python

DESARROLLO DE VIDEOJUEGOS

Curso de pygame

Descripción breve

Este tutorial está realizado a través del curso publicado en YouTube del cana Mundo Python

Pere Manel Verdugo Zamora
pereverdkugo@gmial.com

Contenido

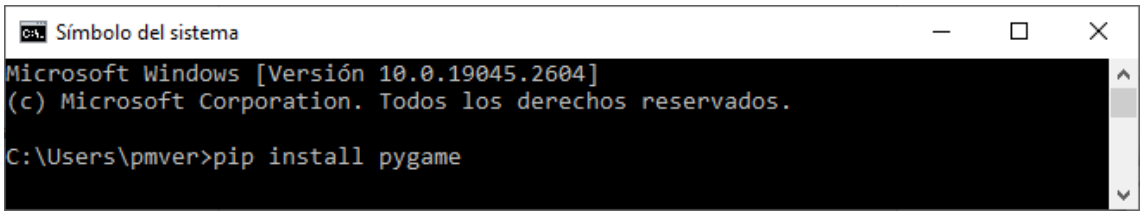
1.- Creando una ventana y estructura de nuestro juego

Pygame básicamente es un módulo que nos permite crear videojuegos utilizando el lenguaje de programación Python si hacemos una búsqueda en Google.

Podemos encontrar una pequeña descripción de lo que es pygame:

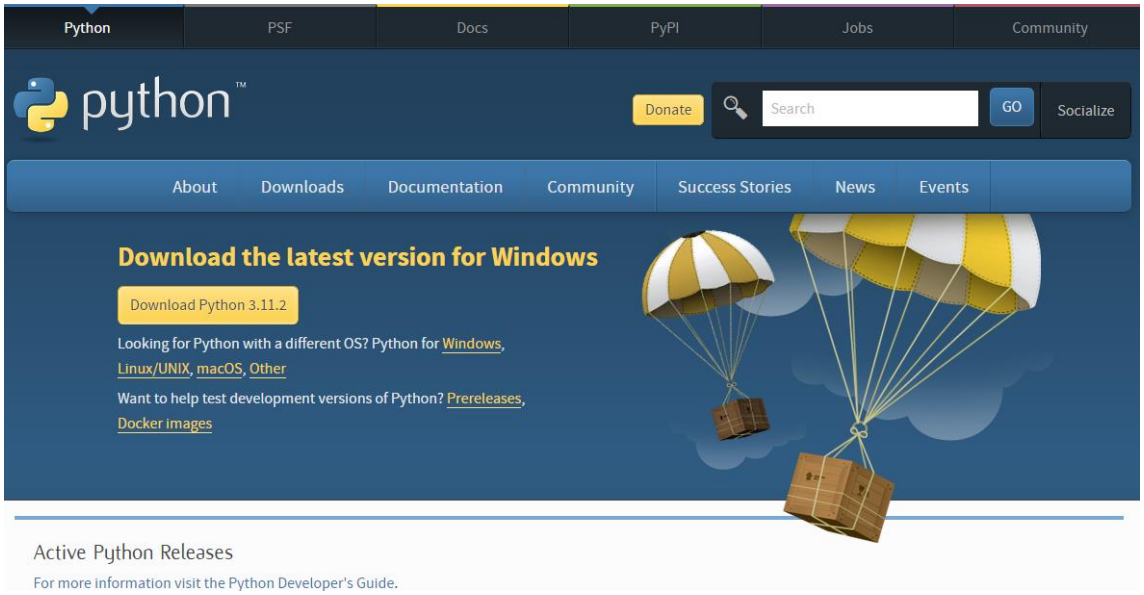
Y además el primer enlace que encontramos nos llevará a su página oficial.

Para instalarlo desde Cmd de Windows escribiremos: pip install pygame.

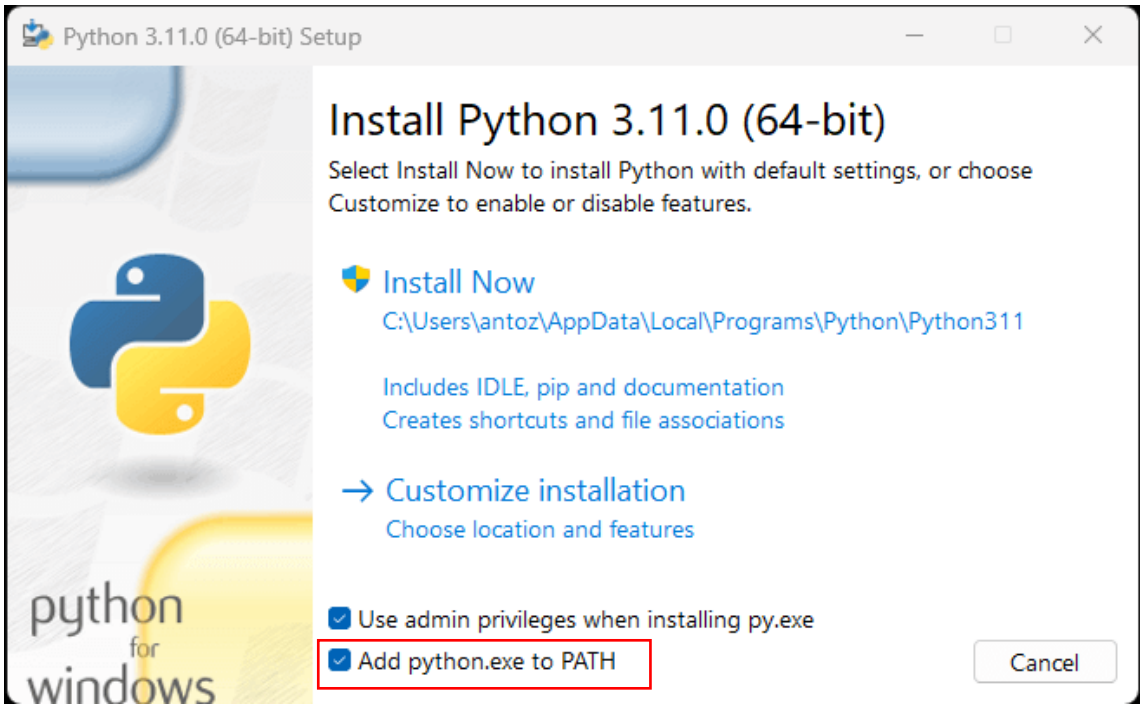


Pero antes de instalar pygame, lo primero que vamos a necesitar es tener instalado Python y si es posible que sea la última versión, para ello iremos a la siguiente dirección:

https://www.python.org/downloads/



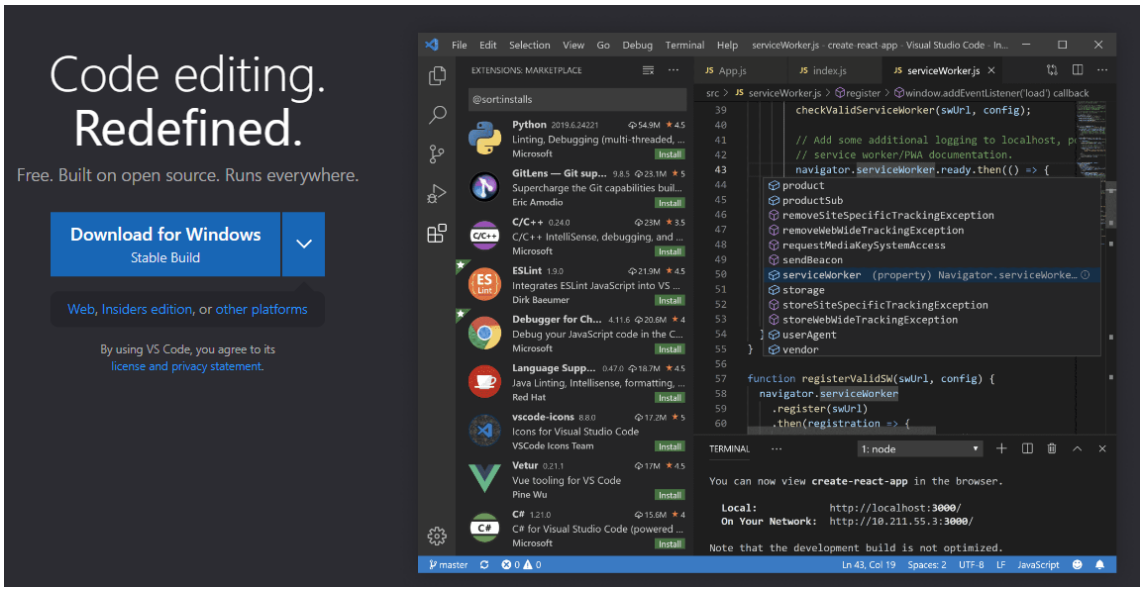
Una vez lo hayas descargado, la instalación es fácil, lo único que has de tener en cuenta que durante el proceso de instalación actives una determinada opción:



Esto lo que hará es que tu ordenador pueda ejecutar el programa de Python desde cualquier carpeta que estés de otro modo solo podrás ejecutarlos desde la carpeta donde está el programa.

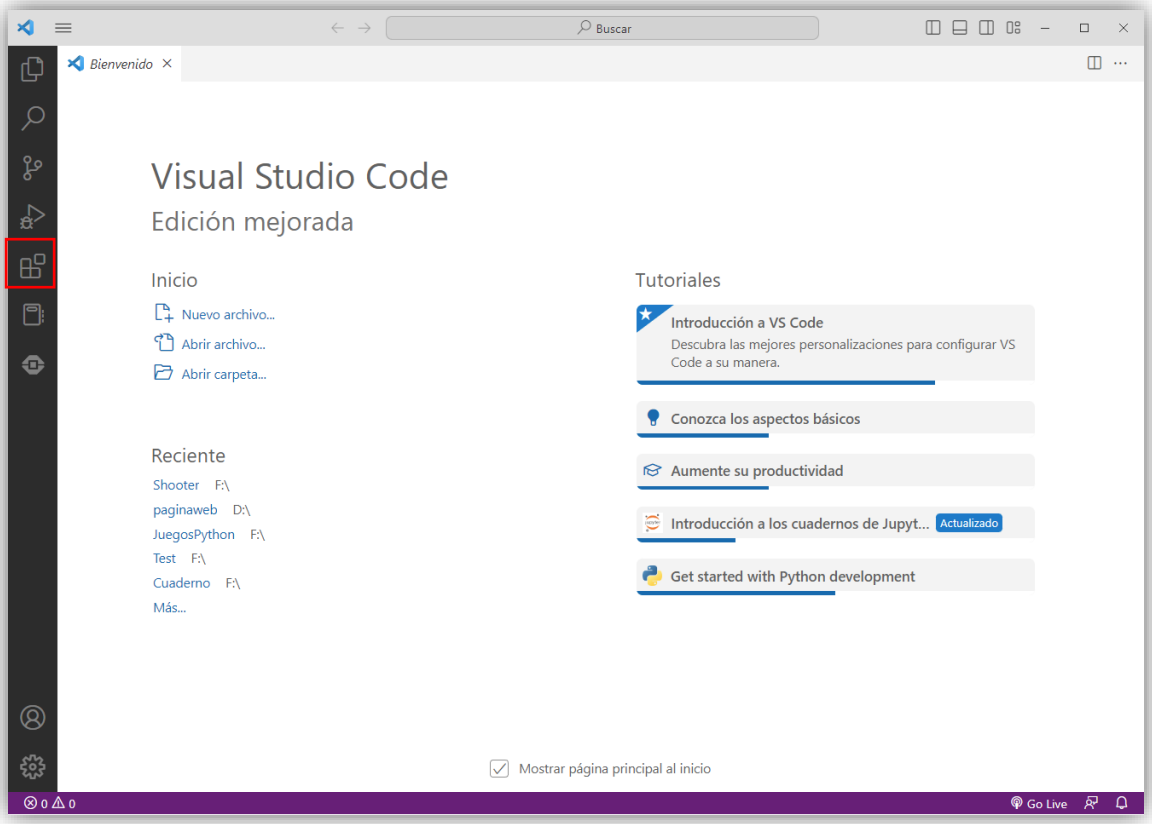
Yo como editor utilizo Visual Studio Code, lo vas a poder descargar desde la siguiente dirección:

https://code.visualstudio.com/

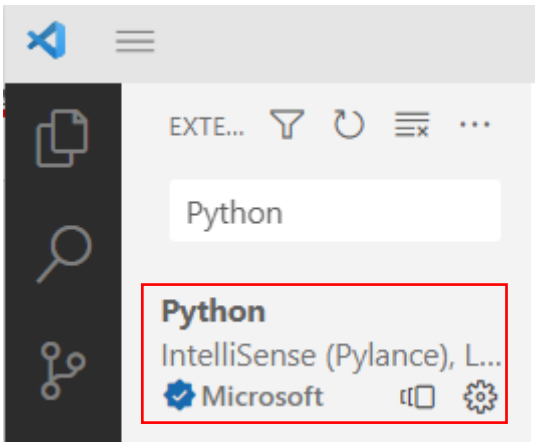


Lo descargas y lo instalas.

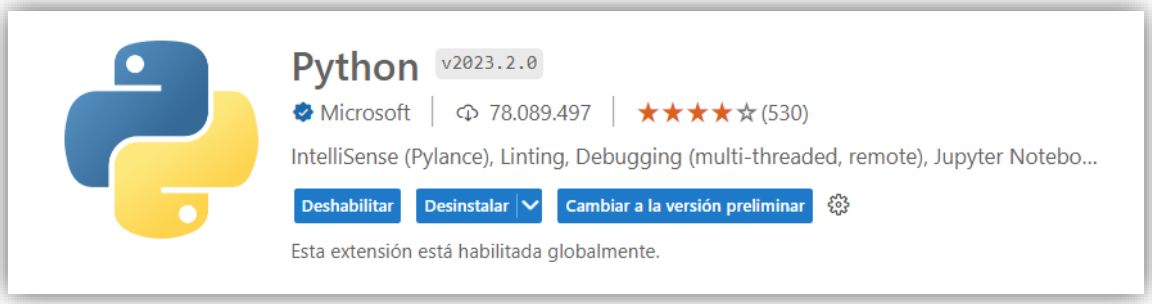
Una vez instalado y ejecutado:



Nos vamos al apartado de extensiones.



