Raspberry Pi for Beginners

LESSON 9

MAKERHOUSE

EMPOWERING MAKERS

TABLE OF CONTENTS

GAMES PROGRAMMING

1 INTRODUCTION

Python has a fully supported library that purposedly made for more accessible games writing called PyGame. With PyGame, you could:

1. Draw a graphic
2. Animation control
3. Catch mouse and keyboard event and more.

In this lesson; games programming – we will go through the basic constructions of PyGame and start building your games.

Are you getting excited? Ok, let's go.

1.1 HELLO PYGAME

We will start our learning by constructing the hello world of Pygame -- better understanding the main frameworks – with Python. Figure 1 shows what we will be doing – controlling the Raspberry Pi logos based on the mouse event (movement; co-ordinate).

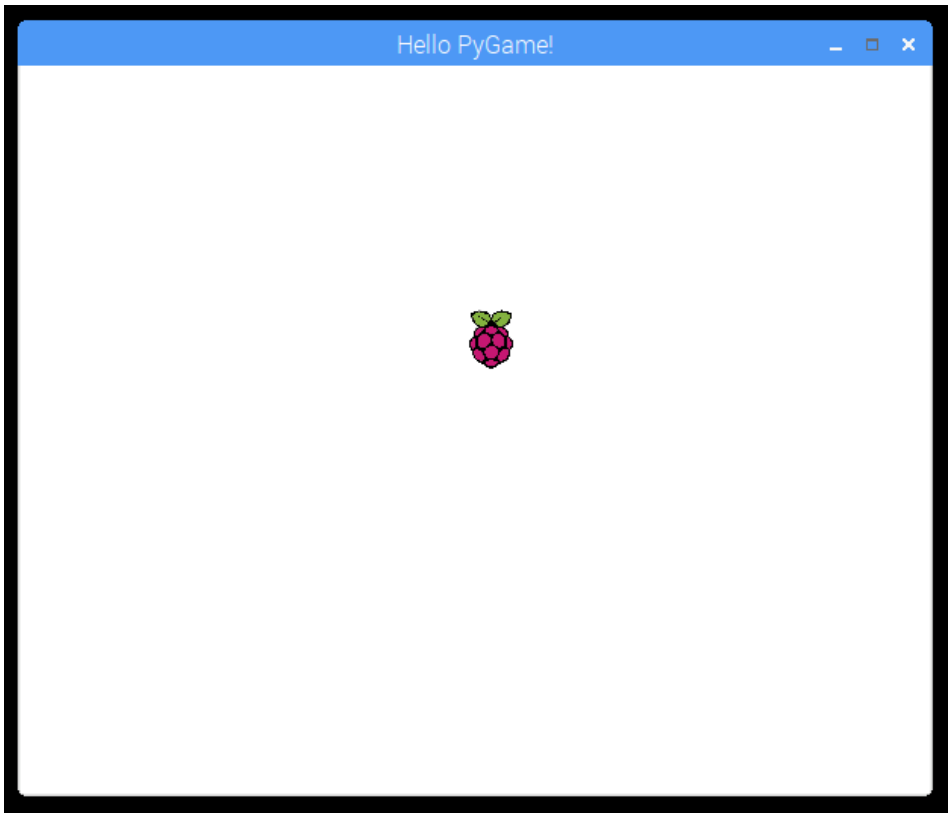

Figure 1: Hello PyGame!

1.2 THE CODE

Figure 2: Code used to construct the Hello PyGame

1.3 CODE EXPLANATION

1.3.1 SHEBANG

A shebang line defines where the interpreter is located. In this case, the python3 interpreter is located in **/usr/bin/python3**. Without the shebang line, the operating system does not know it's a python script, even if you set the execution flag on the script and run it like **./script.py**. To make the script run by default in python3, either invoke it as python3 script.py or set the shebang line.

```
#!/usr/bin/env python3
```

1.3.2 TITLE, AUTHOR AND DESCRIPTION

```
# Title: Hello PyGame
# Author: Autobotic
# Description: Hello World PyGame
```


1.3.3 IMPORTING THE PYGAME FRAMEWORK

```
import pygame
from pygame.locals import *
import os
import sys
```

1.3.4 INITIALIZING PYGAME

```
pygame.init()
```

1.3.5 SET THE CLOCK IN PYGAME

```
fpsClock = pygame.time.Clock()
```

1.3.6 CREATE WINDOWS SIZE, TITLE AND BACKGROUND

```
screen = pygame.display.set_mode((600, 480))
pygame.display.set_caption("Hello PYGame")
background = pygame.Color(255, 255, 255)
```

1.3.7 INSERT IMAGE

```
ball = pygame.image.load("img/raspberry.jpg")
ball = pygame.transform.scale(ball, (50, 50))
ball = ball.convert()
```

1.3.8 THE MAIN LOOP

```
while True:
    screen.fill(background)

    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    ball_x, ball_y = pygame.mouse.get_pos()

    screen.blit(ball, (ball_x, ball_y))

    pygame.display.update()
    fpsClock.tick()
```

2 CHALLENGE: ADDING LOGIC AND FUN IN HELLO PYGAME SCRIPT

This time we are going to re-make a hello PyGame scripts to make it more fun – catch the Raspberry Pi with basket and update the score. The player will control the basket using the mouse.

