Coding Games With Pygame Zero & Python

Student workbook (2nd edition)

Richard Smith

Chapter 8

Tutorial: Chase game

In this chapter we will build a complete chase game together, step by step. The Python we will use is very simple: just conditionals and loops.

The techniques here should be familiar to you because we used them in Program 4.4, 4.5, 5.1 and 5.2

Now we will show you how to put them all together in one program.

8.1 Moving Actor over a background

We must create two image files for our game. You can use a program such as Krita[1] to draw them and save them in the **`images`** folder (accessible with the **`images`** button in Mu). One is the player, called **`player.png`**. It should be small, about 64×64 pixels. Ideally it should have a transparent background.

The other is the background for the game itself. It can look like whatever you want, but it must be the same size as the game window, which will be 600×600 pixels.

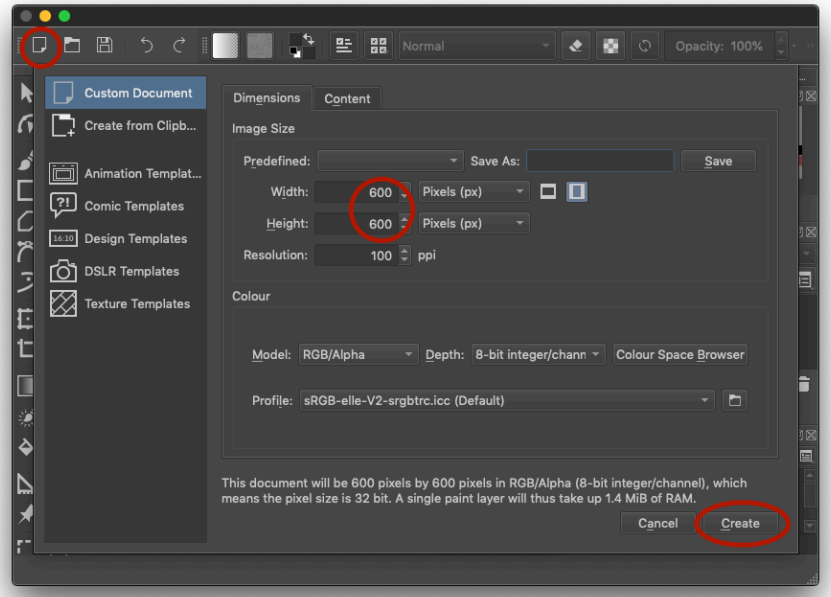

Figure 8.1: New image in Krita, 600×600 pixels

[1] https://krita.org

Program 8.1: Chase game

```
WIDTH = 600
HEIGHT = 600

background = Actor("background")
player = Actor("player")
player.x = 200
player.y = 200

def draw():
    screen.clear()
    background.draw()
    player.draw()

def update():
    if keyboard.right:
        player.x = player.x + 4
    if keyboard.left:
        player.x = player.x - 4
    if keyboard.down:
        player.y = player.y + 4
    if keyboard.up:
        player.y = player.y - 4
```

Exercise 8.1.

Run the program and move the Actor around with the keys.

8.2 Screen wrap-around

One problem you will soon find with the program is that you can move off the edge of the screen and get lost. One way to solve this would be to stop movement at the screen edges. Instead we going to make the player teleport to the opposite edge when he leaves the screen. Add this code to the end of the program, and make sure it is indented so that it becomes part of the `update()` function.

```
    if player.x > WIDTH:
        player.x = 0
    if player.x < 0:
        player.x = WIDTH
    if player.y < 0:
        player.y = HEIGHT
    if player.y > HEIGHT:
        player.y = 0
```

Exercise 8.2.

Change the code so that the player stops at the edges rather than wraps-around.

8.3 Enemy chases the player

Let's add an enemy to chase the player. At the top of the program, create a variable to store the enemy Actor:

```
enemy = Actor("alien")
```

At the end of the **`draw()`** function (but still indented so part of the function), draw the enemy. Remember there is only ever one single **`draw()`** function. No program has two. All drawing goes inside this function.

Here is the complete function including the new line at the end:

```
def draw():
    screen.clear()
    background.draw()
    player.draw()
    enemy.draw()
```

At the end of the **`update()`** function (but still indented so part of the function), add these lines to move the enemy:

```
    if enemy.x < player.x:
        enemy.x = enemy.x + 1
    if enemy.x > player.x:
        enemy.x = enemy.x - 1
    if enemy.y < player.y:
        enemy.y = enemy.y + 1
    if enemy.y > player.y:
        enemy.y = enemy.y - 1
    if player.colliderect(enemy):
        exit()
```

Exercise 8.3.

Run the program verify the enemy chases the player.

Exercise 8.4.

Make the enemy faster so the game is more difficult.

Exercise 8.5.

Create an image file **`enemy.png`** and save it in the **`images`** folder. Change the code so it loads **`"enemy"`** instead of **`"alien"`**.

8.4 Collecting items

Create a small image file **`coin.png`** and save it in the **`images`** folder. It should look like a coin or something else you would like to collect. We will also need a variable to store the *score*, i.e. number of coins collected.

```
coin = Actor("coin", pos=(300,300))
score = 0
```

At the end of the **`draw()`** function (but still indented so part of the function), draw the coin. Remember there is only ever one single **`draw()`** function. No program has two. All drawing goes inside this function.

Here is the complete function including the new line at the end:

```
def draw():
    screen.clear()
    background.draw()
    player.draw()
    enemy.draw()
    coin.draw()
```

At the end of the **`update()`** function (but still indented so part of the function), add these lines to move the coin when it is collected:

```
    if coin.colliderect(player):
        coin.x = random.randint(0, WIDTH)
        coin.y = random.randint(0, HEIGHT)
        score = score + 1
        print("Score:", score)
```

Exercise 8.6.