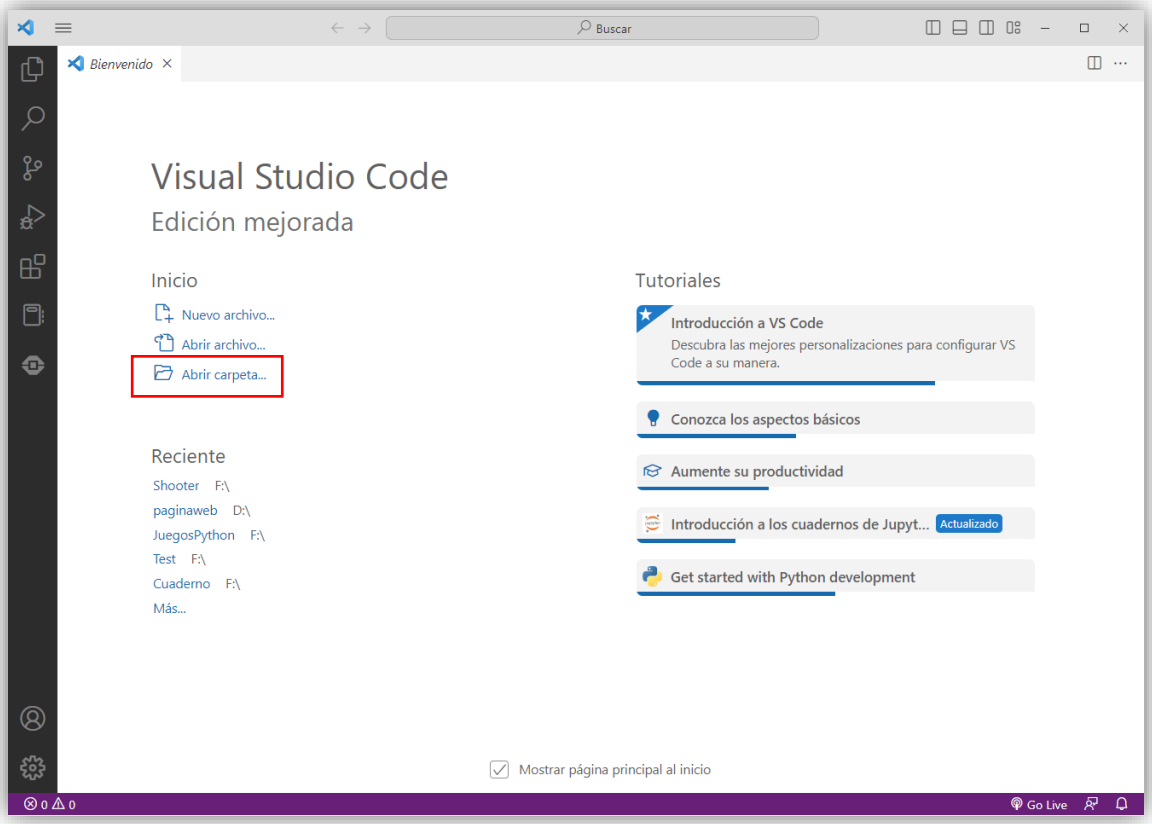
He instalas esta extensión:



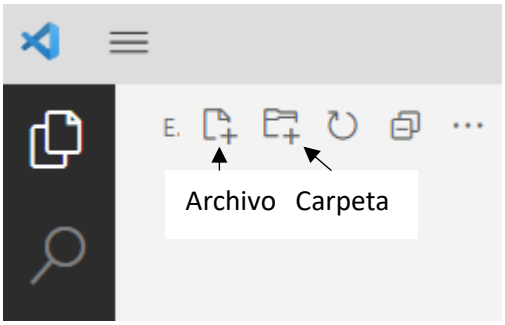
No ayudará a la hora de programar, completando palabras y avisándote de posible errores que puedas hacer.

Para empezar vamos a crear una carpeta donde vamos a realizar todos los proyectos de este tutorial, a la carpeta la vamos a llamar CursoPygame.

Ahora vamos a ejecutar Visual Studio.

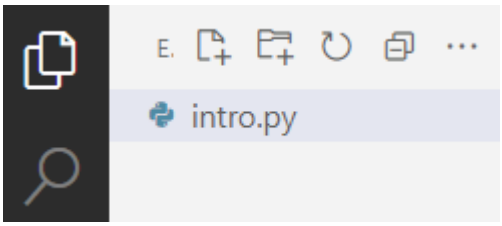


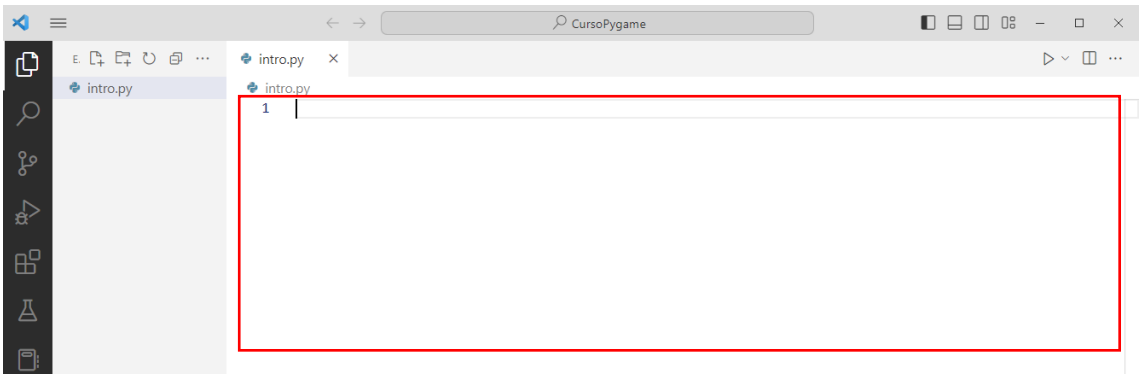
Seleccionaremos Abrir carpeta…, para seleccionar la carpeta que creamos con anterioridad.



En la parte superior izquierda vamos a encontrar unos botones para crear archivos, carpetas, etc.

Vamos a crear nuestro primer proyecto a la que le vamos a llamar intro.py, recuerda que no debes olvidarte de la extensión.





En la zona seleccionada es donde vamos a escribir nuestros programas.

```
1   # Importamos la librería
2   import pygame
3
4   # Vamos a inicializar la librería
5   pygame.init()
6
7   # Para definir las dimensiones de la ventana,
8   # creo una variable asignandole el valor en Tupla
9   size = (800, 500)
10
11  # Vamos a crear la ventana
12  screen = pygame.display.set_mode(size)
13
14  # Vamos a crear un bucle infinito
15  while True:
16      pass
```

Si ejecutamos la aplicación se mostrará una ventana, pero como aun no la hemos terminado esta se puede trabar, vamos a seguir hasta poderla ejecutar.

```
1   # Importamos la librería
2   import pygame, sys
3
4   # Vamos a inicializar la librería
5   pygame.init()
6
```

Importamos la librería sys.

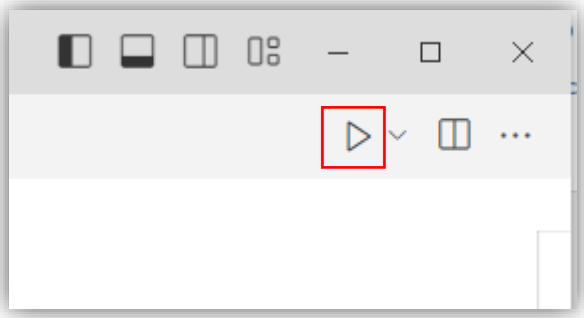
```
7   # Para definir las dimensiones de la ventana,
8   # creo una variable asignandole el valor en Tupla
9   size = (800, 500)
10
11  # Vamos a crear la ventana
12  screen = pygame.display.set_mode(size)
13
14  # Vamos a crear un bucle infinito
15  while True:
16      for event in pygame.event.get():
17          if event.type== pygame.QUIT:
18              sys.exit()
```

Recoge todos los eventos que le podemos hacer en la ventana.

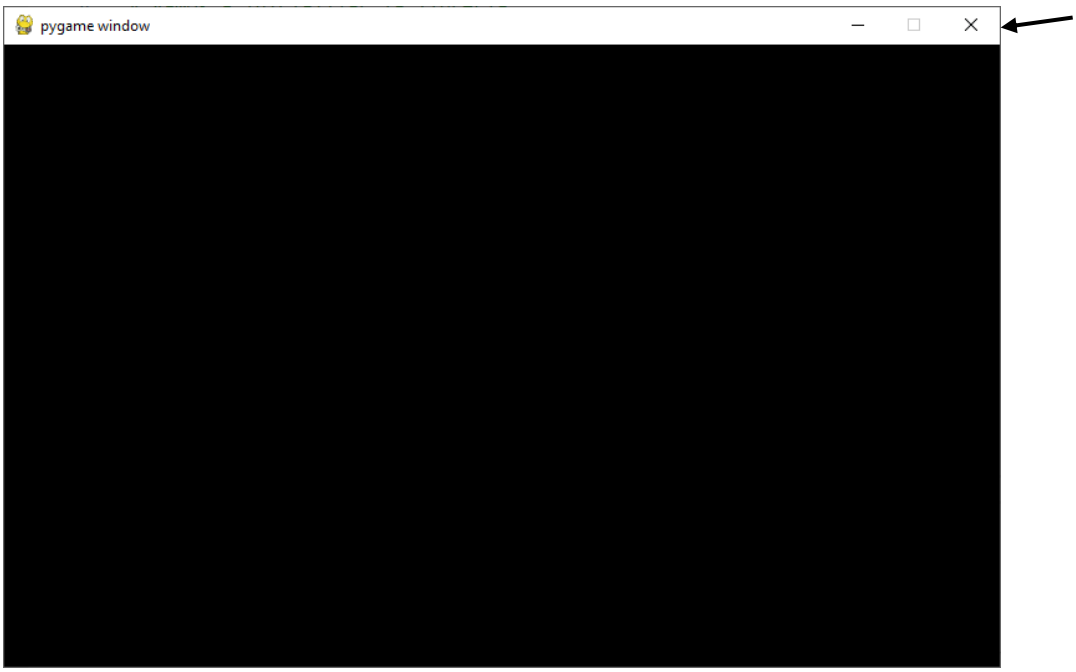
Si seleccionamos de la ventana la X de cerrar ventana.

Cerramos la ventan.

Para ejecutar el programa:



Este será el resultado:



Para cerrar la ventana le damos a la x.

Básicamente esta va a ser nuestra estructura para programar videojuego, lo que te aconsejo que tengas una plantilla que puedas reutilizar y así evitar de tener que copiar este código cada vez.

```
1   import pygame, sys
2
3   pygame.init()
4
5   size = (800, 500)
6
7   screen = pygame.display.set_mode(size)
8
9   while True:
10      for event in pygame.event.get():
11          if event.type== pygame.QUIT:
12              sys.exit()
```

Cuando empieces con otro proyecto podrás copiar este código.

Como curiosidad si agregamos:

```
1   import pygame, sys
2
3   pygame.init()
4
5   size = (800, 500)
6
7   screen = pygame.display.set_mode(size)
8
9   while True:
10      for event in pygame.event.get():
11          print(event)
12          if event.type== pygame.QUIT:
13              sys.exit()
```

Y ejecutamos de nuevo.

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
<Event(32768-ActiveEvent {'gain': 0, 'state': 1})>
<Event(32784-WindowLeave {'window': None})>
<Event(32768-ActiveEvent {'gain': 1, 'state': 2})>
<Event(32785-WindowFocusGained {'window': None})>
<Event(2304-ClipboardUpdate {})>
<Event(32770-VideoExpose {})>
<Event(32776-WindowExposed {'window': None})>
```

En la consola de muestran todos los eventos que hacemos a la ventana.

2.- Dibujando figuras en la pantalla

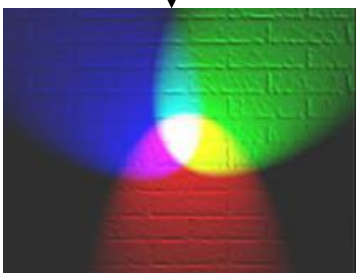
En este capítulo vamos a ver como dibujar figuras geométricas en nuestra ventana.

Vamos a crear un nuevo archivo llamado dibujando_lineas.py y copiaremos el código del archivo intro.py.

```
1   import pygame, sys
2   pygame.init()
3
4   # Definir colores
5   BLACK = (0, 0, 0)
6   WHITE = (255, 255, 255)
7   GREEN = (0, 255, 0)
8   RED = (255, 0, 0)
9   BLUE = (0, 0, 255)
10
11  size = (800, 500)
12
13  screen = pygame.display.set_mode(size)
14
15  while True:
16      for event in pygame.event.get():
17          if event.type== pygame.QUIT:
18              sys.exit()
19      # Pinta la ventana de color blanco
20      screen.fill(WHITE)
21      # Actualiza la ventana
22      pygame.display.flip()
```

Los colores de define en RGB:

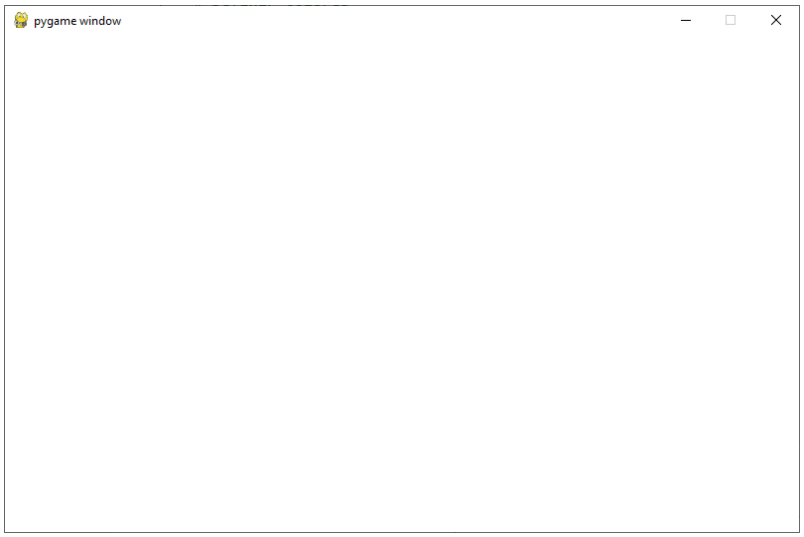
Cuando hablamos de RGB en español rojo, verde y azul, estamos haciendo referencia a un sistema de composición de colores basado en la adición de los colores primarios de la luz.



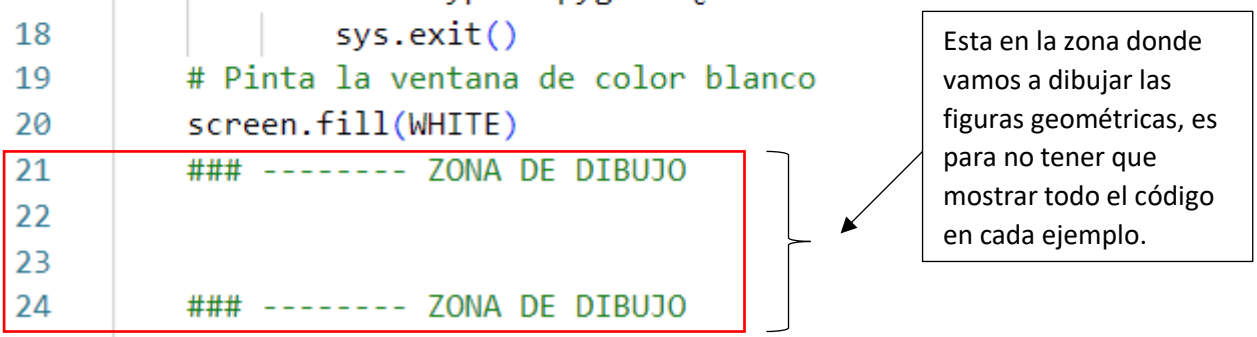
Con la sentencia fill (llenar) pintamos la ventana de color blanco.

Para que se muestre el resultado siempre hay que actualizar.

Este será el resultado:



```python
import pygame, sys
pygame.init()

# Definir colores
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
GREEN = (0, 255, 0)
RED = (255, 0, 0)
BLUE = (0, 0, 255)

size = (800, 500)

screen = pygame.display.set_mode(size)

while True:
    for event in pygame.event.get():
        if event.type== pygame.QUIT:
            sys.exit()
    # Pinta la ventana de color blanco
    screen.fill(WHITE)
    ### -------- ZONA DE DIBUJO


