**Pygame Dev Instructions**

1. You need to Import the Pygame Library and initialise the pygame library in the project.

#Pygame Game Dev

import pygame

pygame.init()

1. Now we can set the screen size and test that Pygame is working.

screen = pygame.display.set\_mode((800, 600))

1. We need to add Game Loop which is consistent across all games I have seen that use Pygame.

game\_running = True

while game\_running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

game\_running = False

pygame.quit()

1. We are now going to create a player object. The class player will have attributes of position, speed and size. We are also going to create an **instance** of the player. This is called **instantiation**.

class Player():

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

self.position = [0, 40]

self.speed = 20

self.size = [20,20]

#Instantiation of player\_one

player\_one = Player()

1. Adding a method to the player class

class Player():

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

self.position = [0, 40]

self.speed = 20

self.size = [20,20]

# Adding a method to the player class for movement

def move\_right(self):

self.position[0] += self.speed

def move\_left(self):

self.position[0] -= self.speed

player\_one = Player()

1. We now need to allow the player to have the ability to move around the screen. We will be using Pygame’s Event handling.

We need to add the following code to the Game Loop. This will check for a keypress left or right.

**while** game\_running:

# Event Handling

**for** event **in** pygame.event.get():

**if** event.type == pygame.KEYDOWN:

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

player\_one.move\_right()

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

player\_one.move\_left()

1. Drawing the player onto the screen.

We will need to place this code at the end of the Game Loop after the event

handling in line with the for loop.

# Draw player here

pygame.draw.rect(screen, (0,0,0), (player\_one.position[0], player\_one.position[1], player\_one.size[0], player\_one.size[1]))

# Swap buffers

pygame.display.flip()

**Note**: If we add a clear screen command directly before this piece of code we will clear the previous object from the screen otherwise there will be trailing objects.

# Clear the screen

screen.fill((255, 255, 255))

1. Test your program and make sure your player is moving as planned to the right and left. Add up and down to the player methods.

def move\_up(self):

self.position[1] -= self.speed

def move\_down(self):

self.position[1] += self.speed

1. Add event handling for Up and Down to the game loop.

if event.type == pygame.KEYDOWN:

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

player\_one.move\_right()

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

player\_one.move\_left()

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

player\_one.move\_up()

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

player\_one.move\_down()

1. Adding Obstacles – we need to create another class called Obstacle.

# Creating a player Obstacle

class Obstacle():

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

self.position = [100,100]

self.size = [20, 20]

After player\_one we need to add an instance of the obstacle by creating and instance variable. We can replace this later with more obstacles.

player\_one = Player()

obstacle = Obstacle()

1. Adding a random position for the obstacle on the screen.

To do this you will need to import the random function at the top of the program.

# Pygame Game Dev

import pygame

import random

In the Obstacle class you will need to use the random.randint function to randomly generate an object on the screen. Update the position of the obstacle to the code below.

#Create Obstacle class

Def class Obstacle():

self.position = [random.randint(0, 39)\*20, random.randint(0,28)\*20+20]

self.size = [20, 20]

1. Adding the Object to the screen – this goes after the player

# Draw obstacle

pygame.draw.rect(screen, (255,0,0), (obstacle.position[0], obstacle.position[1], obstacle.size[0], obstacle.size[1]))

text = font.render("Score: " + str(score), 1, (10, 10, 10))

screen.blit(text, (10,10))

1. Collision detection

We need to add a function to detect a collision between the player and the obstacle. Place this in the code after the obstacle class.

def check\_collision(player, obstacle):

if (player.position[0] == obstacle.position[0] and

player.position[1] == obstacle.position[1]):

return True

return False

1. Setting the Score

Place the score under the screen size setting.

screen = pygame.display.set\_mode((800, 600))

# Initialise Score

font = pygame.font.Font(None, 36)

score = 0

1. Adding to the score upon collision of the player and the obstacle.

if check\_collision(player\_one, obstacle):

score += 1

obstacle = Obstacle()

1. Adding to the score re-instantiating the obstacle.

# Incrementing the score

if check\_collision(player\_one, obstacle):

score += 1

obstacle = Obstacle()

1. Test the game and after the player collides with the object it will be re-initiated in a new position.