Run the program and collect a coin. What happens?

```
NameError: name 'random' is not defined
```

This happens because we are using a function **`randint()`** to get a random number. This function is not build-in to Python; it is part of the *random* library. So at the top of the program, add the first line:

```
import random
```

Exercise 8.7.

Run the program again and collect a coin. Does it work now?

No!

```
UnboundLocalError: local variable 'score' referenced before assignment
```

You will get an error because `score` is a global variable and we are trying to modify it inside the `update()` function. Therefore at the **top** of the `update()` function, add this line to declare to Python our intention to modify a global variable:

```
global score
```

It must be the **first** line in the function and it must be **indented**. The lines surrounding it should look like this:

```
def update():
    global score
    if keyboard.right:
```

Exercise 8.8.

Run the program again and verify it works!

8.5 Player 2

We can make any game into a two player game. At the top of the program, create a variable to store the Actor for the second player:

```
player2 = Actor("alien")
```

At the end of the **`draw()`** function (but still indented so part of the function), draw the enemy. Here is the complete function with the new line at the end:

```
def draw():
    screen.clear()
    background.draw()
    player.draw()
    enemy.draw()
    coin.draw()
    player2.draw()
```

At the end of the **`update()`** function (but still indented so part of the function), add these lines to move the second player:

```
    if keyboard.d:
        player2.x = player2.x + 4
    if keyboard.a:
        player2.x = player2.x - 4
    if keyboard.s:
        player2.y = player2.y + 4
    if keyboard.w:
        player2.y = player2.y - 4
    if player.colliderect(player2):
        exit()
```

Exercise 8.9. Advanced

Create a variable **`score2`** and store the score for player two, i.e. it goes up when he collides with a coin.

8.6 Showing the score on the screen

At the end of the **`draw()`** function (but still indented so part of the function), draw a title on the screen:

```
    screen.draw.text("My game", (200,0), color='red')
```

The **`draw.text()`** function is not like **`print()`** - it can only print strings of text, not numbers. Therefore we must convert the score into a string. Add these lines to the end of the **`draw()`** function:

```
    score_string = str(score)
    screen.draw.text(score_string, (0,0), color='green')
```

Exercise 8.10.

Change the colour of the text.

Exercise 8.11. Advanced

Display the word "Score: " before the score.

Exercise 8.12. Advanced

When the **`score`** reaches 10, show a message on the screen to congratulate the player

8.7 Timer

Add a variable at the top of the program (but preferably after any **`import`** statements) to store the number of seconds of time remaining in the game:

```
time = 20
```

Pygame Zero calls our **`update()`** function many times per second. We can ask it to tell us how much time has passed by adding a *parameter* to the function, **`delta`**. We then subtract this from the remaining time. Modify **`update()`** so the first lines look like this:

```
def update(delta):
    global score, time
    time = time - delta
    if time <= 0:
        exit()
```

We can also display the time on the screen. At the end of the **`draw()`** function (but still indented so part of the function) add these lines:

```
    time_string = str(time)
    screen.draw.text(time_string, (50,0), color='green')
```

Exercise 8.13.

Run the program. Could the displayed time be improved?

We don't need to see the decimal places in the time. Modify those lines to use the **`round()`** function, like this:

```
    time_string = str(round(time))
    screen.draw.text(time_string, (50,0), color='green')
```

8.8 Finished game

Here is the finished game with all the changes included:

Program 8.2: Finished chase game

```
import random

WIDTH = 600
HEIGHT = 600

background = Actor("background")
player = Actor("player")
player.x = 200
player.y = 200

enemy = Actor("alien")
player2 = Actor("player")
coin = Actor("alien", pos=(300,300))
score = 0
time = 20


def draw():
    screen.clear()
    background.draw()
    player.draw()
    enemy.draw()
    player2.draw()
    coin.draw()
    screen.draw.text("My game", (200,0), color='red')
    score_string = str(score)
    screen.draw.text(score_string, (0,0), color='green')
    time_string = str(round(time))
    screen.draw.text(time_string, (50,0), color='green')

def update(delta):
    global score, time
    time = time - delta
    if time <= 0:
        exit()
    if keyboard.right:
        player.x = player.x + 4
```

```
    if keyboard.left:
        player.x = player.x - 4
    if keyboard.down:
        player.y = player.y + 4
    if keyboard.up:
        player.y = player.y - 4

    if player.x > WIDTH:
        player.x = 0
    if player.x < 0:
        player.x = WIDTH
    if player.y < 0:
        player.y = HEIGHT
    if player.y > HEIGHT:
        player.y = 0

    if enemy.x < player.x:
        enemy.x = enemy.x + 1
    if enemy.x > player.x:
        enemy.x = enemy.x - 1
    if enemy.y < player.y:
        enemy.y = enemy.y + 1
    if enemy.y > player.y:
        enemy.y = enemy.y - 1
    if player.colliderect(enemy):
        exit()

    if keyboard.d:
        player2.x = player2.x + 4
    if keyboard.a:
        player2.x = player2.x - 4
    if keyboard.s:
        player2.y = player2.y + 4
    if keyboard.w:
        player2.y = player2.y - 4
    if player.colliderect(player2):
        exit()

    if coin.colliderect(player):
        coin.x = random.randint(0, WIDTH)
        coin.y = random.randint(0, HEIGHT)
        score = score + 1
        print("Score:", score)
```

However that is not the end! There are many things you could add to this game.

- Add more enemies.
- Give the player three lives.