21

Fruit Ninja Games — Code

① Basic fruit ninja game

Our starting version of the game consists of an apple that appears at random positions on the screen and the computer prints out "Good shot!" and draws a new apple when you click on the apple, otherwise prints "You missed!" and ends the game.

fruit_ninja_basic

```
from random import randint

apple = Actor("apple")


def draw():
    screen.clear()
    apple.draw()


def place_apple():
    apple.x = randint(10, 800)
    apple.y = randint(10, 600)


def on_mouse_down(pos):
    if apple.collidepoint(pos):
        print("Good shot!")
        place_apple()
    else:
        print("You missed!")
        quit()


place_apple()
```

Things to check and to talk to coders about:

- Programs saved in the folder `coderdojo_tramore/fruilt_ninja_games`.
- Difference between defining a function and calling a function.
- Outline of life-cycle of a pgzero program.

CoderDojo, Tramore, Waterford. (kmurphy@wit.ie)

② Code review

The *Coding Games with Python* is very good, but it does do a few things that we can improve on. So before we move on we are going to look at our code and see if we can improve it.

fruit_ninja_refactored

```
from random import randint

WIDTH, HEIGHT = 600, 600

fruit = Actor("apple")


def draw():
    screen.clear()
    fruit.draw()


def place_fruit():
    fruit.x = randint(20, WIDTH - 20)
    fruit.y = randint(20, HEIGHT - 20)


def on_mouse_down(pos):
    if fruit.collidepoint(pos):
        print("Good shot!")
        place_fruit()
    else:
        print("You missed!")
        quit()


place_fruit()
```

Things to check and to talk to coders about:

- Coders create a new file (and keep old for reference later) using naming convention given.
- Why do we refactor code?
- What happens now if we change the screen height? width? what happens in out previous program? concept of "Magic Numbers".
- Why do we care about names of our identifiers?

③ Adding sounds

One of the reasons we switched over from turtle graphics to **Pygame Zero** is to simplify generating sounds — both short terms sound effects and long term background music.

fruit_ninja_sounds

```
1  from random import randint
2
3  WIDTH, HEIGHT = 600, 600
4
5  fruit = Actor("apple")
6
7
8  def draw():
9      screen.clear()
10     fruit.draw()
11
12
13 def place_fruit():
14     fruit.x = randint(20, WIDTH - 20)
15     fruit.y = randint(20, HEIGHT - 20)
16
17
18 def on_mouse_down(pos):
19     if fruit.collidepoint(pos):
20         print("Good shot!")
21         if randint(1,10)==1:
22             sounds.shoot_and_cheer.play()
23         else:
24             sounds.shoot.play()
25         place_fruit()
26     else:
27         print("You missed!")
28         quit()
29
30 music.set_volume(0.5)
31 music.play("the_french_lovers_song")
32 place_fruit()
```

Things to check and to talk to coders about:

- The worksheet covers the steps used to find, download and convert the sound effects and the background music — this is so coders can pick their own games assets later. In the short term, all of the required game assets are in the correct folders already or can be found in the `resources` folders.
- Line 21 is an example of common programming task — we want something to occur randomly at a specified rate. Here we want the cheer to happens every 10 times on average. What happens when the 10 parameter is lowered down to 2, or even 1? What happens if it is increased?
- Some coders found it difficult to know where to put the code — indicating lack of understanding of the program structure. Not sure if this would occur anyway or implies a rewrite of the worksheet.

④ Keeping score

Currently the game just prints out a different message whenever you miss or hit a fruit. It would be nicer to keep score so that you can show off your shooting skills. To do this we will create a new variable called `score`. Also we will add a heads up display (HUD) to show the score and game title.

fruit_ninja_score

```
from random import randint

WIDTH, HEIGHT = 600, 600

fruit = Actor("apple")
fruit.score = 0
fruit.lives = 3


def draw():
    screen.clear()
    fruit.draw()
    screen.draw.text("Score %s" % fruit.score,
        topleft=(0, 0))
    screen.draw.text("Fruit Ninja",
        midtop=(WIDTH // 2, 0), color="orange", fontsize=60)
    screen.draw.text("Lives %s" % fruit.lives,
        topright=(WIDTH, 0))


def place_fruit():
    fruit.x = randint(20, WIDTH - 20)
    fruit.y = randint(20, HEIGHT - 20)


def on_mouse_down(pos):
    if fruit.collidepoint(pos):
        fruit.score = fruit.score + 1
        sounds.shoot.play()
        place_fruit()
    else:
        fruit.lives = fruit.lives - 1
        if fruit.lives == 0:
            quit()
        else:
            place_fruit()


place_fruit()
```

Things to check and to talk to coders about:

- Big concept here is the placement of text boxes within the screen, in particular the use of different anchor points.
- Stored `score` in object `fruit` so don't have to worry about **`global`**. The concept of scope and **`global`** keyword will be introduced in next game.

⑤ Moving targets

Hitting a stationary object is easy ... what happens if they are moving?

