import pygame

from pygame.locals import \*

from pygame import mixer

import pickle

from os import path

import time

pygame.mixer.pre\_init(44100, -16, 2, 512)

mixer.init()

pygame.init()

clock = pygame.time.Clock()

fps = 70

screen\_width = 1000

screen\_height = 1000

screen = pygame.display.set\_mode((screen\_width, screen\_height))

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

#define font

font = pygame.font.SysFont('Bauhaus 93', 70)

font\_score = pygame.font.SysFont('Bauhaus 93', 30)

#define colours

white = (255,255,255)

blue = (0,0,255)

#define game variables

tile\_size = 50

game\_over = 0

main\_menu = True

level = 7

max\_levels = 7

score = 0

lives\_used = 0

#load images

sun\_img = pygame.image.load('sun.png')

bg\_img = pygame.image.load('sky.png')

restart\_img = pygame.image.load('restart\_btn.png')

start\_img = pygame.image.load('start\_btn.png')

exit\_img = pygame.image.load('exit\_btn.png')

exit\_door = pygame.image.load('exit.png')

coin\_img = pygame.image.load('coin.png')

platform\_x\_img = pygame.image.load('platform\_x.png')

platform\_y\_img = pygame.image.load('platform\_y.png')

#loud sounds

pygame.mixer.music.load('music.wav')

pygame.mixer.music.play(-1,0.0,5000)

coin\_fx = pygame.mixer.Sound('img\_coin.wav')

coin\_fx.set\_volume(0.5)

jump\_fx = pygame.mixer.Sound('img\_jump.wav')

jump\_fx.set\_volume(0.5)

game\_over\_fx = pygame.mixer.Sound('img\_game\_over.wav')

game\_over\_fx.set\_volume(0.5)

def draw\_text(text,font,text\_col,x,y):

img = font.render(text, True, text\_col)

screen.blit(img,(x,y))

def reset\_level(level):

player.reset(100,screen\_height - 130)

blob\_group.empty()

lava\_group.empty()

exit\_group.empty()

platform\_group.empty()

coin\_group.empty()

if path.exists((f'level{level}\_data')):

pickle\_in = open(f'level{level}\_data', 'rb')

world\_data = pickle.load(pickle\_in)

world = World(world\_data)

return world

class Button():

def \_\_init\_\_(self, x, y, image):

self.image = image

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

self.clicked = False

def draw(self):

action = False

#get mouse position

pos = pygame.mouse.get\_pos()

#check mouseover and clicked conditions

if self.rect.collidepoint(pos):

if pygame.mouse.get\_pressed()[0] == 1 and self.clicked == False:

action = True

self.clicked = True

if pygame.mouse.get\_pressed()[0] == 0:

self.clicked = False

#draw button

screen.blit(self.image, self.rect)

return action

class Player():

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

self.reset(x, y)

def update(self, game\_over):

dx = 0

dy = 0

walk\_cooldown = 5

col\_thresh = 20

if game\_over == 0:

#get keypresses

key = pygame.key.get\_pressed()

if key[pygame.K\_SPACE] and self.jumped == False and self.in\_air == False:

jump\_fx.play()

self.vel\_y = -17

self.jumped = True

if key[pygame.K\_SPACE] == False:

self.jumped = False

if key[pygame.K\_LEFT]:

dx -= 5

self.counter += 1

self.direction = -1

if key[pygame.K\_RIGHT]:

dx += 5

self.counter += 1

self.direction = 1

if key[pygame.K\_LEFT] == False and key[pygame.K\_RIGHT] == False:

self.counter = 0

self.index = 0

if self.direction == 1:

self.image = self.images\_right[self.index]

if self.direction == -1:

self.image = self.images\_left[self.index]

#handle animation

if self.counter > walk\_cooldown:

self.counter = 0

self.index += 1

if self.index >= len(self.images\_right):

self.index = 0

if self.direction == 1:

self.image = self.images\_right[self.index]

if self.direction == -1:

self.image = self.images\_left[self.index]

#add gravity

self.vel\_y += 1

if self.vel\_y > 10:

self.vel\_y = 10

dy += self.vel\_y

#check for collision

self.in\_air = True

for tile in world.tile\_list:

#check for collision in x direction

if tile[1].colliderect(self.rect.x + dx, self.rect.y, self.width, self.height):

dx = 0

#check for collision in y direction

if tile[1].colliderect(self.rect.x, self.rect.y + dy, self.width, self.height):

