Code Club Challenge: Space Invaders

Setting up

One of the very first computer games was called "Space Invaders"[1] and we are going to try to recreate it in Python.

To get started, open LXTerminal and use the following command to download the start of the project:

Listing 1: Get the the starting point
```
$ git clone http://github.com/jrmhaig/space_invaders
```

This should create a new directory called space_invaders containing a Python program and some icons. Open the program with:

Listing 2: Opening the starting point
```
$ cd space_invaders
$ idle space_invaders.py
```

> **Note**
>
> We need to use idle (for Python version 2) instead of idle3 (for Python version 3) because the PyGame module only works for Python version 2 on the Raspberry Pi at the moment.
> You could also launch Idle from the desktop icon and then browse to the correct directory to open the file.

Controlling the ship

The first thing to do is to add the ship that you (the player) will control. This will move left and right along the bottom of the screen while the aliens, which you have to shoot, will go at the top. Look at the script below and see what you need to change.

> **Note**
>
> Can you identify the parts that:
>
> - Finds the picture file to use for the ship?
>
> - Tells the computer where to put the ship?
>
> - Finds out if you are pressing the left or right buttons?

---

[1]It was apparently originally created in 1978, which makes it almost as old as me!

Listing 3: Controlling the ship

```python
import pygame


# Colours
WHITE = (255,255,255)


# Size of the screen
WIDTH = 400
HEIGHT = 300


pygame.init()
clock = pygame.time.Clock()
game_speed = 85
screen = pygame.display.set_mode((WIDTH, HEIGHT), 0)

# Images                                              # New
ship_image = pygame.image.load('icons/ship.png')      # New

# Position of the ship                                # New
ship = {                                              # New
        'x': 150,                                     # New
        'y': 260,                                     # New
        }                                             # New


run = True


while run:
    screen.fill(WHITE)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        elif event.type == pygame.KEYDOWN:             # New
            # A key has been pressed                   # New
            if event.key == pygame.K_LEFT:             # New
                # Move the ship left                   # New
                ship['x'] = ship['x'] - 1              # New
            elif event.key == pygame.K_RIGHT:          # New
                # Move the ship right                  # New
                ship['x'] = ship['x'] + 1              # New

    # Put the ship on the screen                       # New
    screen.blit(ship_image, (ship['x'], ship['y']))    # New
```

```
     # Refresh the screen
45   pygame.display.update()
     clock.tick(game_speed)
```

Make the movement easier

You have probably noticed that the movement of the ship is not very easy. You really want to be able to hold the key pressed rather than having to press it lots of times. The reason it is acting the way it is is because when it detects a keyboard *event* with `event.type == pygame.KEYDOWN` it sees the single action when you press the key and moves the ship once. What we really want is for it to tell when the key is pressed and then keep moving the ship until it is released, with `event.type == pygame.KEYUP`.

> **Note**
>
> I have not printed all the code, indicating missing bits with '...', to save space.

Listing 4: Easier control of the ship

```
...

# Position of the ship
ship = {
5        'x': 150,
         'y': 260,
         }
ship_move_left = False                                    # New
ship_move_right = False                                   # New
10

run = True

while run:
    screen.fill(WHITE)
15
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        elif event.type == pygame.KEYDOWN:
20          # A key has been pressed
            if event.key == pygame.K_LEFT:
                # Start moving the ship left         # Changed
                ship_move_left = True                # Changed
```

```
              elif event.key == pygame.K_RIGHT:
25                  # Start moving the ship right        # Changed
                    ship_move_right = True               # Changed
            elif event.type == pygame.KEYUP:             # New
                # A key has been released                # New
                if event.key == pygame.K_LEFT:           # New
30                  # Stop moving the ship left           # New
                    ship_move_left = False               # New
                elif event.key == pygame.K_RIGHT:        # New
                    # Stop moving the ship right          # New
                    ship_move_right = False              # New

35
        if ship_move_left:                               # New
            ship['x'] = ship['x'] - 1                    # New
        if ship_move_right:                              # New
            ship['x'] = ship['x'] + 1                    # New

40
        # Put the ship on the screen
        screen.blit(ship_image, (ship['x'], ship['y']))


        # Refresh the screen
45      pygame.display.update()
        clock.tick(game_speed)
```

Add a bit of *class*

Now we have managed to get the ship moving more easily but do you see what happens when you go to the edge of the window? We will solve this later. First, we are going to introduce a new programming idea; *Classes* and *Objects.*

So far, we have just a single thing in our game – a ship that you can control with the arrow keys. Later we will also have a number of aliens as well as bullets, and all of these have very similar information about them and actions. For example, they all have a position (x and y) and they all need to move. With a Class we can write code for them all once and only once.