    ### -------- ZONA DE DIBUJO
    # Actualiza la ventana
    pygame.display.flip()
```



Vamos a dibujar una línea:

pygame.draw.line(screen, GREEN, [0, 100], [100, 100], 5)

Después de escribir pygame.draw.line que es para dibujar la línea tenemos que agregar unos parámetros para decir cómo queremos la línea.

screen → En que ventana queremos dibujar la línea, en el código anterior en la línea 13 estamos creando una ventana y como nombre es screen.

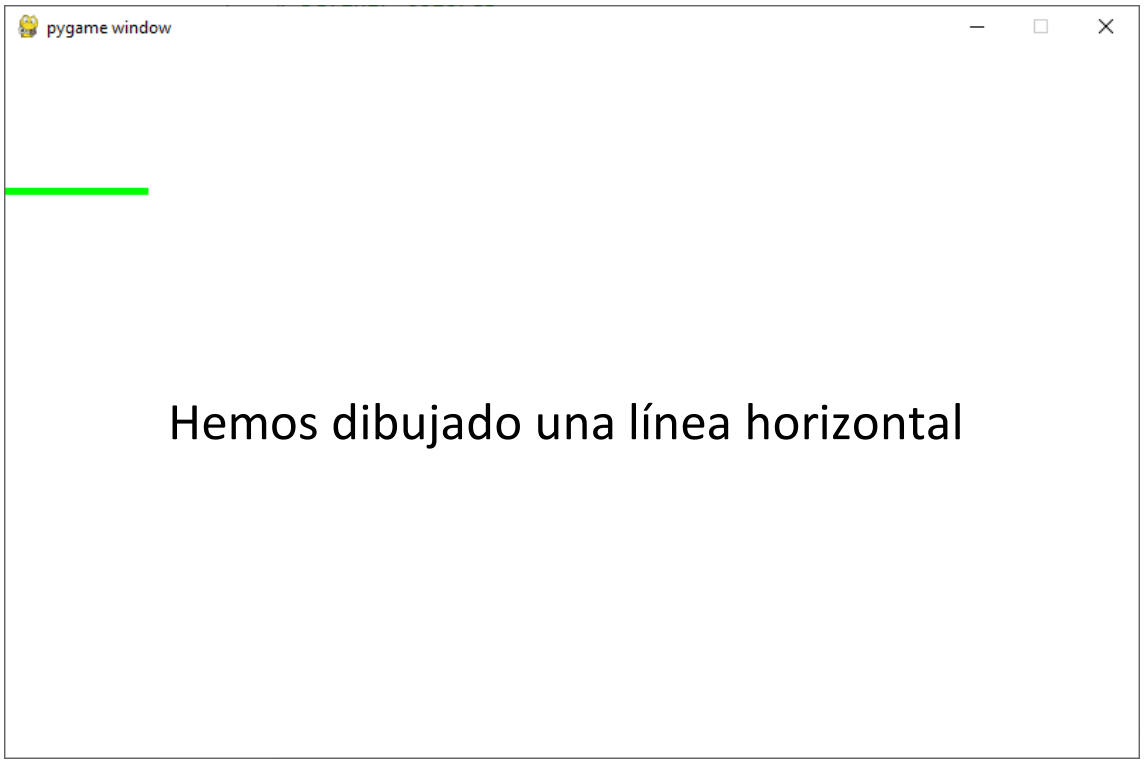
[0, 100] → Punto coordenadas (x, y) de donde empieza la línea.

[100, 100] → Punto coordenadas (x, y) de donde termina la línea.

5 → El grosor de la línea.

```
21      ### -------- ZONA DE DIBUJO
22      pygame.draw.line(screen, GREEN, [0, 100], [100, 100], 5)
23
24      ### -------- ZONA DE DIBUJO
```

Si ejecutamos este será el resultado:



Podemos ir jugando con los parámetros para ir modificando la línea.

Ahora vamos a dibujar un rectángulo:

paygame.draw.rect(screen, BLACK, (100, 100, 80, 80))

Para dibujar el rectángulo pondremos paygame.draw.rect y agregaremos los siguientes parámetros.

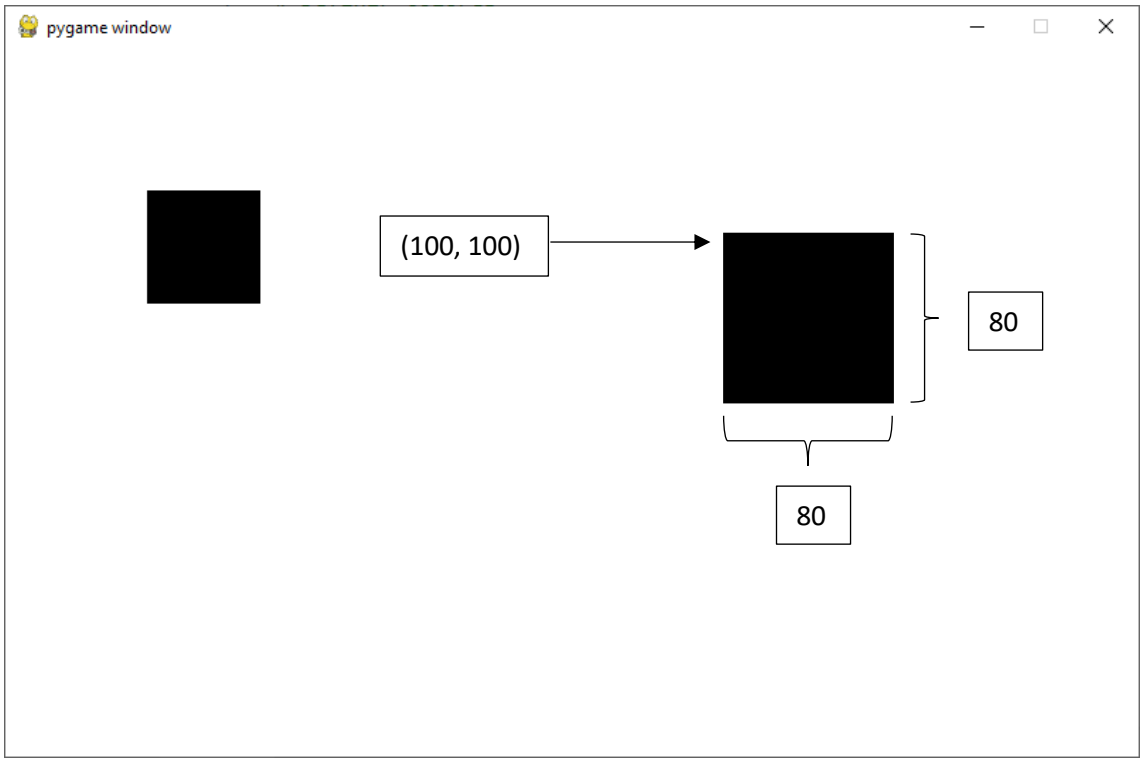
screen → En que ventana lo dibujamos.

BLACK → Color de relleno del rectángulo.

(100, 100, 80, 80) → La esquina superior izquierda irá en la coordenada (x, y) de 100, 100, los 80, 80 es el ancho y el alto del rectángulo con la coordenada inicial.

```
21      ### -------- ZONA DE DIBUJO
22      pygame.draw.rect(screen, BLACK, (100,100, 80, 80))
23
24      ### -------- ZONA DE DIBUJO
```

Este será el resultado:



Para dibujar un círculo:

pygame.draw.circle(screen, BLACK, (200, 200), 30)

Con pygame.draw.circle dibujamos una circunferencia con los siguientes parámetros:

screen → El nombre de la ventana donde queremos dibujar la circunferencia.

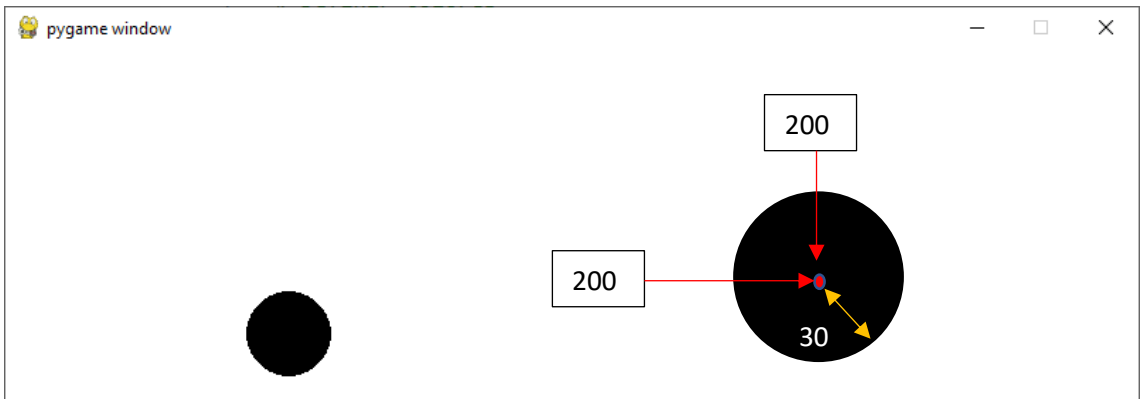
BLACK → Color de la circunferencia.

(200, 200) → Una tupla con las coordenadas, (x, y).

30 → El radio de la circunferencia.

```
21      ### -------- ZONA DE DIBUJO
22      pygame.draw.circle(screen, BLACK, (200, 200), 30)
23
24      ### -------- ZONA DE DIBUJO
```

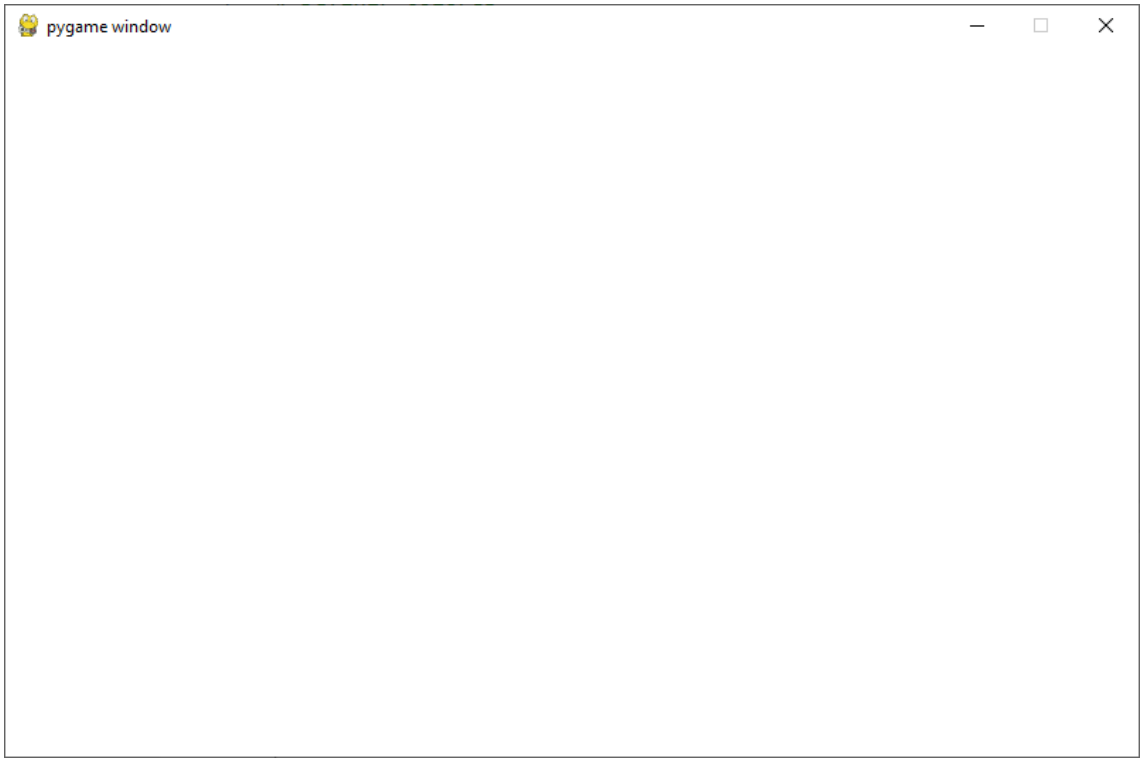
Este será el resultado:



Una cosa importante las figuras se tienen que poner después de pintar la pantalla, si las dibujamos antes cuando pintemos la pantalla estas figuras no se verán.

```
20      screen.fill(WHITE)
21      ### -------- ZONA DE DIBUJO
22      pygame.draw.circle(screen, BLACK, (200, 200), 30)
23
24      ### -------- ZONA DE DIBUJO
25      # Pinta la ventana de color blanco
26      screen.fill(WHITE)
```

Este será el resultado:



3.- Figuras con for loops

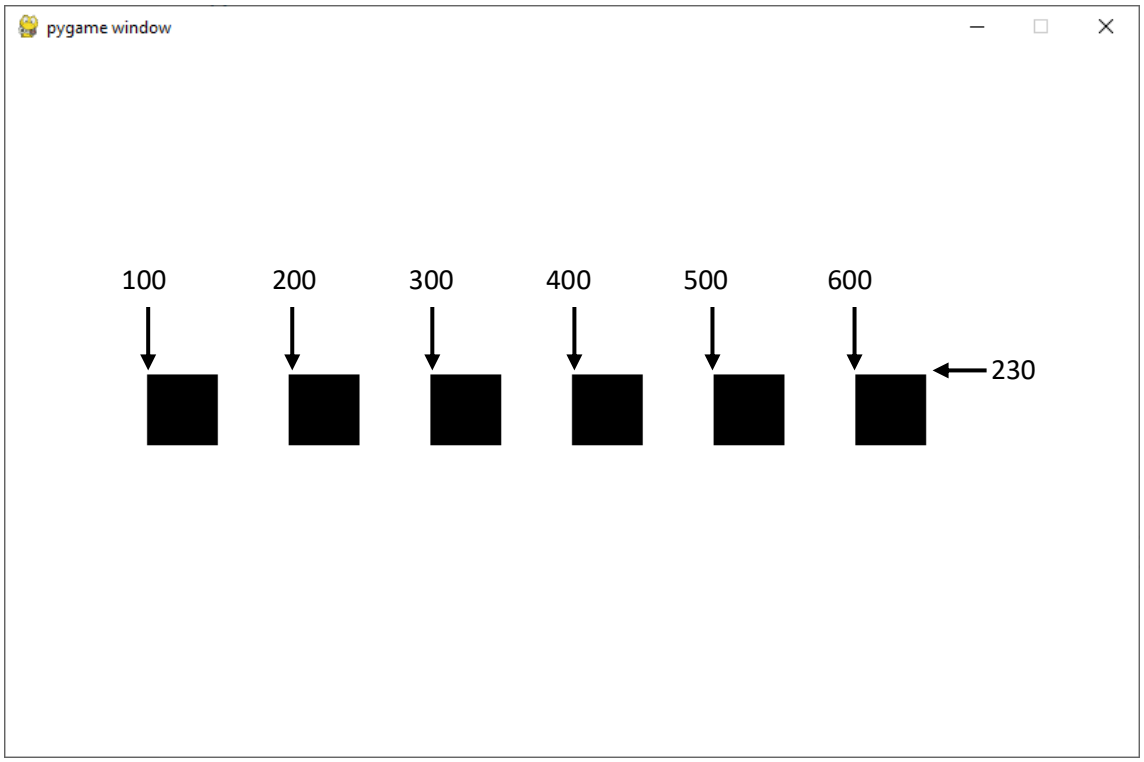
Vamos a continuar con el mismo archivo.

Vamos a dibujar figuras utilizando el for loops verás que resultados tan incesantes.

