\*\*Install Python\*\*

1. Download and install Python 3.7 or higher from https://www.python.org/downloads/

2. Add Python to your system PATH

\*\*Install Pygame\*\*

3. Open command prompt/terminal

4. Run:

```

pip install pygame

```

This will install the Pygame library

\*\*Create Project Folder\*\*

5. Create a folder for your project

6. Add the game code (.py file) to this folder

7. Create subfolders for assets like images, sounds, etc and add those files

\*\*Import Required Modules\*\*

The game code needs the following imports:

```python

import pygame

import random

```

\*\*Initialize Pygame\*\*

8. Initialize Pygame at the start of the code:

```python

pygame.init()

```

\*\*Run the Game\*\*

9. Open command prompt/terminal and navigate to the project folder

10. Run the game file:

```

python game.py

```

11. The game window should now open and the game should run

Code

import pygame

import random

# Initialize pygame

pygame.init()

# Game screen

SCREEN\_WIDTH = 800

SCREEN\_HEIGHT = 600

screen = pygame.display.set\_mode((SCREEN\_WIDTH, SCREEN\_HEIGHT))

pygame.display.set\_caption("Side Scroller")

# Colors

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

RED = (255, 0, 0)

# Game fonts

font = pygame.font.SysFont("Arial", 30)

font\_small = pygame.font.SysFont("Arial", 20)

# Game variables

scroll\_thresh = 200

gravity = 1

# Player variables

current\_level= 1

player\_lives = 3

player\_health = 100

player\_max\_health = 100

projectile\_damage = 25

# Enemy variables

enemy\_health = 100

enemy\_damage = 25

# Levels

level\_1 = [

"XXXXXXXXXXXX",

"X X",

"X X",

"X X",

"X E X",

"X X X",

"X XXXXXX",

"X X",

"XXXXXXXXXXXX"]

level\_2 = [

"XXXXXXXXXXXXXX",

"X X",

"X X",

"X X",

"X E X",

"X X",

"X X",

"X XXXX X",

"X X",

"XXXXXXXXXXXXXX"]

level\_3 = [

"XXXXXXXXXXXXXXXXX",

"X X",

"X X",

"X X",

"X X",

"X X",

"X E X",

"X X",

"X XXXXXXXXX X",

"X X",

"XXXXXXXXXXXXXXXXX"]

levels = [level\_1, level\_2, level\_3]

# Load images

bg = pygame.image.load("bg.png")

player\_img = pygame.image.load("player.png")

projectile\_img = pygame.image.load("projectile.png")

enemy\_img = pygame.image.load("enemy.png")

health\_pickup\_img = pygame.image.load("health.png")

ammo\_pickup\_img = pygame.image.load("ammo.png")

# Sprite classes

class Player(pygame.sprite.Sprite):

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

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

self.image = player\_img

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

self.rect.center = (x, y)

self.health = player\_health

self.max\_health = player\_max\_health

self.direction = 1

self.vel\_y = 0

self.jump = False

self.alive = True

def update(self):

dx = 0

dy = 0

walk\_speed = 10

jump\_speed = 15

key = pygame.key.get\_pressed()

if key[pygame.K\_SPACE] and self.alive:

self.vel\_y = -jump\_speed

self.jump = True

if key[pygame.K\_LEFT]:

dx -= walk\_speed

self.direction = -1

if key[pygame.K\_RIGHT]:

dx += walk\_speed

self.direction = 1

def apply\_gravity(self):

dx=0

dy=0

# Gravity

self.vel\_y += gravity

if self.vel\_y > 10:

self.vel\_y = 10

dy += self.vel\_y

# Prevent going off edges

if self.rect.left + dx < 0:

dx = -self.rect.left

if self.rect.right + dx > SCREEN\_WIDTH:

dx = SCREEN\_WIDTH - self.rect.right

# Check collision with floor

if self.rect.bottom + dy > SCREEN\_HEIGHT - 110:

dy = SCREEN\_HEIGHT - 110 - self.rect.bottom

self.jump = False

# Update player position

self.rect.x += dx

self.rect.y += dy

def shoot(self):

if self.alive:

if self.direction == 1:

return Projectile(self.rect.centerx + 25, self.rect.centery - 10, 10, projectile\_damage)

else:

return Projectile(self.rect.centerx - 25, self.rect.centery - 10, -10, projectile\_damage)

else:

return None

def take\_damage(self, amount):

if self.alive:

self.health -= amount

if self.health <= 0:

self.health = 0

self.alive = False

class Projectile(pygame.sprite.Sprite):

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

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

self.image = projectile\_img

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

self.rect.center = (x, y)

self.vx = vx

self.damage = damage

def update(self):

self.rect.x += self.vx

if self.rect.right < 0 or self.rect.left > SCREEN\_WIDTH:

self.kill()

class Enemy(pygame.sprite.Sprite):

