#This introduces the ability to scroll the world sideways

import pygame

BLACK = ( 0, 0, 0)

WHITE = ( 255, 255, 255)

BLUE = ( 0, 0, 255)

RED = ( 255, 0, 0)

GREEN = ( 0, 255, 0)

SCREEN\_WIDTH = 800

SCREEN\_HEIGHT = 600

class Player(pygame.sprite.Sprite):

"""

This class represents the bar at the bottom that the player controls.

"""

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

""" Constructor function """

super().\_\_init\_\_()

width = 40

height = 60

self.image = pygame.Surface([width, height])

self.image.fill(RED)

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

self.change\_x = 0

self.change\_y = 0

self.level = None

def update(self):

""" Move the player. """

self.calc\_grav()

self.rect.x += self.change\_x

block\_hit\_list = pygame.sprite.spritecollide(self, self.level.platform\_list, False)

for block in block\_hit\_list:

if self.change\_x > 0:

self.rect.right = block.rect.left

elif self.change\_x < 0:

self.rect.left = block.rect.right

self.rect.y += self.change\_y

block\_hit\_list = pygame.sprite.spritecollide(self, self.level.platform\_list, False)

for block in block\_hit\_list:

if self.change\_y > 0:

self.rect.bottom = block.rect.top

elif self.change\_y < 0:

self.rect.top = block.rect.bottom

self.change\_y = 0

def calc\_grav(self):

""" Calculate effect of gravity. """

if self.change\_y == 0:

self.change\_y = 1

else:

self.change\_y += .35

if self.rect.y >= SCREEN\_HEIGHT - self.rect.height and self.change\_y >= 0:

self.change\_y = 0

self.rect.y = SCREEN\_HEIGHT - self.rect.height

def jump(self):

""" Called when user hits 'jump' button. """

self.rect.y += 2

platform\_hit\_list = pygame.sprite.spritecollide(self, self.level.platform\_list, False)

self.rect.y -= 2

if len(platform\_hit\_list) > 0 or self.rect.bottom >= SCREEN\_HEIGHT:

self.change\_y = -10

def go\_left(self):

""" Called when the user hits the left arrow. """

self.change\_x = -6

def go\_right(self):

""" Called when the user hits the right arrow. """

self.change\_x = 6

def stop(self):

""" Called when the user lets off the keyboard. """

self.change\_x = 0

class Platform(pygame.sprite.Sprite):

""" Platform the user can jump on """

def \_\_init\_\_(self, width, height):

super().\_\_init\_\_()

self.image = pygame.Surface([width, height])

self.image.fill(GREEN)

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

class Level():

""" This is a generic super-class used to define a level.

Create a child class for each level with level-specific

info. """

platform\_list = None

enemy\_list = None

# How far this world has been scrolled left/right

world\_shift = 0

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

""" Constructor. Pass in a handle to player. Needed for when moving

platforms collide with the player. """

self.platform\_list = pygame.sprite.Group()

self.enemy\_list = pygame.sprite.Group()

self.player = player

# Update everythign on this level

def update(self):

""" Update everything in this level."""

self.platform\_list.update()

self.enemy\_list.update()

def draw(self, screen):

""" Draw everything on this level. """

screen.fill(BLUE)

self.platform\_list.draw(screen)

self.enemy\_list.draw(screen)

def shift\_world(self, shift\_x):

""" When the user moves left/right and we need to scroll everything: """

# Keep track of the shift amount

self.world\_shift += shift\_x

# Go through all the sprite lists and shift

for platform in self.platform\_list:

platform.rect.x += shift\_x

for enemy in self.enemy\_list:

enemy.rect.x += shift\_x

class Level\_01(Level):

""" Definition for level 1. """

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

""" Create level 1. """

Level.\_\_init\_\_(self, player)

self.level\_limit = -1000

level = [[210, 70, 500, 500],

[210, 70, 800, 400],

[210, 70, 1000, 500],

[210, 70, 1120, 280],

]

for platform in level:

block = Platform(platform[0], platform[1])

block.rect.x = platform[2]

block.rect.y = platform[3]

block.player = self.player

self.platform\_list.add(block)

class Level\_02(Level):

""" Definition for level 2. """

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

""" Create level 1. """

Level.\_\_init\_\_(self, player)

self.level\_limit = -1000

level = [[210, 30, 450, 570],

[210, 30, 850, 420],

[210, 30, 1000, 520],

[210, 30, 1120, 280],

]

for platform in level:

block = Platform(platform[0], platform[1])

block.rect.x = platform[2]

block.rect.y = platform[3]

block.player = self.player

self.platform\_list.add(block)

""" Main Program """

pygame.init()

size = [SCREEN\_WIDTH, SCREEN\_HEIGHT]

screen = pygame.display.set\_mode(size)

pygame.display.set\_caption("Side-scrolling Platformer")

# Create the player

player = Player()

# Create all the levels

level\_list = []

level\_list.append(Level\_01(player))

level\_list.append(Level\_02(player))

current\_level\_no = 0

current\_level = level\_list[current\_level\_no]

active\_sprite\_list = pygame.sprite.Group()

player.level = current\_level

player.rect.x = 340

player.rect.y = SCREEN\_HEIGHT - player.rect.height

active\_sprite\_list.add(player)

done = False

clock = pygame.time.Clock()

# -------- Main Program Loop -----------

while not done:

for event in pygame.event.get():

if event.type == pygame.QUIT:

done = True

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

player.go\_left()

if event.key == pygame.K\_RIGHT:

player.go\_right()

if event.key == pygame.K\_UP:

player.jump()

if event.type == pygame.KEYUP:

if event.key == pygame.K\_LEFT and player.change\_x < 0:

player.stop()

if event.key == pygame.K\_RIGHT and player.change\_x > 0:

player.stop()

# Update the player.

active\_sprite\_list.update()

# Update items in the level

current\_level.update()

# If the player gets near the right side, shift the world left (-x)

if player.rect.right >= 500:

diff = player.rect.right - 500

player.rect.right = 500

current\_level.shift\_world(-diff)

# If the player gets near the left side, shift the world right (+x)

if player.rect.left <= 120:

diff = 120 - player.rect.left

player.rect.left = 120

current\_level.shift\_world(diff)

# If the player gets to the end of the level, go to the next level

current\_position = player.rect.x + current\_level.world\_shift

if current\_position < current\_level.level\_limit:

player.rect.x = 120

if current\_level\_no < len(level\_list)-1:

current\_level\_no += 1

current\_level = level\_list[current\_level\_no]

player.level = current\_level

current\_level.draw(screen)

active\_sprite\_list.draw(screen)

clock.tick(60)

pygame.display.flip()

pygame.quit()