```
21      ### -------- ZONA DE DIBUJO
22      for x in range(100, 700, 100):
23          pygame.draw.rect(screen, BLACK, (x, 230, 50, 50))
24
25      ### -------- ZONA DE DIBUJO
```

En la línea 22 hacemos un ciclo for donde x asumirá al principio el valor 100, tiene que llegar a 700 pero con un incremento de 100, es decir la x valdrá 100, 200, 300, 400, 500 y 600. A 700 no llega.

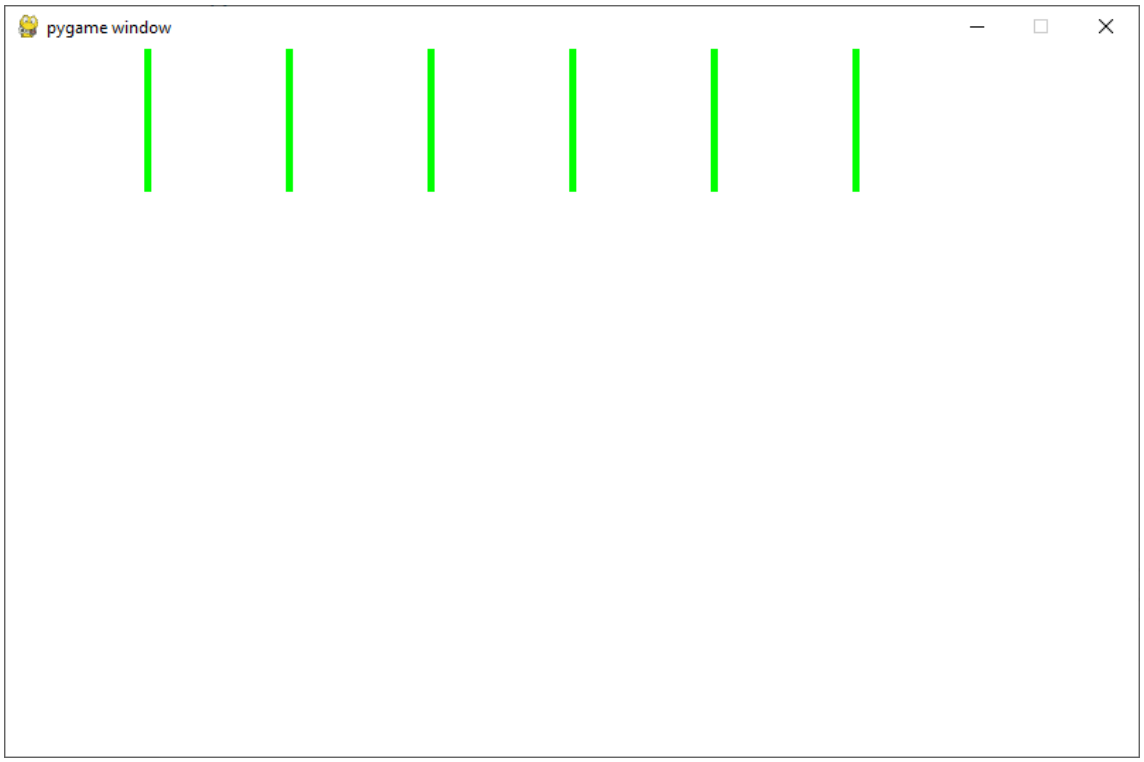
Como este bucle se repite 6 veces la coordenada x irá asumiendo los valores comentados anterior mente, si ejecutamos este será el resultado:



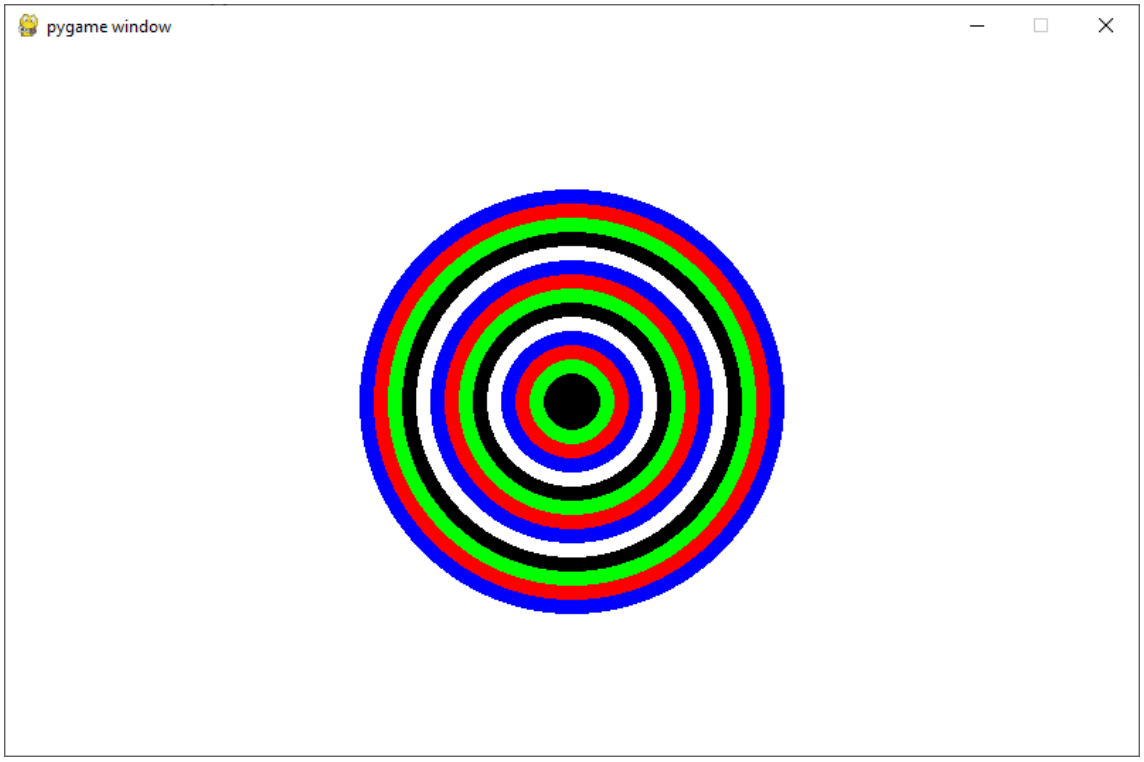
Vamos a ver otro ejemplo:

```
21      ### -------- ZONA DE DIBUJO
22      for x in range(100, 700, 100):
23          pygame.draw.line(screen, GREEN, (x, 0), (x, 100), 5)
24
25      ### -------- ZONA DE DIBUJO
```

Utilizando el mismo for vamos ahora a dibujar líneas:



Te recomiendo que juegues, que practiques para hacer figuras interesantes, te propongo este ejemplo.



En la siguiente página tienes la solución.

```python
import pygame, sys
pygame.init()

# Definir colores
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
GREEN = (0, 255, 0)
RED = (255, 0, 0)
BLUE = (0, 0, 255)

COLORES = [WHITE, BLACK, GREEN, RED, BLUE,
           WHITE, BLACK, GREEN, RED, BLUE,
           WHITE, BLACK, GREEN, RED, BLUE]

size = (800, 500)

screen = pygame.display.set_mode(size)

while True:
    for event in pygame.event.get():
        if event.type== pygame.QUIT:
            sys.exit()
    # Pinta la ventana de color blanco
    screen.fill(WHITE)
    ### -------- ZONA DE DIBUJO
    for x in range(14, 0, -1):
        pygame.draw.circle(screen, COLORES[x], (400, 250), (x+1) * 10)

    ### -------- ZONA DE DIBUJO

    # Actualiza la ventana
    pygame.display.flip()
```

4.- Animaciones

Para este capítulo vamos a un archivo nuevo llamado animación.py donde copiaremos el código del documento intro.py (que utilizamos como plantilla).

```
1   import pygame, sys
2   pygame.init()
3
4   # Definir color
5   RED = (255, 0, 0); WHITE = (255, 255, 255)
6
7   size = (800, 500)
8   screen = pygame.display.set_mode(size)
9
10  # Controlar FPS
11  clock = pygame.time.Clock()
12
13  # Coordenadas del cuadrado
14  cord_x = 400
15  cord_y = 200
16
17  # Velocidad a la que se moverá mi cuadrado
18  speed_x = 3
19  speed_y = 3
20
21  while True:
22      for event in pygame.event.get():
23          if event.type== pygame.QUIT:
24              sys.exit()
25      if(cord_x > 720 or cord_x < 0):
26          speed_x *= -1
27      if(cord_y > 420 or cord_y < 0):
28          speed_y *= -1
29      cord_x += speed_x
30      cord_y += speed_y
31      # Pinta la ventana de color blanco
32      screen.fill(WHITE)
33      ### -------- ZONA DE DIBUJO
34      pygame.draw.rect(screen, RED, (cord_x, cord_y, 80, 80))
35      ### -------- ZONA DE DIBUJO
36
37      pygame.display.flip()
38      clock.tick(60)
```

Creamos un objeto llamado clock de tipo time.ClocK() que nos permitirá controlar en el bucle los fotogramas por segundo.

Coordenadas iniciales del cuadrado.

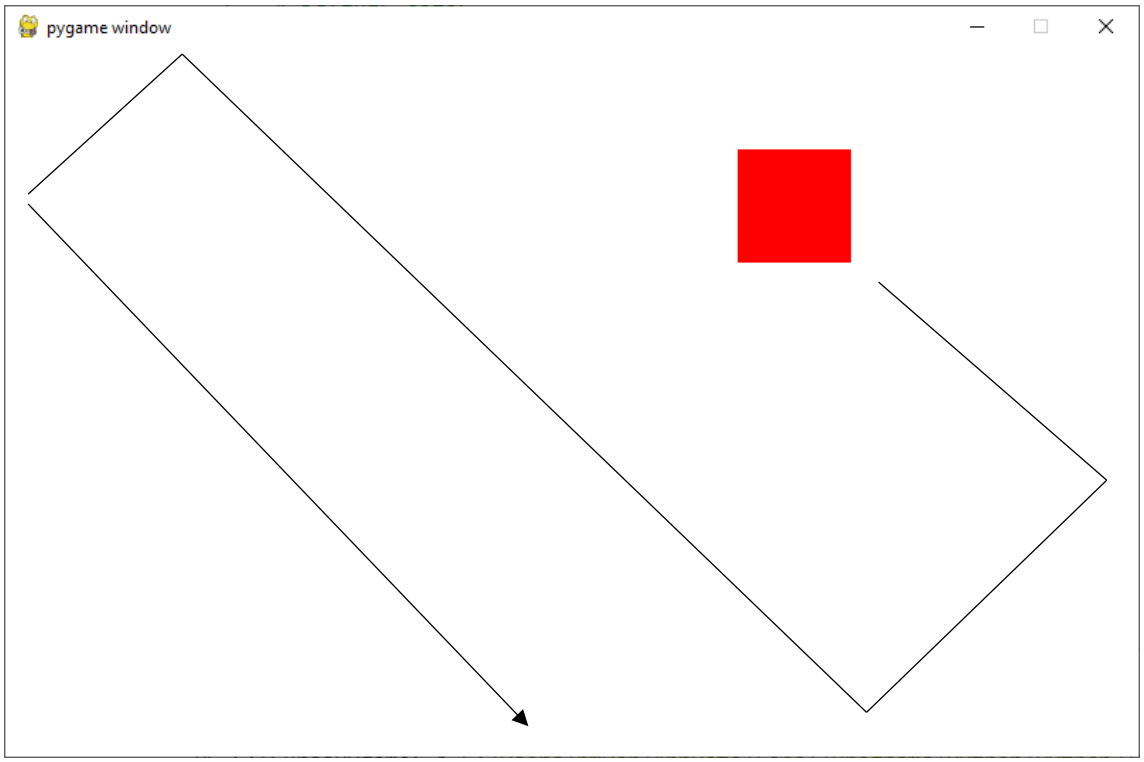
Velocidad tanto en la coordenada x como la coordenada y

Si la coordenada x es mayor de 720 o menor de 0, multiplicamos speed_x por -1 de este modo invertimos su dirección, del mismo modo lo hacemos con la coordendada speed_y si es mayor a 420 o menor a 0.

Uso de contadores

Variando las coordenadas da un efecto de movimiento.

Este será el resultado:



Hace una animación de efecto de rebote.

Todo el código que hace que el cuadrado se mueve el la parte de la lógica.

```
21  while True:
22      for event in pygame.event.get():
23          if event.type== pygame.QUIT:
24              sys.exit()
25      # --- LOGICA -----------------
26      if(cord_x > 720 or cord_x < 0):
27          speed_x *= -1
28      if(cord_y > 420 or cord_y < 0):
29          speed_y *= -1
30      cord_x += speed_x
31      cord_y += speed_y
32      # --- LOGICA -----------------
33      # Pinta la ventana de color blanco
34      screen.fill(WHITE)
35      ### -------- ZONA DE DIBUJO
36      pygame.draw.rect(screen, RED, (cord_x, cord_y, 80, 80))
37      ### -------- ZONA DE DIBUJO
38
39      pygame.display.flip()
40      clock.tick(60)
```

Dentro del bucle hemos delimitado la zona de programación perteneciente a la lógica, cuando realicemos un proyecto nuevo ya dejaremos limitada la zona.

Así será la nueva plantilla.

```python
import pygame, sys
pygame.init()

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
GREEN = (0, 255, 0)
RED = (255, 0, 0)
BLUE = (0, 0, 255)

size = (800, 500)
screen = pygame.display.set_mode(size)

# Controlar FPS
clock = pygame.time.Clock()

# Inicializar variables

while True:
    for event in pygame.event.get():
        if event.type== pygame.QUIT:
            sys.exit()
    # --- LOGICA -----------------

    # --- LOGICA -----------------
    # Pinta la ventana de color blanco
    screen.fill(WHITE)
    ### -------- ZONA DE DIBUJO

    ### -------- ZONA DE DIBUJO

    pygame.display.flip()
    clock.tick(60)
```

5.- Animaciones parte 2

Para este capítulo vamos a crear un nuevo archivo llamado lluvia.py y copiando la plantilla.