#check if below the ground i.e. jumping

if self.vel\_y < 0:

dy = tile[1].bottom - self.rect.top

self.vel\_y = 0

#check if above the ground i.e. falling

elif self.vel\_y >= 0:

dy = tile[1].top - self.rect.bottom

self.vel\_y = 0

self.in\_air = False

#check for collision with enemies

if pygame.sprite.spritecollide(self, blob\_group, False):

game\_over = -1

lives\_used = +1

game\_over\_fx.play()

#check for collision with lava

if pygame.sprite.spritecollide(self, lava\_group, False):

game\_over = -1

lives\_used = +1

game\_over\_fx.play()

#check for collision with exit

if pygame.sprite.spritecollide(self, exit\_group, False):

game\_over = +1

#check for collision with platforms

for platform in platform\_group:

#collision in the x direction

if platform.rect.colliderect((self.rect.x + dx, self.rect.y, self.width, self.height)):

dx = 0

#collisions in the y direction

if platform.rect.colliderect((self.rect.x, self.rect.y + dy, self.width, self.height)):

#check if below platform

if abs((self.rect.top + dy) - platform.rect.bottom) < col\_thresh:

self.vel\_y = 0

dy = platform.rect.bottom - self.rect.top

#check if above platform

elif abs((self.rect.bottom + dy) - platform.rect.top) < col\_thresh:

self.rect.bottom = platform.rect.top - 1

self.in\_air = False

dy = 0

#move sideways with platform

if platform.move\_x != 0:

self.rect.x += platform.move\_direction

#update player coordinates

self.rect.x += dx

self.rect.y += dy

elif game\_over == -1:

self.image = self.dead\_image

draw\_text('UNLUCKY RETRY!!', font, blue,(screen\_width // 2) - 200, screen\_height// 2)

if self.rect.y > 200:

self.rect.y -= 5

#draw player onto screen

screen.blit(self.image, self.rect)

return game\_over

def reset(self, x, y):

self.images\_right = []

self.images\_left = []

self.index = 0

self.counter = 0

for num in range(1, 5):

img\_right = pygame.image.load('guy1.png')

img\_right = pygame.transform.scale(img\_right, (40, 80))

img\_left = pygame.transform.flip(img\_right, True, False)

self.images\_right.append(img\_right)

self.images\_left.append(img\_left)

self.dead\_image = pygame.image.load('ghost.png')

self.image = self.images\_right[self.index]

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

self.width = self.image.get\_width()

self.height = self.image.get\_height()

self.vel\_y = 0

self.jumped = False

self.direction = 0

self.in\_air = True

class World():

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

self.tile\_list = []

#load images

glowstone\_img = pygame.image.load('glowstone.png')

emerald\_img = pygame.image.load('emerald.png')

row\_count = 0

for row in data:

col\_count = 0

for tile in row:

if tile == 1:

img = pygame.transform.scale(glowstone\_img, (tile\_size, tile\_size))

img\_rect = img.get\_rect()

img\_rect.x = col\_count \* tile\_size

img\_rect.y = row\_count \* tile\_size

tile = (img, img\_rect)

self.tile\_list.append(tile)

if tile == 2:

img = pygame.transform.scale(emerald\_img, (tile\_size, tile\_size))

img\_rect = img.get\_rect()

img\_rect.x = col\_count \* tile\_size

img\_rect.y = row\_count \* tile\_size

tile = (img, img\_rect)

self.tile\_list.append(tile)

if tile == 3:

blob = Enemy(col\_count \* tile\_size, row\_count \* tile\_size + 15)

blob\_group.add(blob)

if tile == 4:

platform = Platform(col\_count \* tile\_size, row\_count \* tile\_size, 1, 0)

platform\_group.add(platform)

if tile == 5:

platform = Platform(col\_count \* tile\_size, row\_count \* tile\_size,0,1 )

platform\_group.add(platform)

if tile == 6:

lava = Lava(col\_count \* tile\_size, row\_count \* tile\_size + (tile\_size // 2))

lava\_group.add(lava)

if tile == 7:

coin = Coin(col\_count \* tile\_size // 2, row\_count \* tile\_size + (tile\_size // 2))

coin\_group.add(coin)

if tile == 8:

exit = Exit(col\_count \* tile\_size, row\_count \* tile\_size -(tile\_size // 2))

exit\_group.add(exit)

col\_count += 1

row\_count += 1

def draw(self):

for tile in self.tile\_list:

screen.blit(tile[0], tile[1])

class Enemy(pygame.sprite.Sprite):