> **Note**
>
> Note, some of the lines marked "Changed" have very small changes, even just a single character, while others replace several lines with just one.
> Can you see how I stopped the ship going off the side of the window?
> Can you work out how to make it move faster or slower?

Listing 5: Adding classes

```
import pygame

# Colours
WHITE = (255,255,255)
5
# Size of the screen
WIDTH = 400
HEIGHT = 300

10 pygame.init()
clock = pygame.time.Clock()
game_speed=85
screen = pygame.display.set_mode((WIDTH, HEIGHT), 0)

15 class GamePiece:                                      # New
    def __init__(self, x, y, image):                 # New
        self.x = x                                   # New
        self.y = y                                   # New
        self.speed = 1                               # New
20      self.image = image                           # New
        self.move_left = False                       # New
        self.move_right = False                      # New
        self.min_x = 10                              # New
        self.max_x = 340                             # New
25
    def move(self):                                  # New
        if self.move_left:                           # New
            if self.x > self.min_x:                  # New
                self.x = self.x - self.speed         # New
30      if self.move_right:                          # New
            if  self.x < self.max_x:                 # New
                self.x = self.x + self.speed         # New

    def draw(self):                                  # New
35      screen.blit(self.image, (self.x, self.y))    # New

# Images
ship_image = pygame.image.load('icons/ship.png')

40 # Game objects                                     # Changed
ship = GamePiece(150, 260, ship_image)               # Changed

run = True
```

```python
45  while run:
        screen.fill(WHITE)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
50              run = False
            elif event.type == pygame.KEYDOWN:
                # A key has been pressed
                if event.key == pygame.K_LEFT:
                    # Start moving the ship left
55                  ship.move_left = True                    # Changed
                elif event.key == pygame.K_RIGHT:
                    # Start moving the ship right
                    ship.move_right = True                   # Changed
            elif event.type == pygame.KEYUP:
60              # A key has been released
                if event.key == pygame.K_LEFT:
                    # Stop moving the ship left
                    ship.move_left = False                   # Changed
                elif event.key == pygame.K_RIGHT:
65                  # Stop moving the ship right
                    ship.move_right = False                  # Changed

        # Move and draw the ship                             # Changed
        ship.move()                                          # Changed
70      ship.draw()                                          # Changed

        # Refresh the screen
        pygame.display.update()
        clock.tick(game_speed)
```

Explanation

In the previous section we added a *Class* called `GamePiece`. By itself, this does not do anything to it provides a pattern from which we created an *Object* for the ship. Later, we will add another object for an alien and our code will contain:

Listing 6: Object creation example

```python
ship = GamePiece(150, 260, ship_image)
alien = GamePiece(150, 30, alien_image)
```

Each of these is separate from the other and has its own x and y coordinates. They also both know how to move and draw themselves in the correct position in the window if we call the functions:

Listing 7: Function call example

```
ship.move()
alien.move()

ship.draw()
5  alien.draw()
```

and we can even set their speeds with:

Listing 8: Setting object variables

```
ship.speed = 0.6
alien.speed = 0.3
```

> **Note**
>
> This may be the answer to one of the questions in the last section!

As well as the `move` and `draw` functions you will see another function called `__init__` (that is a double '_' before and after the word 'init'. This is a special function that is always run once when the object is created with `ship = GamePiece(150, 260, ship_image)`. This makes sure that all the variables have the correct values at the start.

Add an alien

The real Space Invaders game has lots of aliens in rows but for the moment we will start with just one. As I said in the last section, a lot of the work is already done for us as we can use the `GamePiece` class.

First, make some changes to the `GamePiece` class to let the alien "bounce" when it reaches the side of the screen.