```
1   import pygame, sys
2   pygame.init()
3
4   WHITE = (255, 255, 255)
5   RED = (255, 0, 0)
6
7   size = (800, 500)
8   screen = pygame.display.set_mode(size)
9   clock = pygame.time.Clock()
10
11  while True:
12      for event in pygame.event.get():
13          if event.type== pygame.QUIT:
14              sys.exit()
15
16      screen.fill(WHITE)
17      pygame.display.flip()
18      clock.tick(30)
```

Hemos eliminado lo que no necesitábamos, podemos ejecutar para ver su funciona correctamente.

Vamos a escribir el siguiente código:

```
1   import pygame, sys, random
2   pygame.init()
3
4   WHITE = (255, 255, 255)
5   RED = (255, 0, 0)
6
7   size = (800, 500)
8   screen = pygame.display.set_mode(size)
9   clock = pygame.time.Clock()
10
11  coor_list = []
12  for i in range(60):
13      x = random.randint(0, 800)
14      y = random.randint(0, 500)
15      coor_list.append([x, y])
16
17  while True:
18      for event in pygame.event.get():
19          if event.type== pygame.QUIT:
20              sys.exit()
21
22      screen.fill(WHITE)
23
24      for coord in coor_list:
25          x = coord[0]
26          y = coord[1]
27          pygame.draw.circle(screen, RED, coord, 2)
28          coord[1] += 1
29          if coord[1] > 500:
30              coord[1] = 0
31      pygame.display.flip()
32      clock.tick(30)
```

Creamos una lista vacía llamada coor_list. Como queremos dibujar 60 puntos hacemos un ciclo for para que se repita 60 veces. La variable x recibe un valor aleatorio entre 0 y 800. La variable y recibe un valor aleatorio entre 0 y 500. Los valores [x, y] se añaden a la lista coor_list.

Hacemos un for que se repite tantas veces como elementos tiene coor_list (60). A x se le asigna coord[0] y a y se le asigna coord.[1].

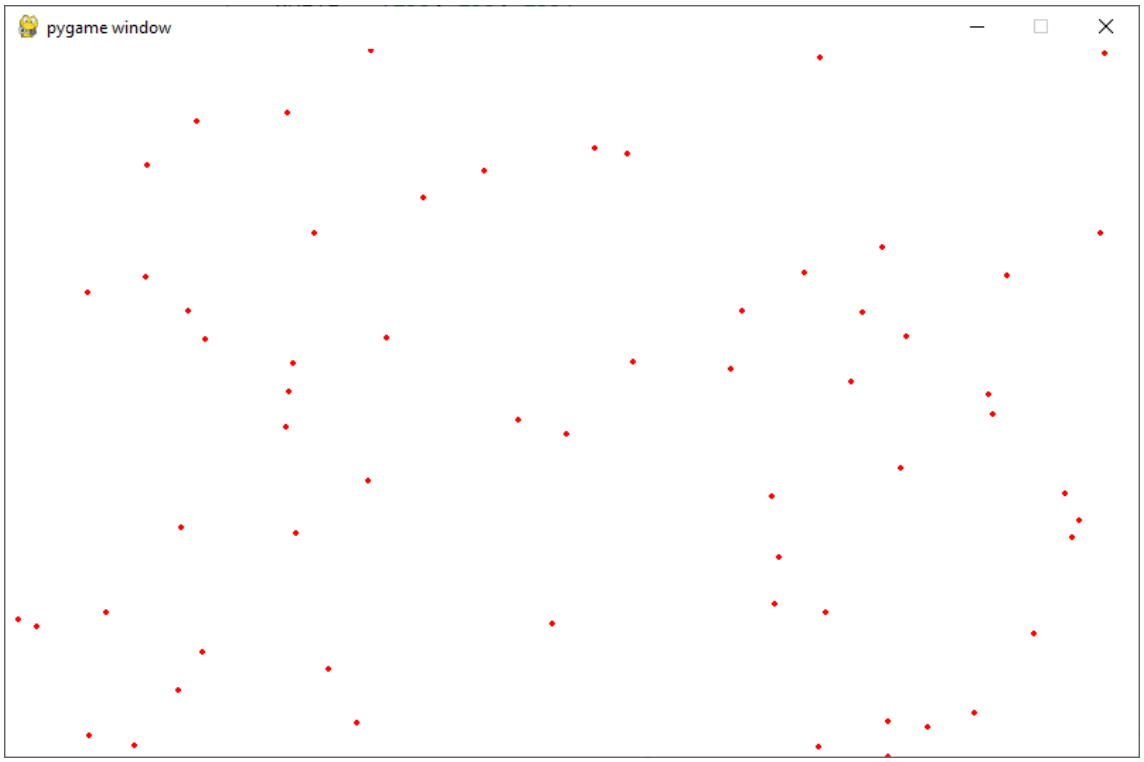
Dibujamos un circulo de 2 de radio, que parece un punto, según los valores de coord.

La coodenada y que es coord[1] a su valor le incrementamos 1.

Con la condición contro-lamos que si coord[1] es mayor de 500 que vuelva a valer 0 de este modo los puntos que se pierde por abajo aparecen de nuevo por arriba.

Cuando se actualiza la ventana nos da un efecto de lluvia.

Este será el resultado:



6.- Mouse

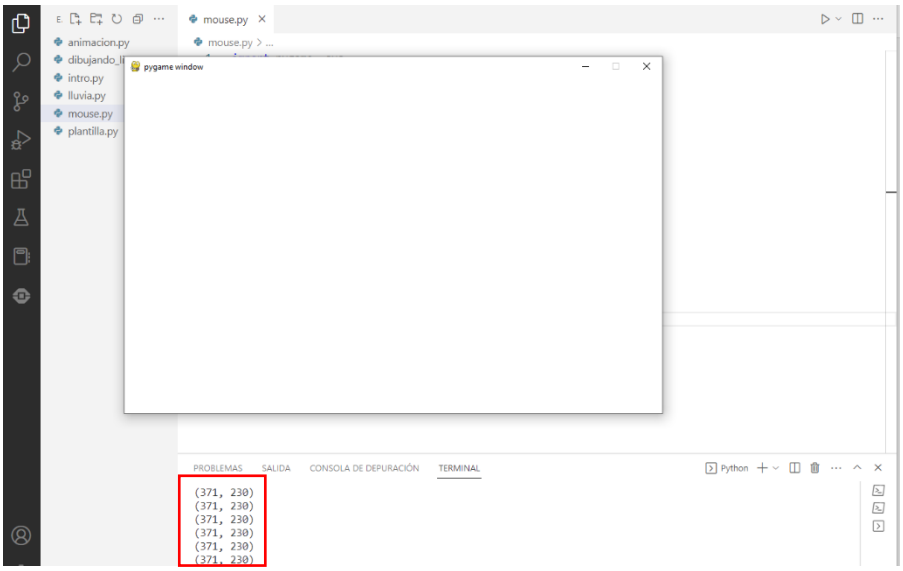
En este capítulo vamos a mover objetos utilizando nuestro mouse, para esto vamos a realizar un nuevo proyecto llamado mouse.py y copiando la plantilla.

```
1   import pygame, sys
2   pygame.init()
3
4   BLACK = (0, 0, 0)
5   WHITE = (255, 255, 255)
6   RED = (255, 0, 0)
7
8   size = (800, 500)
9   screen = pygame.display.set_mode(size)
10  clock = pygame.time.Clock()
11
12  while True:
13      for event in pygame.event.get():
14          if event.type== pygame.QUIT:
15              sys.exit()
16
17      mouse_pos = pygame.mouse.get_pos()
18      print(mouse_pos)
19      screen.fill(WHITE)
20
21      pygame.display.flip()
22      clock.tick(60)
```

El método .mouse.get_pos() nos retorna dos valores la posición x e y de nuestro mouse en la ventana.

Para demostrarlo hemos añadido un print de dichos valores.

Este es el resultado cuando ejecutamos.



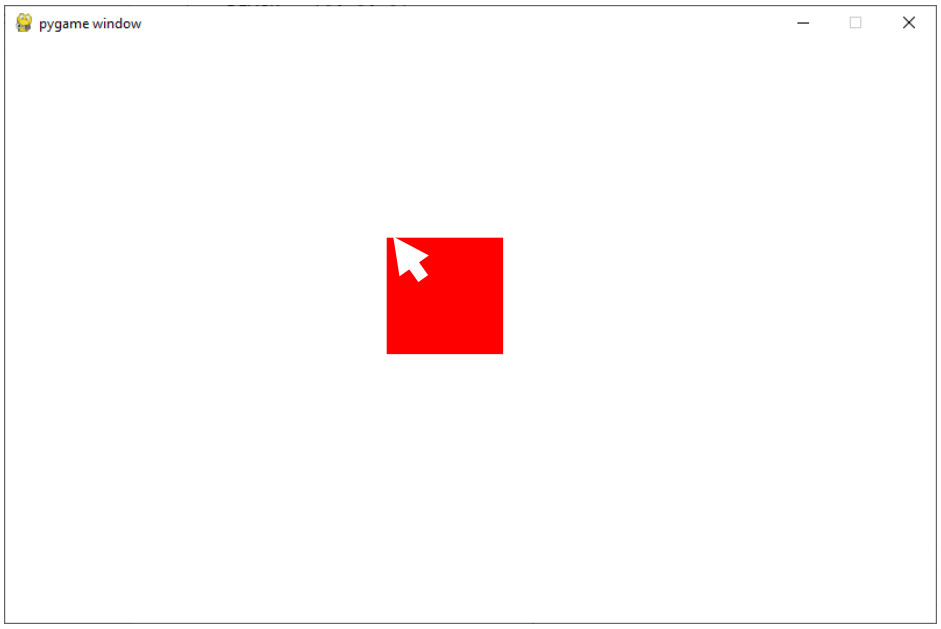
Vemos las coordenadas de nuestro ratón mientras se mueve por la ventana.

Vamos a seguir con el código:

```
1   import pygame, sys
2   pygame.init()
3
4   BLACK = (0, 0, 0)
5   WHITE = (255, 255, 255)
6   RED = (255, 0, 0)
7
8   size = (800, 500)
9   screen = pygame.display.set_mode(size)
10  clock = pygame.time.Clock()
11
12  while True:
13      for event in pygame.event.get():
14          if event.type== pygame.QUIT:
15              sys.exit()
16
17      mouse_pos = pygame.mouse.get_pos()
18      x = mouse_pos[0]
19      y = mouse_pos[1]
20
21      screen.fill(WHITE)
22      pygame.draw.rect(screen, RED, (x, y, 100, 100))
23      pygame.display.flip()
24      clock.tick(60)
```

x e y asumen las coordenadas del puntero del ratón cuando se mueve por la ventana.

Estos valores son asignados al cuadrado.



Mueves el ratón y el cuadro hace el mismo movimiento.

Si queremos que nuestro mouse no esté visible fuera del bucle pondremos:

pygame.mouse.set_visible(0) → 0 que no esté visible 1 que esté visible.

7.- Keyboard / Teclado

En este capítulo vamos a mover objeto con el teclado, vamos a seguir con el mismo código del capítulo anterior.

```
1   import pygame, sys
2   pygame.init()
3
4   BLACK = (0, 0, 0)
5   WHITE = (255, 255, 255)
6   RED = (255, 0, 0)
7
8   size = (800, 500)
9   screen = pygame.display.set_mode(size)
10  clock = pygame.time.Clock()
11
12  while True:
13      for event in pygame.event.get():
14          if event.type== pygame.QUIT:
15              sys.exit()
16
17      screen.fill(WHITE)
18      # pygame.draw.rect(screen, RED, (x, y, 100, 100))
19      pygame.display.flip()
20      clock.tick(60)
```

Hemos borrado todo lo referente al mouse, así te tiene que quedar.

Hemos comentado la línea 18 por las variables x e y no están inicializadas.

Adjunto relación de teclas

```
pygame
Constant      ASCII   Description
--------------------------------
K_BACKSPACE   \b      backspace
K_TAB         \t      tab
K_CLEAR               clear
K_RETURN      \r      return
K_PAUSE               pause
K_ESCAPE      ^[      escape
K_SPACE               space
K_EXCLAIM     !       exclaim
K_QUOTEDBL    "       quotedbl
K_HASH        #       hash
K_DOLLAR      $       dollar
K_AMPERSAND   &       ampersand
K_QUOTE               quote
K_LEFTPAREN   (       left parenthesis
K_RIGHTPAREN  )       right parenthesis
K_ASTERISK    *       asterisk
```

```
K_PLUS          +       plus sign
K_COMMA         ,       comma
K_MINUS         -       minus sign
K_PERIOD        .       period
K_SLASH         /       forward slash
K_0             0       0
K_1             1       1
K_2             2       2
K_3             3       3
K_4             4       4
K_5             5       5
K_6             6       6
K_7             7       7
K_8             8       8
K_9             9       9
K_COLON         :       colon
K_SEMICOLON     ;       semicolon
K_LESS          <       less-than sign
K_EQUALS        =       equals sign
K_GREATER       >       greater-than sign
K_QUESTION      ?       question mark
K_AT            @       at
K_LEFTBRACKET   [       left bracket
K_BACKSLASH     \       backslash
K_RIGHTBRACKET  ]       right bracket
K_CARET         ^       caret
K_UNDERSCORE    _       underscore
K_BACKQUOTE     `       grave
K_a             a       a
K_b             b       b
K_c             c       c
K_d             d       d
K_e             e       e
K_f             f       f
K_g             g       g
K_h             h       h
K_i             i       i
K_j             j       j
K_k             k       k
K_l             l       l
K_m             m       m
K_n             n       n
K_o             o       o
K_p             p       p
K_q             q       q
K_r             r       r
K_s             s       s
K_t             t       t
K_u             u       u
K_v             v       v
K_w             w       w
K_x             x       x
K_y             y       y
K_z             z       z
K_DELETE                delete
K_KP0                   keypad 0
K_KP1                   keypad 1
K_KP2                   keypad 2
K_KP3                   keypad 3
```

```
K_KP4                keypad 4
K_KP5                keypad 5
K_KP6                keypad 6
K_KP7                keypad 7
K_KP8                keypad 8
K_KP9                keypad 9
K_KP_PERIOD   .      keypad period
K_KP_DIVIDE   /      keypad divide
K_KP_MULTIPLY *      keypad multiply
K_KP_MINUS    -      keypad minus
K_KP_PLUS     +      keypad plus
K_KP_ENTER    \r     keypad enter
K_KP_EQUALS   =      keypad equals
K_UP                 up arrow
K_DOWN               down arrow
K_RIGHT              right arrow
K_LEFT               left arrow
K_INSERT             insert
K_HOME               home
K_END                end
K_PAGEUP             page up
K_PAGEDOWN           page down
K_F1                 F1
K_F2                 F2
K_F3                 F3
K_F4                 F4
K_F5                 F5
K_F6                 F6
K_F7                 F7
K_F8                 F8
K_F9                 F9
K_F10                F10
K_F11                F11
K_F12                F12
K_F13                F13
K_F14                F14
K_F15                F15
K_NUMLOCK            numlock
K_CAPSLOCK           capslock
K_SCROLLOCK          scrollock
K_RSHIFT             right shift
K_LSHIFT             left shift
K_RCTRL              right control
K_LCTRL              left control
K_RALT               right alt
K_LALT               left alt
K_RMETA              right meta
K_LMETA              left meta
K_LSUPER             left Windows key
K_RSUPER             right Windows key
K_MODE               mode shift
K_HELP               help
K_PRINT              print screen
K_SYSREQ             sysrq
K_BREAK              break
K_MENU               menu
K_POWER              power
K_EURO               Euro
K_AC_BACK            Android back button
```

Este será el Código:

```
1   import pygame, sys
2   pygame.init()
3
4   BLACK = (0, 0, 0)
5   WHITE = (255, 255, 255)
6   RED = (255, 0, 0)
7
8   size = (800, 500)
9   screen = pygame.display.set_mode(size)
10  clock = pygame.time.Clock()
11
12  # Coordenadas cuadrado
13  coord_x = 10
14  coord_y = 10
15  # Velocidad
16  x_speed = 0
17  y_speed = 0
18
19  while True:
20      for event in pygame.event.get():
21          if event.type== pygame.QUIT:
22              sys.exit()
23          # Eventos teclado
24          if event.type == pygame.KEYDOWN: # Presionar tecla
25              if event.key == pygame.K_LEFT:
26                  x_speed = -3
27              if event.key == pygame.K_RIGHT:
28                  x_speed = 3
29
30          if event.type == pygame.KEYUP: # Dejar de presionar tecla
31              if event.key == pygame.K_LEFT:
32                  x_speed = 0
33              if event.key == pygame.K_RIGHT:
34                  x_speed = 0
35
36      screen.fill(WHITE)
37      coord_x += x_speed
38      pygame.draw.rect(screen, RED, (coord_x, coord_y, 100, 100))
39      pygame.display.flip()
40      clock.tick(60)
```

Definimos las coordenadas iniciales para nuestro cuadrado.

Definimos la velocidad en x y en y con un valor inicial a 0.

Controlamos si pulsamos las teclas derecha o izquierda para dar valor a x_speed -3 0 +3

Controlamos si soltamos las teclas derecha o izquierda para dar valor a x_speed 0

Según el valor x_speed este se incrementará o disminuirá la variable coord_x con lo que el cuadrado se moverá de izquierda a derecha o de derecha a izquierda.

Ahora te voy a proponer que con la fecha arriba y flecha abajo puedas mover el cuadrado en dichas direcciones.