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

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

self.image = enemy\_img

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

self.rect.center = (x, y)

self.vx = -3

self.health = enemy\_health

def update(self):

self.rect.x += self.vx

if self.rect.right < 0:

self.kill()

def take\_damage(self, amount):

self.health -= amount

if self.health <= 0:

self.kill()

class HealthPickup(pygame.sprite.Sprite):

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

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

self.image = health\_pickup\_img

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

self.rect.center = (x, y)

def update(self):

if pygame.sprite.collide\_rect(self, player):

if player.health < player.max\_health:

player.health += 25

self.kill()

# Tile class

class Tile(pygame.sprite.Sprite):

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

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

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

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

# Level class

class Level():

def \_\_init\_\_(self, level\_data, surface):

self.display\_surface = surface

# Level setup

self.setup\_level(level\_data)

self.world\_shift = 0

self.current\_x = 0

# Player

self.player = pygame.sprite.GroupSingle()

player\_sprite = Player(100, 450)

self.player.add(player\_sprite)

# Health

self.health\_pickup = pygame.sprite.GroupSingle()

health\_pickup\_sprite = HealthPickup(375, 300)

self.health\_pickup.add(health\_pickup\_sprite)

# Enemies

self.enemies = pygame.sprite.Group()

self.enemy\_setup(level\_data)

# Collectibles

self.collectibles = pygame.sprite.Group()

self.collectible\_setup(level\_data)

def setup\_level(self, layout):

# Tiles

self.tiles = pygame.sprite.Group()

for row\_index, row in enumerate(layout):

for col\_index, cell in enumerate(row):

x = col\_index \* 64

y = row\_index \* 64

if cell == "X":

tile = Tile((x, y), 64, 64)

self.tiles.add(tile)

def enemy\_setup(self, layout):

for row\_index, row in enumerate(layout):

for col\_index, cell in enumerate(row):

x = col\_index \* 64

y = row\_index \* 64

if cell == "E":

enemy = Enemy(x, y)

self.enemies.add(enemy)

def collectible\_setup(self, layout):

for row\_index, row in enumerate(layout):

for col\_index, cell in enumerate(row):

x = col\_index \* 64

y = row\_index \* 64

if cell == "C":

collectible = Collectible(x, y)

self.collectibles.add(collectible)

def scroll\_x(self):

player = self.player.sprite

player\_x = player.rect.centerx

direction\_x = player.direction

if player\_x < 300 and direction\_x < 0:

self.world\_shift = 8

player.speed = 0

elif player\_x > 500 and direction\_x > 0:

self.world\_shift = -8

player.speed = 0

else:

self.world\_shift = 0

player.speed = 8

def horizontal\_collision(self):

player = self.player.sprite

player.rect.x += player.direction \* player.speed

for sprite in self.tiles.sprites():

if sprite.rect.colliderect(player.rect):

if player.direction < 0:

player.rect.left = sprite.rect.right

player.direction \*= -1

elif player.direction > 0:

player.rect.right = sprite.rect.left

player.direction \*= -1

for enemy in self.enemies:

if enemy.rect.colliderect(player.rect):

pygame.sprite.spritecollide(enemy, self.player, False)

if player.rect.left < 0:

player.rect.left = 0

if player.rect.right > SCREEN\_WIDTH:

player.rect.right = SCREEN\_WIDTH

def vertical\_collision(self):

player = self.player.sprite

player.apply\_gravity()

for sprite in self.tiles.sprites():

if sprite.rect.colliderect(player.rect):

if player.direction > 0:

player.rect.bottom = sprite.rect.top

player.direction = 0

elif player.direction < 0:

player.rect.top = sprite.rect.bottom

player.direction = 0

def run(self):

# Run the entire level

level\_complete = False

active\_sprites = pygame.sprite.Group()

active\_sprites.add(self.player)

while not level\_complete:

for event in pygame.event.get():

if event.type == pygame.QUIT:

level\_complete = True

# Update sprites

active\_sprites.update()

# Level scrolling

self.scroll\_x()

# Collision detection

self.horizontal\_collision()

self.vertical\_collision()

# Drawing

self.tiles.draw(self.display\_surface)

active\_sprites.draw(self.display\_surface)

# Update display

pygame.display.update()

# Reset level after completion

self.reset\_level()

def reset\_level(self):

# Reset level specific parameters

pass

# Rest of game code

# Create sprite groups

projectile\_group = pygame.sprite.Group()

enemy\_group = pygame.sprite.Group()

health\_pickup\_group = pygame.sprite.Group()

# Create level instances

level\_1 = Level(level\_1, screen)

level\_2 = Level(level\_2, screen)

level\_3 = Level(level\_3, screen)

# Game loop

run = True

while run:

# Event handling

for event in pygame.event.get():

if event.type == pygame.QUIT:

run = False

# Draw background

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

# Run level

if current\_level == 1:

level\_1.run()

elif current\_level == 2:

level\_2.run()

else:

level\_3.run()

# Update display

pygame.display.update()

pygame.quit()

Output