2.1 THE GAMES

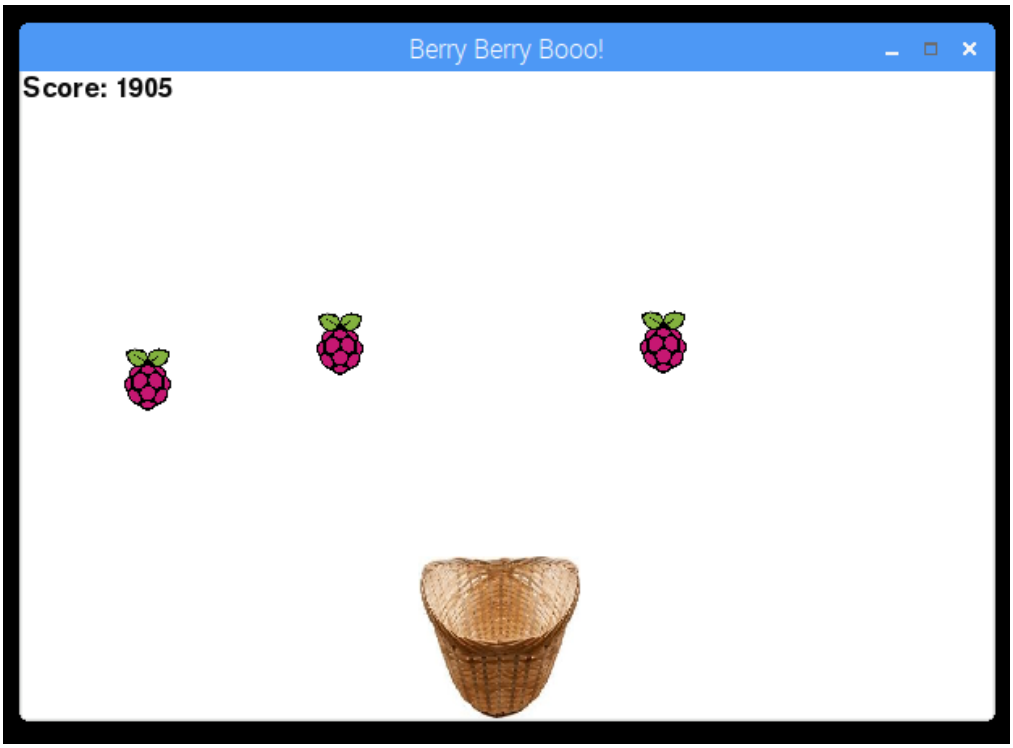

Figure 13: Cat and Dog Game Interface

2.1.1 THE FULL CODES

```
#!/usr/bin/env python3


# Title: Hello PyGame
# Author: Autobotic
# Description: Hello World PyGame


# importing pygame frameworks
import pygame
from pygame.locals import *
import os
import sys
```



```
import random

# initilaize score
score = 0

# initialize screen size
screen_width = 600
screen_height = 400

# initialize basket location
basket_x = screen_width // 2
basket_y = screen_height - 100

# update  basket position function
def update_basket():
        global basket_x
        global basket_y

        # read current mouse location (ignoring the y-axis
        basket_x, _ = pygame.mouse.get_pos()

        # limit the basket_x movement (not-overflow)
        if basket_x >= screen_width - 100:
                basket_x = screen_width - 100
        elif basket_x <= 0:
                basket_x = -20

        # update basket position
        screen.blit(basket, (basket_x, basket_y))

# Berry Class
class Berry:
        # Berry attributes
        x = 0
        y = 0
```



```
    dy = 0

    def __init__(self):
        self.x = random.randint(10, screen_width)
        self.y = 0
        self.dy = random.randint(1, 3)

    # update raspberry position function
    def update_berry(self):
        self.y += self.dy

        if self.y > basket_y:
            self.y = 0
            self.x = random.randint(10, screen_width)
        self.x += random.randint(-5, 5)
        if self.x < 10:
            self.x = 10
        if self.x > screen_width - 20:
            self.x = screen_width - 20
        screen.blit(berry, (self.x, self.y))

    # check is caught
    def is_caught(self):
        return self.y >= basket_y and self.x >= basket_x and self.x < basket_x + 50

# update score
def check_for_catch():
    global score

    for r in rasps:
        if r.is_caught():
            score += 1

# display score
def display(message):
```



```
        font = pygame.font.Font(None, 26)
        text = font.render(message, 1, (10, 10, 10))
        screen.blit(text, (0, 0))

# initializing pygame
pygame.init()

# frame per seconds (fps)
fpsClock = pygame.time.Clock()
rasps = [Berry(), Berry(), Berry()]

# create windows size, title, and background
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("Berry Berry Booo!")
background = pygame.Color(255, 255, 255)

# insert basket
basket = pygame.image.load("img/basket.jpeg")
basket = pygame.transform.scale(basket, (100, 100))
basket = basket.convert()

# insert berry
berry = pygame.image.load("img/raspberry.jpg")
berry = pygame.transform.scale(berry, (50, 50))
berry = berry.convert()

while True:
        # re-fresh the screen background
        screen.fill(background)

        for event in pygame.event.get():
                if event.type == QUIT:
                        pygame.quit()
                        sys.exit()
```



```
# update berry
#update_berry()
for r in rasps:
        r.update_berry()

# update basket
update_basket()

# check for catch
check_for_catch()
display("Score: " + str(score))

pygame.display.update()
fpsClock.tick()
```

Listing 1: Full code of Berry Catch with PyGame in Python

