import pygame

import time

import random

SIZE = 40

#BACKGROUND\_COLOR = (80, 199, 199)

#pygame locals

from pygame import KEYDOWN, QUIT, K\_ESCAPE, K\_UP, K\_DOWN, K\_RIGHT, K\_LEFT, K\_RETURN

class Apple:

def \_\_init\_\_(self, parent\_screen):

self.parent\_screen = parent\_screen

self.image = pygame.image.load("resources/apple.jpg").convert()

self.x = SIZE \* 3 #120

self.y = SIZE \* 3 #120

def draw(self):

self.parent\_screen.blit(self.image, (self.x, self.y))

pygame.display.flip()

def move(self):

# 1000/40 = 25 ... It means block can move 25 bytes || After 25th it will go out the screen/wiondow

# random.randomint(1,10) this function will return the random value between 1 to 10

self.x = random.randint(1, 24) \* SIZE

self.y = random.randint(1, 24) \* SIZE

class Game:

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

pygame.init()

# library of sound is initialized

pygame.mixer.init()

# this function is for sound background music from the starting

self.play\_background\_music()

# to open the main window of the snake gaME .... parameter will be size of the window

self.surface = pygame.display.set\_mode((1000, 1000))

self.snake = Snake(self.surface,1)

self.snake.draw()

self.apple = Apple(self.surface)

self.apple.draw()

#this function is for set the background image

def render\_background\_image(self):

bg = pygame.image.load('resources/background\_image.jpg')

self.surface.blit(bg, (0, 0))

def play\_background\_music(self):

#this code is for play the background music continuously

pygame.mixer.music.load('resources/bg\_music\_1.mp3')

# -1 means it will sound continuously and 0 means from the starting

pygame.mixer.music.play(-1, 0)

def play\_sound(self, sound):

# this method is used to play a music when snake will eat the apple

if sound == 'ding':

sound = pygame.mixer.Sound("resources/ding.mp3")

elif sound == 'crash':

sound = pygame.mixer.Sound("resources/crash.mp3")

pygame.mixer.Sound.play(sound)

def play(self):

self.render\_background\_image()

self.snake.walk()

self.apple.draw()

self.display\_score()

pygame.display.flip()

# This code for to check the collision of all the blocks of the snake

if self.is\_collision(self.snake.x[0], self.snake.y[0], self.apple.x, self.apple.y):

self.play\_sound("ding")

# this function call for increase the length of the snake

self.snake.increase\_length()

# this function call for to move apple from one place to another

self.apple.move()

# this code is for snake colliding with snake or block is colide with self snake's block

# here we have started with 2 bcz head of snake never collide with it's second block..

for i in range(3, self.snake.length):

if self.is\_collision(self.snake.x[0], self.snake.y[0], self.snake.x[i], self.snake.y[i]):

self.play\_sound('crash')

#print("Game Over...")

raise "Game Over"

def reset(self):

self.snake = Snake(self.surface, 1)

self.apple = Apple(self.surface)

def run(self, K\_ENTER=None):

# UI should be in infinity loop when it is in waiting for user input or special event like cancel button event then loop should be stop

running = True

pause = False

while running:

# when we click any button or mouse any event then it will go into this below method

for event in pygame.event.get():

if event.type == KEYDOWN:

# when you enter a escape than it should quit

if event.key == K\_ESCAPE:

running = False

# pause flag is used for when the game is over it should be paused..

# This is for when player hit the enter after game over scenario

if event.key == K\_RETURN:

pygame.mixer.music.unpause()

pause = False

# when you enter arrows than block should be go left and right

if not pause:

if event.key == K\_UP:

self.snake.move\_up()

if event.key == K\_DOWN:

self.snake.move\_down()

if event.key == K\_LEFT:

self.snake.move\_left()

if event.key == K\_RIGHT:

self.snake.move\_right()

elif event.type == QUIT:

running = False

#this is for time where we should move snake continuously and when we hit up and down aerrow then it need to just change the direction

#also this is for some of the abjects for multiple objects we have to call it multiple times, so we should make one function which named as a play

try:

if not pause:

self.play()

except Exception as e:

self.show\_game\_over()

pause = True

self.reset()

time.sleep(0.2)

def is\_collision(self, x1, y1, x2, y2):

#collition means when snake bite the apple or when apple and snake will be overlapped..

if x1 >= x2 and x1 <= x2 + SIZE:

if y1 >= y2 and y1 <= y2 + SIZE:

return True

return False

def show\_game\_over(self):

# self.surface.fill(BACKGROUND\_COLOR)

self.render\_background\_image()

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

line1 = font.render(f"Game is Over!! Your Score is : {self.snake.length}", True, (200, 200, 200))

self.surface.blit(line1,(200,300))

line2 = font.render(f"To play again press Enter!!, to Exit press Escape!! : {self.snake.length}", True, (200, 200, 200))

self.surface.blit(line2, (200, 350))

pygame.display.flip()

pygame.mixer.music.pause()

def display\_score(self):

# to display the scores

# Sysfont method is used to set the size and style of the text

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

score = font.render(f"Score : {self.snake.length}", True, (200, 200, 200))

# if we want to put anything on surface then use blit function..

self.surface.blit(score, (850, 10))

class Snake:

def \_\_init\_\_(self, parent\_screen, length):

self.parent\_screen = parent\_screen

self.block = pygame.image.load("resources/block.jpg").convert()

self.length = length

self.x = [SIZE] \* length

self.y = [SIZE] \* length

self.direction = 'down'

def draw(self):

# to again fill the background with color bcz we have to remove the blocks from previous state so we are calling fill function

# self.parent\_screen.fill(BACKGROUND\_COLOR)

for i in range(self.length):

# to draw the block

self.parent\_screen.blit(self.block, (self.x[i], self.y[i]))

# flip method will display the game window - without flip it will not show the updates

pygame.display.flip()

def increase\_length(self):

# this function for increase teh length of snake after collision occurs...

self.length += 1

# this increment for x and y bcz we are also increasing the blocks so we should increase the valuwe of x and y cordinates of block..

self.x.append(-1)

self.y.append(-1)

def move\_up(self):

# when it is up condition then we should only update y, x will not change

#self.y = self.y - 10

# after changing any variable call below method

#self.draw()

self.direction = 'up'

def move\_down(self):

# when it is up condition then we should only update y, x will not change

#self.y = self.y + 10

# after changing any variable call below method

#self.draw()

self.direction = 'down'

def move\_left(self):

# when it is up condition then we should only update y, x will not change

#self.x = self.x - 10

# after changing any variable call below method

#self.draw()

self.direction = 'left'

def move\_right(self):

# when it is up condition then we should only update y, x will not change

#self.x = self.x + 10

# after changing any variable call below method

#self.draw()

self.direction = 'right'

def walk(self):

for i in range(self.length-1, 0, -1):

#for changing the position of the block and move to ome position left/right or up/down

self.x[i] = self.x[i-1]

self.y[i] = self.y[i-1]

if self.direction == 'left':

#for move to head of the snake to left,right,up and down

self.x[0] -= SIZE

elif self.direction == 'right':

self.x[0] += SIZE

elif self.direction == 'up':

self.y[0] -= SIZE

elif self.direction == 'down':

self.y[0] += SIZE

# we have to call draw function to draw the block

self.draw()

if \_\_name\_\_ == "\_\_main\_\_":

game = Game()

game.run()

# to show the window for 5 seconds

#time.sleep(5)