```
import pygame, sys
pygame.init()

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)

size = (800, 500)
screen = pygame.display.set_mode(size)
clock = pygame.time.Clock()

# Coordenadas cuadrado
coord_x = 10
coord_y = 10
# Velocidad
x_speed = 0
y_speed = 0

while True:
    for event in pygame.event.get():
        if event.type== pygame.QUIT:
            sys.exit()
        # Eventos teclado
        if event.type == pygame.KEYDOWN: # Presionar tecla
            if event.key == pygame.K_LEFT:
                x_speed = -3
            if event.key == pygame.K_RIGHT:
                x_speed = 3
            if event.key == pygame.K_UP:
                y_speed = -3
            if event.key == pygame.K_DOWN:
                y_speed = 3

        if event.type == pygame.KEYUP: # Dejar de presionar tecla
            if event.key == pygame.K_LEFT:
                x_speed = 0
            if event.key == pygame.K_RIGHT:
                x_speed = 0
            if event.key == pygame.K_UP:
                y_speed = 0
            if event.key == pygame.K_DOWN:
                y_speed = 0

    screen.fill(WHITE)
    coord_x += x_speed
    coord_y += y_speed
```

```
    pygame.draw.rect(screen, RED, (coord_x, coord_y, 100, 100))
    pygame.display.flip()
    clock.tick(60)
```

Si te has dado cuenta el cuadrado se sale de la ventana, modifica el código para que el cuadrado cuando llegue a los bordes no se mueva y así impedir que salga de la ventana.

```
import pygame, sys
pygame.init()

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)

size = (800, 500)
screen = pygame.display.set_mode(size)
clock = pygame.time.Clock()

# Coordenadas cuadrado
coord_x = 10
coord_y = 10
# Velocidad
x_speed = 0
y_speed = 0

while True:
    for event in pygame.event.get():
        if event.type== pygame.QUIT:
            sys.exit()
        # Eventos teclado
        if event.type == pygame.KEYDOWN: # Presionar tecla
            if event.key == pygame.K_LEFT:
                x_speed = -3
            if event.key == pygame.K_RIGHT:
                x_speed = 3
            if event.key == pygame.K_UP:
                y_speed = -3
            if event.key == pygame.K_DOWN:
                y_speed = 3

        if event.type == pygame.KEYUP: # Dejar de presionar tecla
            if event.key == pygame.K_LEFT:
                x_speed = 0
            if event.key == pygame.K_RIGHT:
                x_speed = 0
            if event.key == pygame.K_UP:
                y_speed = 0
            if event.key == pygame.K_DOWN:
```

```
            y_speed = 0

screen.fill(WHITE)
if coord_x > 700:
    coord_x = 700
elif coord_x < 0:
    coord_x = 0
else:
    coord_x += x_speed

if coord_y > 400:
    coord_y = 400
elif coord_y < 0:
    coord_y = 0
else:
    coord_y += y_speed

pygame.draw.rect(screen, RED, (coord_x, coord_y, 100, 100))
pygame.display.flip()
clock.tick(60)
```

8.- Pong

Con todo lo visto anteriormente ya tenemos todos los elementos necesarios para poder crear un juego muy sencillo.

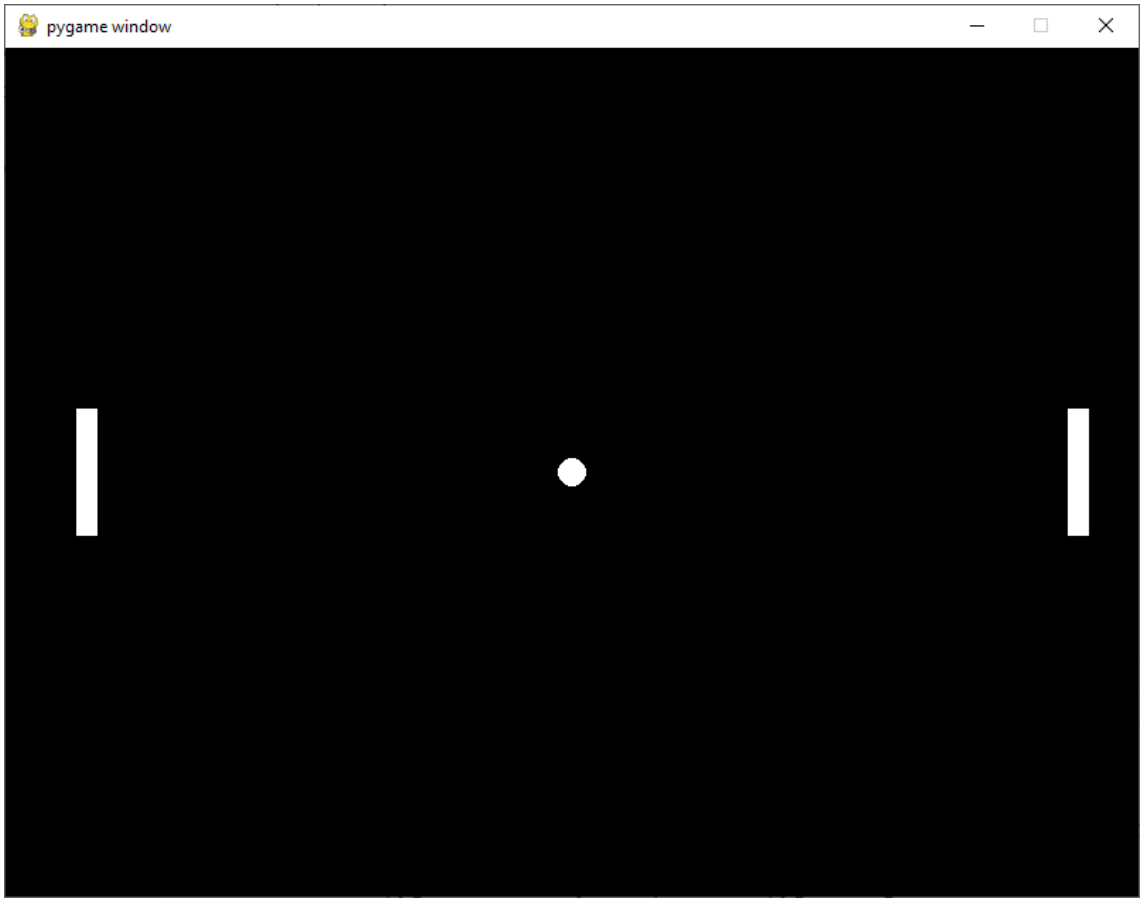
Vamos a crear el juego de pong, es un juego muy simple nos vamos a dedicar a crear o programar la mecánica del juego.

Vamos a empezar un nuevo proyecto llamado pong.py.

```
import pygame
pygame.init()

# Colores
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
screen_size = (800, 600)
player_width = 15
player_height = 90

screen = pygame.display.set_mode(screen_size)
clock = pygame.time.Clock()

# Coordenadas y velocidad del jugador 1
player1_x_coor = 50
player1_y_coor = 300 - 45
player1_y_speed = 0
# Coordenadas y velocidad del jugador 2
player2_x_coor = 750
player2_y_coor = 300 - 45
player2_y_speed = 0
# Coordenadas de la pelota
pelota_x = 400
pelota_y = 300
pelota_speed_x = 0
pelota_speed_y = 3

game_over = False

while not game_over:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            game_over = True

    screen.fill(BLACK)
    # Zona de dibujo
    jugador1 = pygame.draw.rect(screen, WHITE, (player1_x_coor,
player1_y_coor, player_width, player_height))
    jugador2 = pygame.draw.rect(screen, WHITE, (player2_x_coor,
player2_y_coor, player_width, player_height))
```

```
    pelota = pygame.draw.circle(screen, WHITE, (pelota_x, pelota_y), 10)
    pygame.display.flip()
    clock.tick(60)
pygame.quit()
```

Ya tenemos todos los elementos gráficos del juego.



Vamos a escribir el código añadiendo la parte lógica:

```
import pygame
pygame.init()

# Colores
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
screen_size = (800, 600)
player_width = 15
player_height = 90

screen = pygame.display.set_mode(screen_size)
clock = pygame.time.Clock()

# Coordenadas y velocidad del jugador 1
player1_x_coor = 50
player1_y_coor = 300 - 45
```

```
player1_y_speed = 0
# Coordenadas y velocidad del jugador 2
player2_x_coor = 750
player2_y_coor = 300 - 45
player2_y_speed = 0
# Coordenadas de la pelota
pelota_x = 400
pelota_y = 300
pelota_speed_x = 3
pelota_speed_y = 3

game_over = False

while not game_over:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            game_over = True
        if event.type == pygame.KEYDOWN:
            # Jugador 1
            if event.key == pygame.K_w:
                player1_y_speed = -3
            if event.key == pygame.K_s:
                player1_y_speed = 3
            # Jugador 2
            if event.key == pygame.K_UP:
                player2_y_speed = -3
            if event.key == pygame.K_DOWN:
                player2_y_speed = 3

        if event.type == pygame.KEYUP:
            # Jugador 1
            if event.key == pygame.K_w:
                player1_y_speed = 0
            if event.key == pygame.K_s:
                player1_y_speed = 0
            # Jugador 2
            if event.key == pygame.K_UP:
                player2_y_speed = 0
            if event.key == pygame.K_DOWN:
                player2_y_speed = 0

    if pelota_y > 590 or pelota_y < 10:
        pelota_speed_y *= -1

    # Revisa si la pelota sale del lado derecho
    if pelota_x > 800:
        pelota_x = 400
        pelota_y = 300
```

```
            # Si sale de la pantalla, invierte dirección
            pelota_speed_x *= -1
            pelota_speed_y *= -1

        if pelota_x < 0:
            pelota_x = 400
            pelota_y = 300

            # Si sale de la pantalla, invierte dirección
            pelota_speed_x *= -1
            pelota_speed_y *= -1

        # Modifica las coordenadas para dar mov. a los jugadores/pelota
        player1_y_coor += player1_y_speed
        player2_y_coor += player2_y_speed
        # Movimiento pelora
        pelota_x += pelota_speed_x
        pelota_y += pelota_speed_y

        screen.fill(BLACK)
        # Zona de dibujo
        jugador1 = pygame.draw.rect(screen, WHITE, (player1_x_coor,
player1_y_coor, player_width, player_height))
        jugador2 = pygame.draw.rect(screen, WHITE, (player2_x_coor,
player2_y_coor, player_width, player_height))
        pelota = pygame.draw.circle(screen, WHITE, (pelota_x, pelota_y), 10)

        # Colisiones
        if pelota.colliderect(jugador1) or pelota.colliderect(jugador2):
            pelota_speed_x *= -1

        pygame.display.flip()
        clock.tick(60)
pygame.quit()
```

Los dos jugadores pueden salirse por la parte superior e inferior de la ventana, el reto es controlar que no se puedan salir.

Hemos modificado el apartado siguiente:

```
# Modifica las coordenadas para dar mov. a los jugadores/pelota
    if player1_y_coor < 0:
        player1_y_coor = 0
    else:
        player1_y_coor += player1_y_speed
    if player1_y_coor > 510:
        player1_y_coor = 510
    else:
        player1_y_coor += player1_y_speed
```

```
if player2_y_coor < 0:
    player2_y_coor = 0
else:
    player2_y_coor += player2_y_speed
if player2_y_coor > 510:
    player2_y_coor = 510
else:
    player2_y_coor += player1_y_speed
```

9.- Imagen de Fondo

Para este capitulo vamos a necesitar el siguiente archivo.

background.png

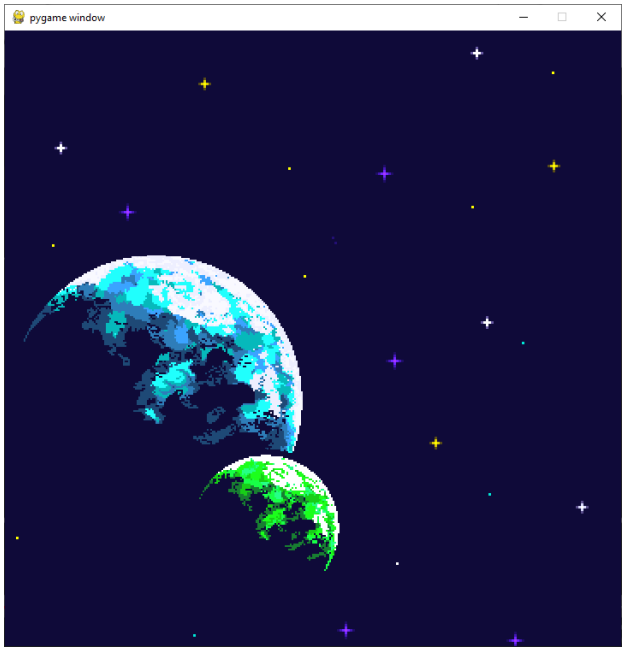
Lo podrás descargar del siguiente enlace: https://github.com/mundo-python/pygame-Scripts

En la misma carpeta donde esta el archivo de imagen crearemos el archivo background_01.py donde escribiremos el código.

```
1   import pygame
2   screen = pygame.display.set_mode([720, 720])
3   clock = pygame.time.Clock()
4
5   done = False
6
7   background = pygame.image.load("background.png").convert()
8
9   while not done:
10      for event in pygame.event.get():
11          if event.type == pygame.QUIT:
12              done = True
13
14      screen.blit(background, [0,0])
15
16      pygame.display.flip()
17      clock.tick(60)
18
19  pygame.quit()
```

Cargamos la imagen

Colocamos la imagen en la ventana screen en la coordenada 0,0.



pygame no va a redimensionar la imagen por ti, tiene que buscar imágenes que se ajusten a la ventana.

10.- Moviendo imágenes

En este capítulo vamos cargar una imagen que va a representar nuestro jugador y también vamos a mover esta imagen.

Necesitaremos la siguiente imagen que en capítulos anteriores te di el enlace para descargártela.

player.png

Vamos a crear otro documento llamado moving02.py donde copiaremos todo el código anterior.