Listing 9: Changes to the class

```
...

class GamePiece:
    def __init__(self, x, y, image):
5       self.x = x
```

```
            self.y = y
            self.speed = 1
            self.image = image
            self.move_left = False
10          self.move_right = False
            self.min_x = 10
            self.max_x = 340
            self.bounce = False                         # New

15      def move(self):
            if self.move_left:
                if self.x > self.min_x:
                    self.x = self.x - self.speed
                elif self.bounce:                       # New
20                  self.y = self.y + 5                 # New
                    self.move_left = False              # New
                    self.move_right = True              # New
            if self.move_right:
                if self.x < self.max_x:
25                  self.x = self.x + self.speed
                elif self.bounce:                       # New
                    self.y = self.y + 5                 # New
                    self.move_left = True               # New
                    self.move_right = False             # New
30
        def draw(self):
            screen.blit(self.image, (self.x, self.y))


    ...
```

Next, this is all that is needed to create the alien.

Listing 10: Create the alien object

```
    ...

    # Images
    ship_image = pygame.image.load('icons/ship.png')
5   alien_image = pygame.image.load('icons/alien1.png')     # New

    ship = GamePiece(150, 260, ship_image)
    alien = GamePiece(150, 30, alien_image)                 # New

10  # Make the alien move by itself                         # New
```

```
     alien.move_left = True                                    # New
     # Make the alien bounce when it hits the edge             # New
     alien.bounce = True                                       # New
     alien.speed = 0.3                                         # New
15
     ...
```

Listing 11: Use the alien object

```
...

        # Move and draw the ship
        ship.move()
5       ship.draw()

        # Move and draw the alien          # New
        alien.move()                       # New
        alien.draw()                       # New
10
...
```

Shoot the alien

The next thing to do is to let you try to shoot the alien. For this, we need to load an image of a bullet with this line (try to find the correct place to put it):

Listing 12: Load the bullet image

```
bullet_image = pygame.image.load('icons/bullet.png')
```

Now the bullet needs to move up the screen, rather than left or right, and then disappear when it reaches the top. For this we need to add some new variables to the __init__ function:

Listing 13: New variables

```
     def __init__(self, x, y, image):
         self.x = x
         self.y = y
         self.speed = 1
5        self.image = image
         self.move_left = False
         self.move_right = False
         self.move_up = False                  # New
```

```
        self.min_y = 20                              # New
10      self.min_x = 10
        self.max_x = 340
        self.bounce = False
```

and add a new part to the move function:

Listing 14: New move function

```
    def move(self):
        if self.move_left:
            if self.x > self.min_x:
                self.x = self.x - self.speed
5           elif self.bounce:
                self.y = self.y + 5
                self.move_left = False
                self.move_right = True
        if self.move_right:
10          if self.x < self.max_x:
                self.x = self.x + self.speed
            elif self.bounce:
                self.y = self.y + 5
                self.move_left = True
15              self.move_right = False
        if self.move_up:                          # New
            if self.y > self.min_y:               # New
                self.y = self.y - self.speed      # New
```

At the start of the game, the bullet doesn't exist and we can indicate this with:

Listing 15: Initialise bullet

```
# Game objects
bullet = None
ship = GamePiece(150, 260, ship_image)
alien = GamePiece(150, 30, alien_image)
```

And finally, the bullet should appear when we press the 'SPACE' bar.

Listing 16: Fire!

```
    for event in pygame.event.get():
        ...
        elif event.type == pygame.KEYDOWN:
```

```
                    # A key has been pressed
5                   ...                              # New (next 4 lines)
                elif event.key == pygame.K_SPACE and bullet == None:
                        # Fire the bullet from the mid-point of the ship
                        bullet = GamePiece(ship.x+22, ship.y, bullet_image)
                        bullet.move_up = True
10      ...


        # Move and draw the alien
        alien.move()
        alien.draw()
15
        # Move and draw the bullet                         # New
        if bullet != None:                                 # New
            bullet.move()                                  # New
            if bullet.y > bullet.min_y:                    # New
20              bullet.draw()                              # New
            else:                                          # New
                bullet = None                              # New
```

Hit the alien

You may have noticed that the bullet currently just goes straight through the alien. This isn't very good! In the GamePiece class we need a new function to detect if it has been hit.