So we are going to change our fixed position fruit to fruit that is thrown upwards from below the screen and then fall back down due to gravity — now we will need some physics!

fruit_ninja_final

```
1  from random import randint
2
3  from pygame.math import Vector2 as Vector
4
5  WIDTH, HEIGHT = 600, 600
6
7  fruit = Actor("apple")
8  fruit.score = 0
9  fruit.lives = 3
10
11
12 def draw():
13     screen.clear()
14     fruit.draw()
15     screen.draw.text("Score %s" % fruit.score, topleft=(0, 0))
16     screen.draw.text("Fruit Ninja", midtop=(WIDTH // 2, 0), color="ora
17     screen.draw.text("Lives %s" % fruit.lives, topright=(WIDTH, 0))
18
19
20 def place_fruit():
21
22     # random position (just below the screen)
23     x = randint(20, WIDTH - 20)
24     y = HEIGHT + 20
25     fruit.pos = Vector(x, y)
26
27     # random velocity - fruit thrown towards the centre
28     dy = -randint(20, 25)
29     dx = randint(0, 5)
30     if fruit.x > WIDTH / 2:
31         dx = -dx
32     fruit.velocity = Vector(dx, dy)
33
34     # acceleration due to gravity
35     fruit.acceleration = Vector(0, 0.5)
```

Things to check and to talk to coders about:

- This is quite a complex concept — so new coders could skip it. On the other hand, it is a nice effect and the general pattern will be used many many times — so try and see how far can they go.
- Line 30 ensures that the fruit will be thrown towards the centre of the screen — so user has a chance of hitting it.
- We often use “d” as a prefix if we are talking about a small change.

fruit_ninja_final

```
def update():
    # apply physics
    fruit.velocity = fruit.velocity + fruit.acceleration
    fruit.pos = fruit.pos + fruit.velocity

    # fell below screen so get new fruit
    if fruit.y > HEIGHT + 30:
        place_fruit()


def on_mouse_down(pos):
    if fruit.collidepoint(pos):
        print("Good shot!")
        fruit.score = fruit.score + 1
        if randint(1, 10) == 1:
            sounds.shoot_and_cheer.play()
        else:
            sounds.shoot.play()
        place_fruit()
    else:
        print("You missed!")
        fruit.lives = fruit.lives - 1
        if fruit.lives == 0:
            quit()
        else:
            place_fruit()


music.set_volume(0.25)
music.play("the_french_lovers_song")
place_fruit()
```

Things to check and to talk to coders about:

- Using `Vector` allows us to group pairs of operations (the horizontal and the vertical) together. Think of a `Vector` as an ordered pair of information — like (firstname,surname) — order is important.
- Note that acceleration never changes, only velocity is updated due to acceleration, and position is updated due to velocity.
- To get physics working in an simulation (from this simple one all the way up to ForteNite) we:
  - Setup object with initial position, velocity and acceleration (do this once).
  - As fast as possible we repeatedly call and `update` to move everything slightly and a `draw` to redraw the screen.

⑥ Varying targets

We don't just want apples! Lets add other fruit also, after all this is 'fruit ninja' not 'apple ninja'.

fruit_ninja_final

```
from random import randint, choice

from pygame.math import Vector2 as Vector

WIDTH, HEIGHT = 600, 600

fruits = ['apple', 'banana', 'cherries', 'grapes', 'lemon', 'pear', 'pi
nonfruits = ['burger', 'fries', 'pizza', 'soda']

fruit = Actor("apple")
fruit.score = 0
fruit.lives = 3


def draw():
    screen.clear()
    screen.fill((255,255,255))
    fruit.draw()
    screen.draw.text("Score %s" % fruit.score, topleft=(0, 0))
    screen.draw.text("Fruit Ninja", midtop=(WIDTH // 2, 0), color="oran
    screen.draw.text("Lives %s" % fruit.lives, topright=(WIDTH, 0))


def place_fruit():

    # random fruit or non-fruit
    if randint(1, 10) == 1:
        fruit.image = choice(nonfruits)
    else:
        fruit.image = choice(fruits)

    # random position below the screen
    x = randint(20, WIDTH - 20)
    y = HEIGHT - 20
    fruit.pos = Vector(x, y)

    # random velocity - fruit thrown towards the centre
    dy = -randint(20, 25)
    dx = randint(0, 10)
    if fruit.x > WIDTH / 2:
        dx = -dx
    fruit.velocity = Vector(dx, dy)

    # acceleration due to gravity
    fruit.acceleration = Vector(0, 0.5)
```

```
46
47
48  def update():
49      fruit.velocity += fruit.acceleration
50      fruit.pos += fruit.velocity
51      if fruit.y > HEIGHT + 30:
52          place_fruit()
53
54
55  def on_mouse_down(pos):
56      if fruit.collidepoint(pos):
57          if fruit.image in fruits:
58              print("Good shot!")
59              fruit.score = fruit.score + 1
60          else:
61              print("Eat healthy :-)")
62              fruit.lives = fruit.lives - 1
63              if randint(1, 10) == 1:
64                  sounds.shoot_and_cheer.play()
65              else:
66                  sounds.shoot.play()
67          place_fruit()
68      else:
69          print("You missed!")
70          fruit.lives = fruit.lives - 1
71          if fruit.lives == 0:
72              quit()
73          else:
74              place_fruit()
75
76
77  music.set_volume(0.25)
78  music.play("the_french_lovers_song")
79  place_fruit()
```

Things to check and to talk to coders about:

- This implementation has both varying fruit and also non-fruits. Better to get coders to do just the varying fruits first.
- See `resources` folder for game assets.