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

pygame.sprite.Sprite.\_\_init\_\_(self)

self.image = pygame.image.load('blob.png')

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

self.move\_direction = 1

self.move\_counter = 0

def update(self):

self.rect.x += self.move\_direction

self.move\_counter += 1

if abs(self.move\_counter) > 50:

self.move\_direction \*= -1

self.move\_counter \*= -1

class Platform(pygame.sprite.Sprite):

def \_\_init\_\_(self, x, y, move\_x, move\_y):

pygame.sprite.Sprite.\_\_init\_\_(self)

img = pygame.image.load('platform.png')

self.image = pygame.transform.scale(img, (tile\_size, tile\_size // 2))

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

self.move\_direction = 1

self.move\_counter = 0

self.move\_x = move\_x

self.move\_y = move\_y

def update(self):

self.rect.x += self.move\_direction \* self.move\_x

self.rect.y += self.move\_direction \* self.move\_y

self.move\_counter += 1

if abs(self.move\_counter) > 50:

self.move\_direction \*= -1

self.move\_counter \*= -1

class Lava(pygame.sprite.Sprite):

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

pygame.sprite.Sprite.\_\_init\_\_(self)

img = pygame.image.load('lava.png')

self.image = pygame.transform.scale(img, (tile\_size, tile\_size // 2))

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

class Coin(pygame.sprite.Sprite):

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

pygame.sprite.Sprite.\_\_init\_\_(self)

img = pygame.image.load('coin.png')

self.image = pygame.transform.scale(img, (tile\_size// 2 , tile\_size // 2))

self.rect = self.image.get\_rect()

self.rect.center = (x,y)

class Exit(pygame.sprite.Sprite):

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

pygame.sprite.Sprite.\_\_init\_\_(self)

img = pygame.image.load('Exit.png')

self.image = pygame.transform.scale(img, (tile\_size, int(tile\_size \* 1.5)))

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

player = Player(100, screen\_height - 130)

blob\_group = pygame.sprite.Group()

platform\_group = pygame.sprite.Group()

lava\_group = pygame.sprite.Group()

coin\_group = pygame.sprite.Group()

exit\_group = pygame.sprite.Group()

#create coin

score\_coin = Coin(tile\_size // 2, tile\_size // 2)

coin\_group.add(score\_coin)

if path.exists((f'level{level}\_data')):

pickle\_in = open(f'level{level}\_data', 'rb')

world\_data = pickle.load(pickle\_in)

world = World(world\_data)

#create buttons

restart\_button = Button(screen\_width // 2 - 50, screen\_height // 2 + 100, restart\_img)

start\_button = Button(screen\_width // 2 - 350, screen\_height //2, start\_img)

exit\_button = Button(screen\_width // 2 + 150, screen\_height //2, exit\_img)

run = True

while run:

clock.tick(fps)

screen.blit(bg\_img, (0, 0))

screen.blit(sun\_img, (100, 100))

if main\_menu == True:

if exit\_button.draw():

run = False

if start\_button.draw():

main\_menu = False

else:

world.draw()

if game\_over == 0:

blob\_group.update()

platform\_group.update()

#update score

#check if coin is collected

if pygame.sprite.spritecollide(player, coin\_group, True ):

score += 1

coin\_fx.play()

draw\_text('X ' + str(score), font\_score, white, tile\_size - 10, 10 )

blob\_group.draw(screen)

platform\_group.draw(screen)

lava\_group.draw(screen)

coin\_group.draw(screen)

exit\_group.draw(screen)

game\_over = player.update(game\_over)

#if player has died

if game\_over == -1:

if restart\_button.draw():

world\_data = []

world = reset\_level(level)

game\_over = 0

score = 0

if game\_over == 1:

#reset game and go to next level

level += 1

if level <= max\_levels:

#reset level

world\_data = []

world = reset\_level(level)

game\_over = 0

else:

draw\_text("YOU WIN", font, blue, (screen\_width // 2 ) - 140, screen\_height // 2 )

#restart game

if restart\_button.draw():

level = 1

world\_data = []

world = reset\_level(level)

game\_over = 0

score = 0

for event in pygame.event.get():

if event.type == pygame.QUIT:

run = False

pygame.display.update()

pygame.quit()