Listing 17: Detect a hit

```
class GamePiece:
    def __init__(self, x, y, image):
        self.x = x
        self.y = y
5       self.width = 0                                # New
        self.height = 0                               # New
...

    def draw(self):
10      self.blit(self.image, (self.x, self.y))

    def detect_hit(self, other):                      # New
        if other.x > self.x \                         # New
                and other.x < self.x + self.width \   # New
15              and other.y > self.y \                # New
```

```
                    and other.y < self.y + self.height:  # New
                return True                              # New
            else:                                        # New
                return False                             # New
20  ...
    # Make the alien move by itself
    alien.move_left = True
    # Make the alien bounce when it hits the edge
    alien.bounce = True
25  alien.speed = 0.3
    # Set the height and width of the alien            # New
    alien.width = 47                                    # New
    alien.height = 22                                   # New
```

At the moment there is only one alien so when it is hit we will put it back at the top and make it move faster. Later, we can have several rows of aliens and try to keep a score.

Listing 18: Hit the alien

```
        # Move and draw the bullet
        if bullet != None:
            bullet.move()
            if bullet.y > bullet.min_y:
5               bullet.draw()
                if alien.detect_hit(bullet):           # New
                    # Move the alien back to the top    # New
                    alien.x = 10                        # New
                    alien.y = 10                        # New
10                  # Move it a bit faster              # New
                    alien.speed = alien.speed + 0.1     # New
                    bullet = None                       # New
            else:
                bullet = None
```

More aliens

Now we will make a line of 5 aliens and store them in an array.

Listing 19: Line of 5 aliens

```
...

bullet = None
```

```
    ship = GamePiece(150, 260, ship_image)
5   aliens = []                                     # Changed
    for i in range(5):                              # New
        alien = GamePiece(30 + i*50, 30, alien_image)  # New
        # Keep aliens in line when they bounce      # New
        alien.min_x = 20 + i*50                     # New
10      alien.max_x = 140 + i*50                    # New
        # Make the alien move by itself             # Changed
        alien.move_left = True                      # Changed
        # Make the alien bounce when it hits the edge  # Changed
        alien.bounce = True                         # Changed
15      alien.speed = 0.3                           # Changed
        # Set the height and width of the alien     # Changed
        alien.width = 47                            # Changed
        alien.height = 22                           # Changed
        aliens.append(alien)                        # New

20
    run = True


    ...
```

And then get them all to move and get hit by bullets.

Listing 20: Line of 5 aliens

```
    ...

        # Move and draw the ship
        ship.move()
5       ship.draw()

        # Move and draw the aliens                      # Changed
        for alien in aliens:                            # Changed
            alien.move()                                # Changed
10          alien.draw()                                # Changed

        # Move and draw the bullet
        if bullet != None:
            bullet.move()
15          if bullet.y > bullet.min_y:
                bullet.draw()
                for alien in aliens:                    # Changed
                    if alien.detect_hit(bullet):        # Changed
                        aliens.remove(alien)            # Changed
```

```
20                        bullet = None                          # Changed
                          break                                  # Changed
              else:
                  bullet = None


25      # Refresh the screen
        pygame.display.update()
        clock.tick(game_speed)
```

and lose if aliens reach the bottom.

Score

It is about time we kept the score. For the moment, we will give 10 points for each hit.
Make variable for the score and prepare the type font.

Listing 21: Score variable

```
import pygame

# Colours
WHITE = (255,255,255)
5  BLUE = (0, 0, 255)                              # New


   ...


   run = True
10 score = 0                                       # New
   font = pygame.font.SysFont("monospace", 12)   # New

   while run:
       screen.fill(WHITE)
15
   ...
```

Add 10 points for each alien hit and display the score.

Listing 22: Score a hit

```
        # Move and draw the bullet
        if bullet != None:
            bullet.move()
            if bullet.y > bullet.min_y:
```

```
 5              bullet.draw()
                for alien in aliens:
                    if alien.detect_hit(bullet):
                        aliens.remove(alien)
                        bullet = None
10                      score = score + 10                          # New
                        break
            else:
                bullet = None

15      score_text = font.render("Score: " + str(score), 1, BLUE) # New
        screen.blit(score_label, (10, 10))                        # New
```

To do

Keep score, alien bullets, more levels.