```
1   import pygame
2   screen = pygame.display.set_mode([720, 720])
3   clock = pygame.time.Clock()
4
5   done = False
6
7   background = pygame.image.load("background.png").convert()
8   player = pygame.image.load("player.png").convert()
9   player.set_colorkey([0 ,0 ,0])
10
11  while not done:
12      for event in pygame.event.get():
13          if event.type == pygame.QUIT:
14              done = True
15
16      mouse_pos = pygame.mouse.get_pos()
17      x = mouse_pos[0]
18      y = mouse_pos[1]
19
20      screen.blit(background, [0,0])
21      screen.blit(player, [x, y])
22
23      pygame.display.flip()
24      clock.tick(60)
25
26  pygame.quit()
```

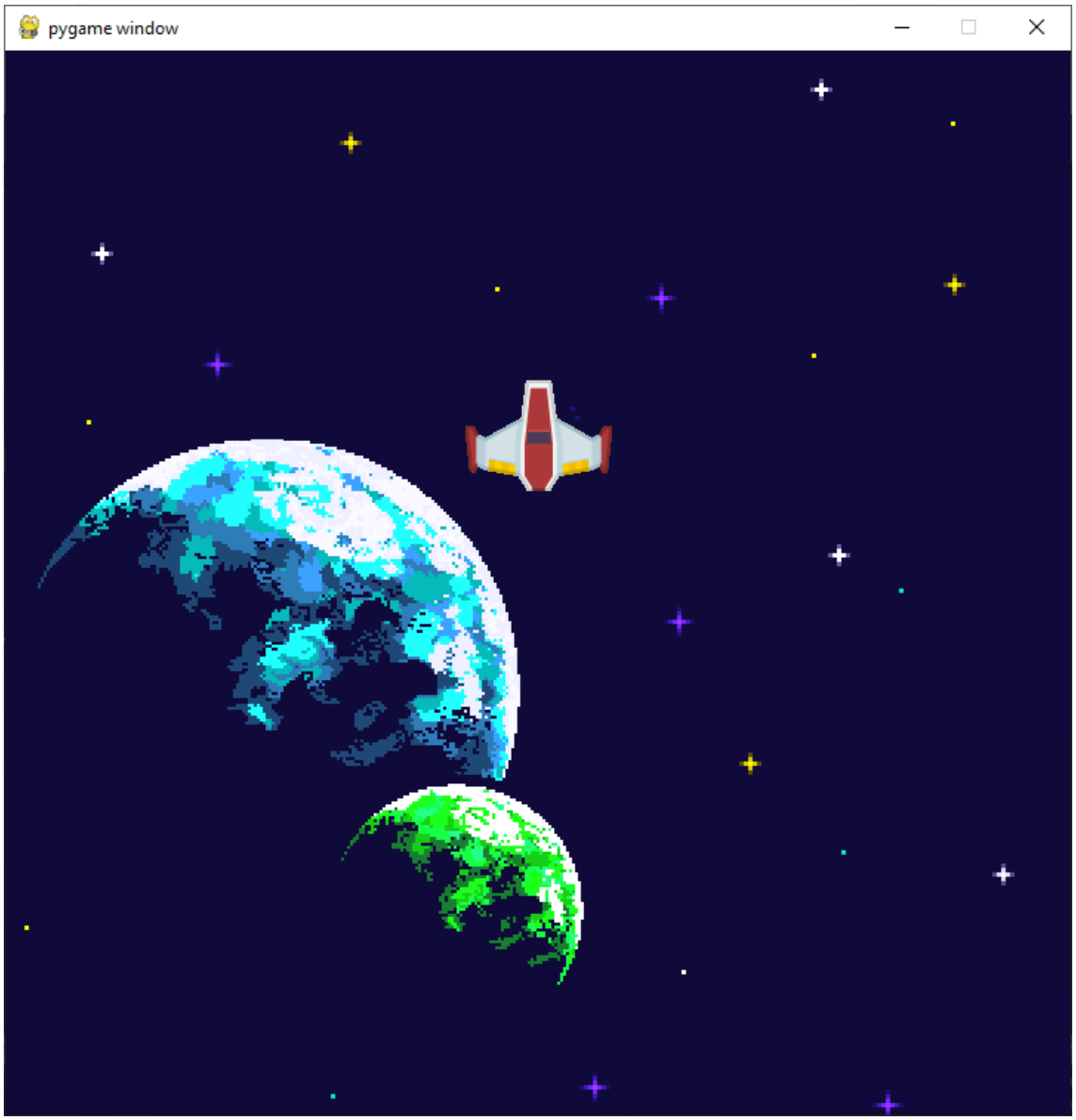
Agregamos la nave

Eliminamos el borde negro que lleva la imagen.

Obtenemos las coordenadas del puntero del mouse y se lo pasamos a las variables x e y.

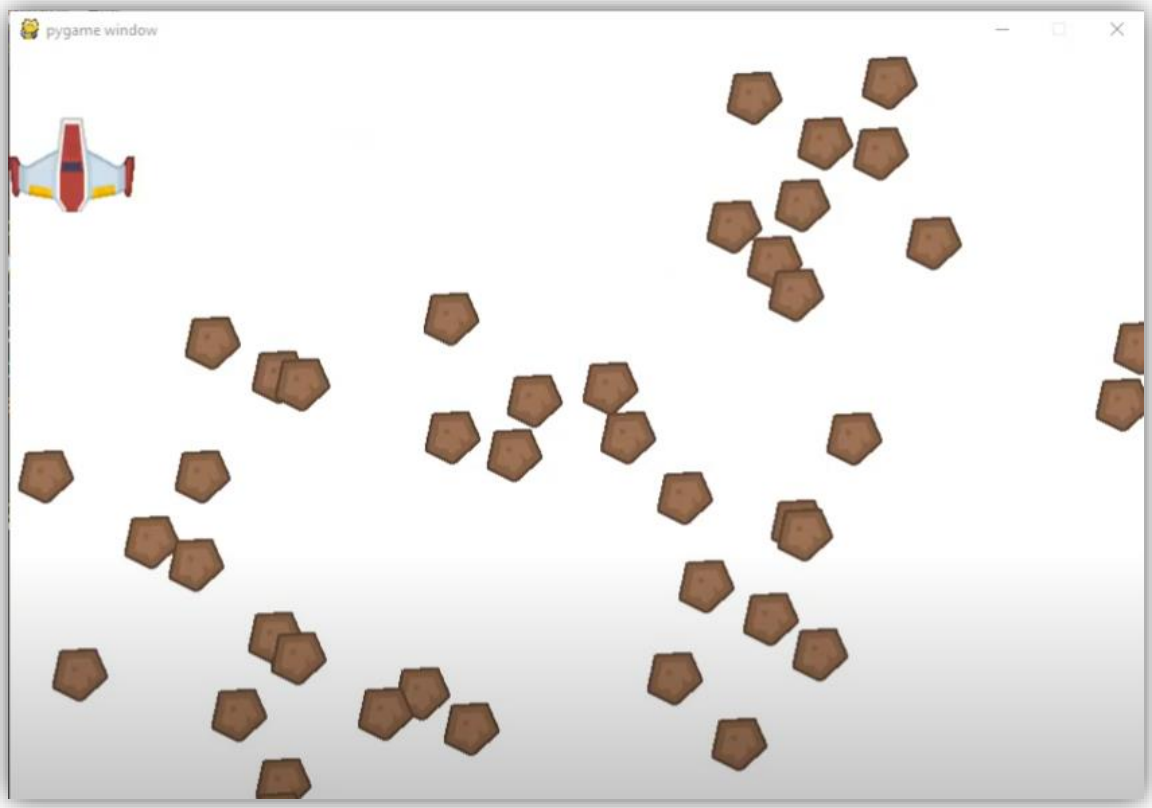
Colocamos la imagen en la ventana screen pero con las coordenadas del puntero del mouse, de este modo al mover el mouse movemos la nave.

Este será el resultado:


pygame window

11.- Sprites y Clases

Para este capítulo vamos a necesitar la siguiente imagen.



La nave tiene que tocar a los meteoritos y estos desaparecerán.
Además de mostrarse un contador por consola de los meteoritos que estamos eliminando, hay un total de 50 meteoritos.
Vamos a crear un nuevo proyecto llamado sprites.py.

```
import pygame, random

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
```

Creamos la clase Meteor

```
class Meteor(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("meteor.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()
```

```
class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("player.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()
```

Creamos al clase Player.

```
pygame.init()
screen = pygame.display.set_mode([900, 600])
clock = pygame.time.Clock()
done = False
score = 0

meteor_list = pygame.sprite.Group()
all_sprites_list = pygame.sprite.Group()
```

Inicializamos pygame

Definimos dos grupos uno llamado meteor_list y otro all_sprites_list.

```
for i in range(50):
    meteor = Meteor()
    meteor.rect.x = random.randrange(900)
    meteor.rect.y = random.randrange(600)
    meteor_list.add(meteor)
    all_sprites_list.add(meteor)
```

Creamos 50 meteor de la clase Meteor, las cooredenadas x e y con valores aleatorio a las dimensiones de la ventana.

Lo agregamos a la lista meteor_list y all_sprite_list que contiene todos los sprites.

```
player = Player()
all_sprites_list.add(player)

while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
```

Creamos el objeto player de la clase Player, y lo agregamos a all_sprite_list.

```
    mouse_pos = pygame.mouse.get_pos()
    player.rect.x = mouse_pos[0]
    player.rect.y = mouse_pos[1]
```

```
    meteor_hit_list = pygame.sprite.spritecollide(player,meteor_list,
True )
```

Leemos las coordenadas de nuestro mouse y se lo pasamos a las coordenadas de player.

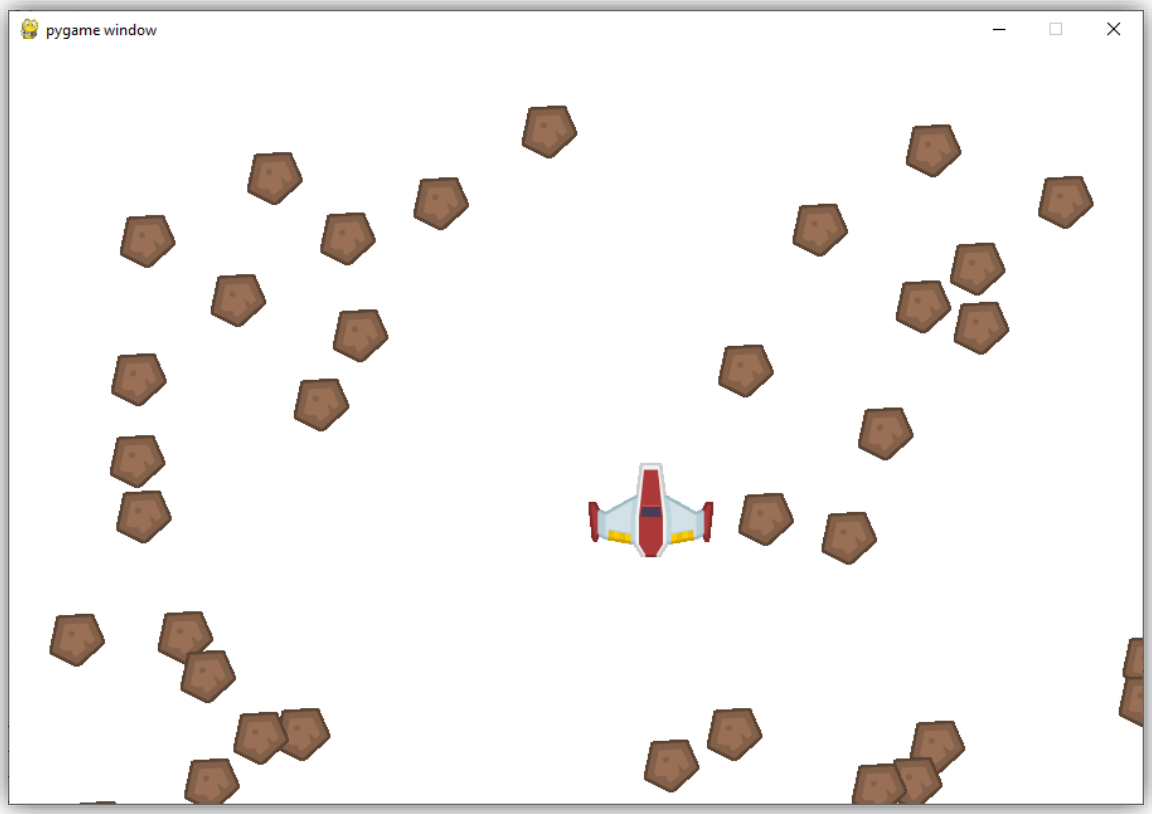
```
    for meteor in meteor_hit_list:
        score += 1
        print(score)

    screen.fill(WHITE)

    all_sprites_list.draw(screen)

    pygame.display.flip()
    clock.tick(60)
```

Si alguno de los meteoritos colisiona con la nave, los meteoritos desaparecen.

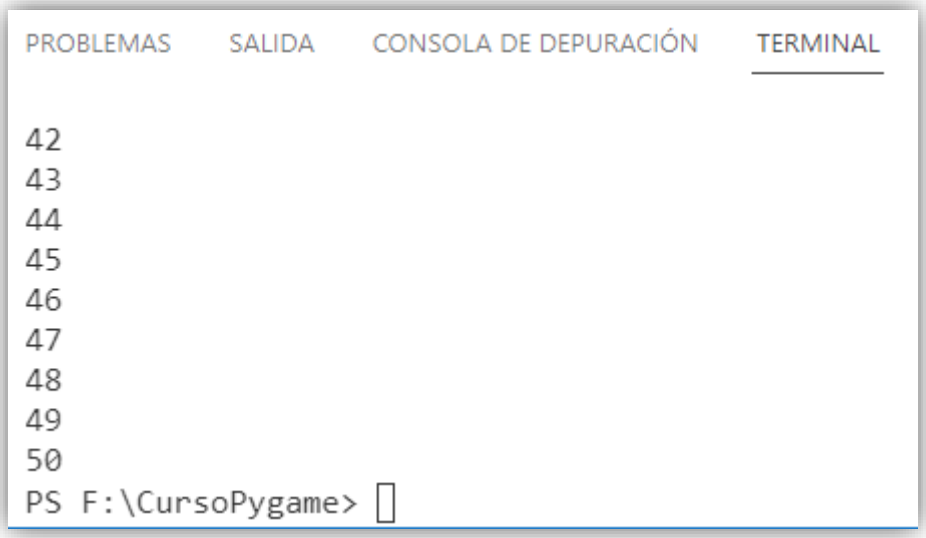
Hacemos un recorrido por todos los meteoritos, en caso de colisión el contador incrementa en 1.

Dibujamos todos los sprites que contiene la lista.

```
pygame.quit()
```



Se muestra por consola los meteoritos que vamos eliminado:



12.- Moviendo Sprites

Hemos modificado el código anterior:

El método update es una función que ya está definida dentro de la clase sprite que puede aceptar cualquier número de parámetros y realmente no contiene nada, entonces ahí se puede poner instrucciones que afecten a todas las instancias de una misma clase.

```
import pygame, random

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

class Meteor(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("meteor.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()

    def update(self):
        self.rect.y += 1

        if self.rect.y > 600:
            self.rect.y = -10
            self.rect.x = random.randrange(900)
```

Cuando los objetos de tipo Meteo llamen el método update, este será el que ejecutará.

```
class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("player.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()

    def update(self):
        mouse_pos = pygame.mouse.get_pos()
        player.rect.x = mouse_pos[0]
        player.rect.y = mouse_pos[1]
```

Cuando los objetos de tipo Player llamen el método update, este será el que ejecutará.

```
pygame.init()
screen = pygame.display.set_mode([900, 600])
clock = pygame.time.Clock()
done = False
score = 0

meteor_list = pygame.sprite.Group()
all_sprites_list = pygame.sprite.Group()
for i in range(50):
    meteor = Meteor()
    meteor.rect.x = random.randrange(900)
    meteor.rect.y = random.randrange(600)
```

Se define un grupo llamado all_sprite_list que contendrá los objetos de tipo Meteor y Player, que instanciemos en el programa.

```
    meteor_list.add(meteor)
    all_sprites_list.add(meteor)

player = Player()
all_sprites_list.add(player)

while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    all_sprites_list.update()

    meteor_hit_list = pygame.sprite.spritecollide(player,meteor_list,
True )

    screen.fill(WHITE)

    all_sprites_list.draw(screen)

    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```

Agregamos objetos de tipo Meteor y Player.

