get_height()/3))
   display.blit(restart_button, (width/2 -
restart_button.get_width()/2, height/1.9 +
restart_button.get_height()))
   display.blit(quit_button, (width/2 - quit_button.get_width()/2,
height/2 + quit_button.get_height()/2))
   display.blit(last_scor, (width/2 - last_scor.get_width()/2,
height/1.3 + last_scor.get_height()/2))

   pygame.display.update()




def draw_ending_screen():

   display.fill((0, 0, 0))
   font = pygame.font.SysFont('Snap ITC', 60)
   font2 = pygame.font.SysFont('Arial', 50)
   title = font.render('The time has passed !!!', True, (orange))
   restart_button = font2.render('R - Restart', True, (255, 255, 255))
   quit_button = font2.render('Q - Quit', True, (255, 255, 255))
   last_scor = font2.render('Your Score : '+str(last_score), True,
(185, 89, 152))

   display.blit(title, (width/2 - title.get_width()/2, height/2.2 -
title.get_height()/3))
   display.blit(restart_button, (width/2 -
restart_button.get_width()/2, height/1.9 +
restart_button.get_height()))
   display.blit(quit_button, (width/2 - quit_button.get_width()/2,
height/2 + quit_button.get_height()/2))
   display.blit(last_scor, (width/2 - last_scor.get_width()/2,
height/1.3 + last_scor.get_height()/2))

   pygame.display.update()
```

Οι συναρτήσεις **append_Score_lines_to_file(**file_path, lines_to_append**)** και **highscore_with_textFile()**, υλοποιούν την αποθήκευση του τελευταίου σκορ σε αρχείο κειμένου και την εύρεση του μεγαλύτερου σκορ μέσα στο αρχείο κειμένου αντίστοιχα :

```
def append_Score_lines_to_file(file_path, lines_to_append):

    try:
        with open(file_path, 'a') as file:
            file.write('\n'.join(lines_to_append) + '\n')
        print('Score succesfully registered in '+ str({file_path}))
    except Exception as e:
        print(" Score register Error: "+str({e}))


def highscore_with_textFile():
   file_highscore = open('scores.txt' , 'r')
   scores = []
   Rank=[]
   for line in file_highscore.readlines():
      score_info = line.split()
      scores.append((int (score_info[0])))

   scores.sort(key = lambda x:x, reverse = True)

   for score in scores[:3]:

      Rank.append(score)

   #return best score
   return Rank[0]
```

Τέλος, η συνάρτηση **game(),** αποτελεί το loop του παιχνιδιού. Μέσα στη συνάρτηση αυτή εκτελούνται όλες οι συναρτήσεις του κώδικα, καταχωρείται το σκορ ή ανασύρεται το μεγαλύτερο σκορ στην/απο βάση δεδομένων και στο/από αρχείο κειμένου(η τελική έκδοση του παιχνιδιού δεν υποστηρίζει εύρεση μεγαλύτερου σκορ μέσω αρχείου κειμένου, γι' αυτό και η σχετική εντολή έχει επισημανθεί ως σχόλιο). Ορίζονται επίσης γεγονότα πλήκτρων τα οποία μπορεί να πατήσει ο παίκτης, ενώ γίνονται και διάφοροι έλεγχοι σχετικά με την κατάσταση στην οποία βρίσκεται το παιχνίδι. Υπάρχουν 4 καταστάσεις : Η "start_menu" , η "game", η "game_over" και η "ending". Ανάλογα με τα αποτελέσματα των ελέγχων, η ροή του παιχνιδιού οδηγείται προς την ανάλογη κατάσταση. Εδώ ορίζεται και το βήμα της σταδιακής αύξησης ταχύτητας των ιπτάμενων μπαλών από την αρχή έως τον τερματισμό του παιχνιδιού (for i in range(noBalls):

balls[i].speed-=0.01

balls[i].show())

```
    def game():

    global counter
    global score
    run = True
    game_state = "start_menu"
    global levelCounter
    global blackBallCounter
    global game_over
    global last_score
    global ending
    global file



    #text file best score operation
    # highscore()


    #data base best score operation
    best_score_with_data_base()


    while run:

        clock_()
        level()

        for event in pygame.event.get():

            if event.type == pygame.QUIT:
             close()

        if game_state == "start_menu":


             pygame.mixer.music.stop()
             draw_start_menu()
             keys = pygame.key.get_pressed()
             if keys[pygame.K_SPACE]:
```

```
            for i in range(noBalls):
             balls[i].speed=-5

             balls[i].y=height-lowerBound

            pygame.mixer.music.load("sound.mp3")
            pygame.mixer.music.play()
            game_state = "game"
            game_over = False
            ending=False


        elif game_state == "game":

         if event.type == pygame.MOUSEBUTTONDOWN:


                for i in range(noBalls):
                 balls[i].burst()
         for i in range(noBalls):
            balls[i].speed-=0.01
            balls[i].show()



         Mouse_Point()
         Caption_for_scores()
         show_Score_Level_BlackBallsBursted()


         if (blackBallCounter==-1 or blackBallCounter<=-2):
          game_over = True
          game_state = "game_over"
          blackBallCounter=2
          last_score=score

          file_path = 'scores.txt'
          lines_to_append = [str(last_score)]

          #save score with text file operation
          append_Score_lines_to_file(file_path, lines_to_append)

          #save score with data base operation
          n=0
          data_entry(n,last_score)
```

```
            pygame.mixer.music.stop()
            clock_sound.stop()
            pygame.mixer.music.load("gameoverSound.mp3")
            pygame.mixer.music.play()


        if (counter<1):

            ending = True
            game_state = "ending"
            counter=-1
            blackBallCounter=2

            file_path = 'scores.txt'
            lines_to_append = [str(last_score)]

            #save score with text file operation
            append_Score_lines_to_file(file_path, lines_to_append)

            #save score with data base operation
            n=0
            data_entry(n,last_score)


            pygame.mixer.music.stop()
            clock_sound.stop()
            pygame.mixer.music.load("ending.mp3")
            pygame.mixer.music.play()


        for i in range(noBalls):
            balls[i].move()

            leveltext = font2.render("Level ", True, yellow)
            display.blit(leveltext, (740, height - lowerBound +10))
            myfont=pygame.font.Font(None,50)
            text=myfont.render("TIME:"+str(int(counter)),1,(0,0,0))




        win.blit(text,(10,5))
        counter=counter-0.1
        last_score=score
```

```
            pygame.display.update()
            clock.tick(60)
            win.blit(bg, (0, 0))



        if game_state == "ending":

            draw_ending_screen()
            keys = pygame.key.get_pressed()

        if keys[pygame.K_r]:

            game_state = "start_menu"
            counter=120
            score=0
            clock_sound.stop()

        if keys[pygame.K_q]:
            pygame.quit()
            quit()


        if game_state == "game_over":

            draw_game_over_screen()
            keys = pygame.key.get_pressed()

        if keys[pygame.K_r]:
            game_state = "start_menu"
            counter=120
            last_score=score
            score=0

            clock_sound.stop()


        if keys[pygame.K_q]:
            pygame.quit()
            quit()




game()
pygame.quit()
```