Cuando llamado al método update desde un grupo de objetos que pertenecen a distintas clases cada objeto ejecuta su update correspondiente.

13.- Disparando con el mouse

Para este capítulo tenemos que agregar el siguiente laser.

laser.png

Vamos a escribir el siguiente código:

```
import pygame, random

class Laser(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("laser.png").convert()
        self.image.set_colorkey(WHITE)
        self.rect = self.image.get_rect()

    def update(self):
        self.rect.y -= 5

class Meteor(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("meteor.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()

class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("player.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()

    def update(self):
        mouse_pos = pygame.mouse.get_pos()
        self.rect.x = mouse_pos[0]
        self.rect.y = 510

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
pygame.init()
screen = pygame.display.set_mode([900,600])
clock = pygame.time.Clock()
done = False
score = 0
```

Definimos la clase Laser y el método update que la coordenada y del láser restamos 5, esto hará que el láser vaya hacia arriba.

```
all_sprite_list = pygame.sprite.Group()
meteor_list = pygame.sprite.Group()
laser_list = pygame.sprite.Group()

for i in range(50):
    meteor = Meteor()
    meteor.rect.x = random.randrange(880)
    meteor.rect.y = random.randrange(450)
    meteor_list.add(meteor)
    all_sprite_list.add(meteor)

player = Player()
all_sprite_list.add(player)

while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
        if event.type == pygame.MOUSEBUTTONDOWN:
            laser = Laser()
            laser.rect.x = player.rect.x + 45
            laser.rect.y = player.rect.y - 20

            laser_list.add(laser)
            all_sprite_list.add(laser)

    all_sprite_list.update()

    for laser in laser_list:
        meteor_hit_list = pygame.sprite.spritecollide(laser, meteor_list,
True)
        for meteor in meteor_hit_list:
            all_sprite_list.remove(laser)
            laser_list.remove(laser)
            score += 1
            print(score)
        if laser.rect.y < -10:
            all_sprite_list.remove(laser)
            laser_list.remove(laser)

    screen.fill(WHITE)
    all_sprite_list.draw(screen)

    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```

Creamos un grupo para los lásers.

Si presionamos el botón del mouse creamos un objeto de tipo Laser.

Le damos las coordenadas iniciales.

Lo añadimos a las listas laser_list y all_sprite_list.

Llamamos a todos los update de todas las clases, para que cada objete seleccione la de su clase.

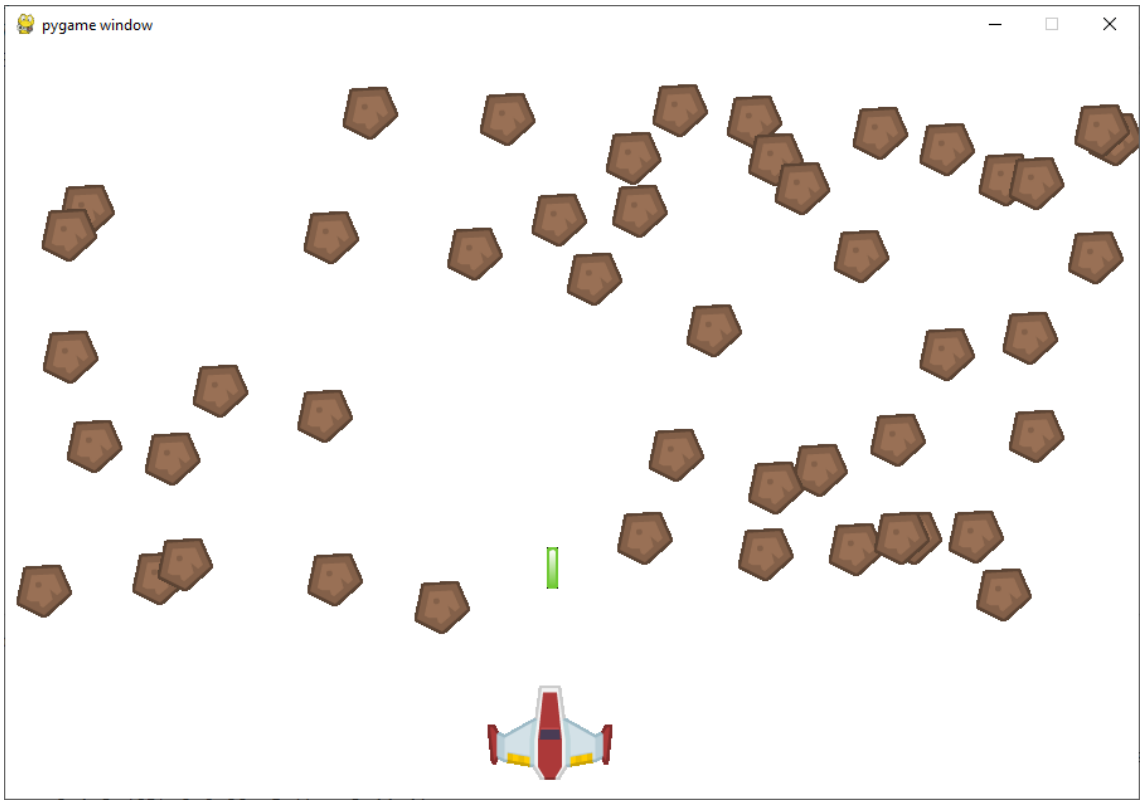
Con el for laser in laser_list: hacemos un recorrido por todos los laser que hay, si hay colisión laser con meteorito que el meteorito desaparezca.

De la lista de aciertos de meteoritos los eliminamos de las listas all_sprite_list y laser_list.

Los puntos se incrementan en 1.

Si el laser en la coordenada y desaparece por la parte superior de la ventana porque no ha tocado ningún meteorito, también lo eliminamos de las listas all_sprite_list y laser_list.

Este será el resultado:



14.- Dispara con el teclado:

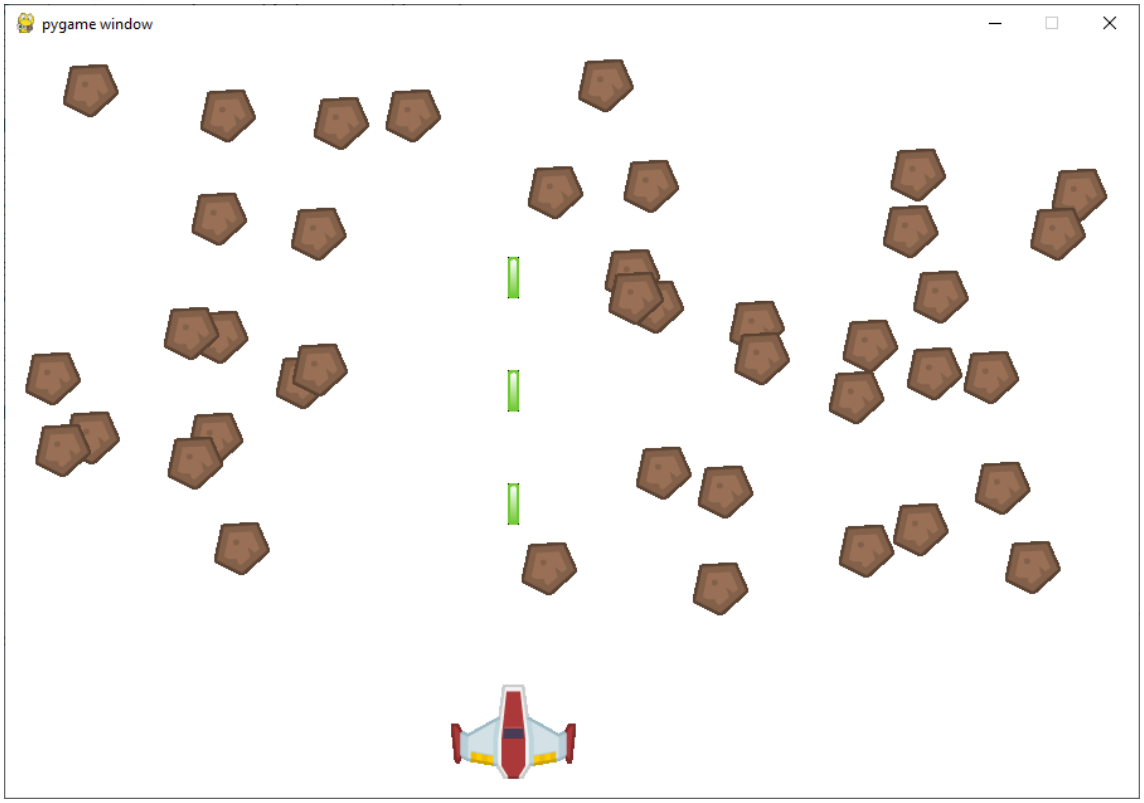
Vamos a modificar el proyecto anterior para mover la nave con flecha derecha y flecha izquierda y con la barra espaciadora disparar los misiles.

Esto es lo que hemos modificado del código:

```
import pygame, random

class Laser(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("laser.png").convert()
        self.image.set_colorkey(WHITE)
        self.rect = self.image.get_rect()

    def update(self):
        self.rect.y -= 5

class Meteor(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("meteor.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()

class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("player.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()
        self.speed_x = 0
        self.speed_y = 0

    def chargespeed(self, x):
        self.speed_x += x

    def update(self):
        self.rect.x += self.speed_x
        player.rect.y = 510

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
pygame.init()
screen = pygame.display.set_mode([900,600])
clock = pygame.time.Clock()
done = False
score = 0
```

```
all_sprite_list = pygame.sprite.Group()
meteor_list = pygame.sprite.Group()
laser_list = pygame.sprite.Group()

for i in range(50):
    meteor = Meteor()
    meteor.rect.x = random.randrange(880)
    meteor.rect.y = random.randrange(450)
    meteor_list.add(meteor)
    all_sprite_list.add(meteor)

player = Player()
all_sprite_list.add(player)

while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                player.chargespeed(-3)
            if event.key == pygame.K_RIGHT:
                player.chargespeed(3)

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT:
                player.chargespeed(3)
            if event.key == pygame.K_RIGHT:
                player.chargespeed(-3)
            if event.key == pygame.K_SPACE:
                laser = Laser()
                laser.rect.x = player.rect.x + 45
                laser.rect.y = player.rect.y - 20

                laser_list.add(laser)
                all_sprite_list.add(laser)

    all_sprite_list.update()

    for laser in laser_list:
        meteor_hit_list = pygame.sprite.spritecollide(laser, meteor_list,
True)
        for meteor in meteor_hit_list:
            all_sprite_list.remove(laser)
            laser_list.remove(laser)
            score += 1
            print(score)
        if laser.rect.y < -10:
            all_sprite_list.remove(laser)
```

```
                laser_list.remove(laser)

        screen.fill(WHITE)
        all_sprite_list.draw(screen)

        pygame.display.flip()
        clock.tick(60)

pygame.quit()
```



15.- Agregando sonido.

Para este capítulo necesitamos agregar un archivo de sonido:

laser5.ogg

Hemos añadido las siguiente líneas:

```
import pygame, random

class Laser(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("laser.png").convert()
        self.image.set_colorkey(WHITE)
        self.rect = self.image.get_rect()

    def update(self):
        self.rect.y -= 5

class Meteor(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("meteor.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()

class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("player.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()
        self.speed_x = 0
        self.speed_y = 0

    def chargespeed(self, x):
        self.speed_x += x

    def update(self):
        self.rect.x += self.speed_x
        player.rect.y = 510

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
```

```
pygame.init()
screen = pygame.display.set_mode([900,600])
clock = pygame.time.Clock()
done = False
score = 0

all_sprite_list = pygame.sprite.Group()
meteor_list = pygame.sprite.Group()
laser_list = pygame.sprite.Group()

for i in range(50):
    meteor = Meteor()
    meteor.rect.x = random.randrange(880)
    meteor.rect.y = random.randrange(450)
    meteor_list.add(meteor)
    all_sprite_list.add(meteor)

sound = pygame.mixer.Sound("laser5.ogg")

player = Player()
all_sprite_list.add(player)

while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                player.chargespeed(-3)
            if event.key == pygame.K_RIGHT:
                player.chargespeed(3)

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT:
                player.chargespeed(3)
            if event.key == pygame.K_RIGHT:
                player.chargespeed(-3)
            if event.key == pygame.K_SPACE:
                laser = Laser()
                laser.rect.x = player.rect.x + 45
                laser.rect.y = player.rect.y - 20

                laser_list.add(laser)
                all_sprite_list.add(laser)
                sound.play()

    all_sprite_list.update()

    for laser in laser_list:
```

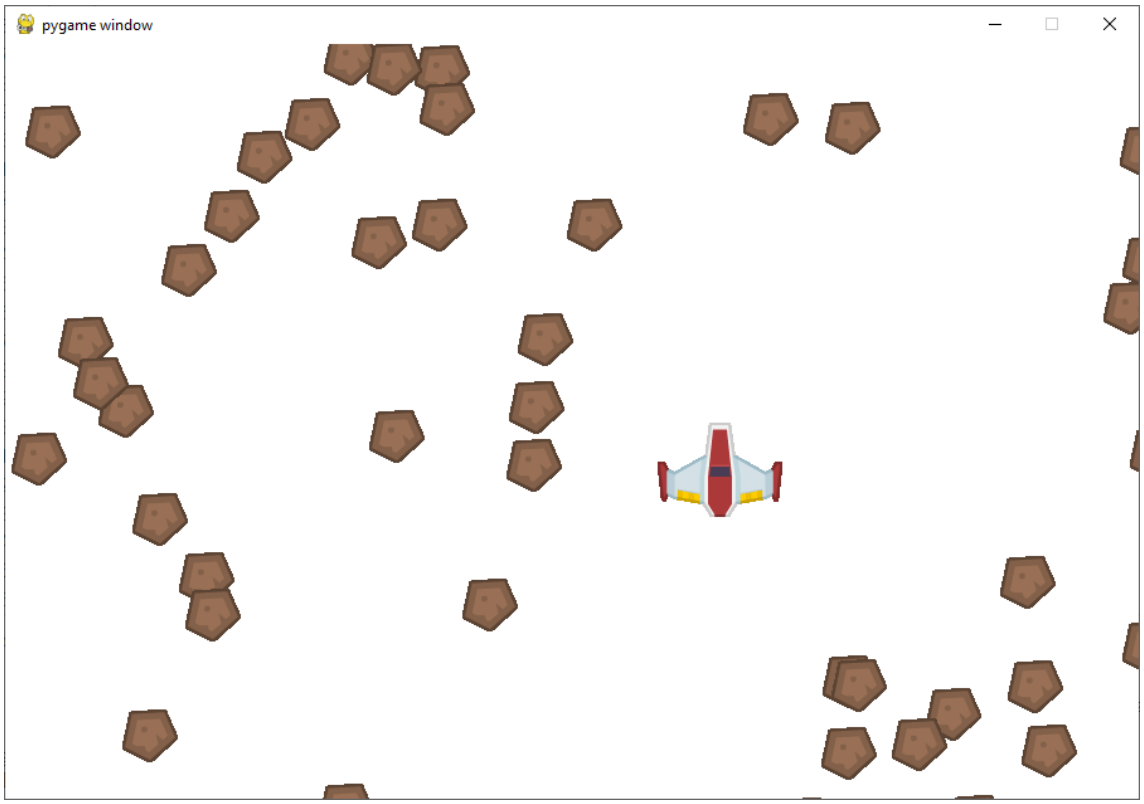
Cargar el sonido.

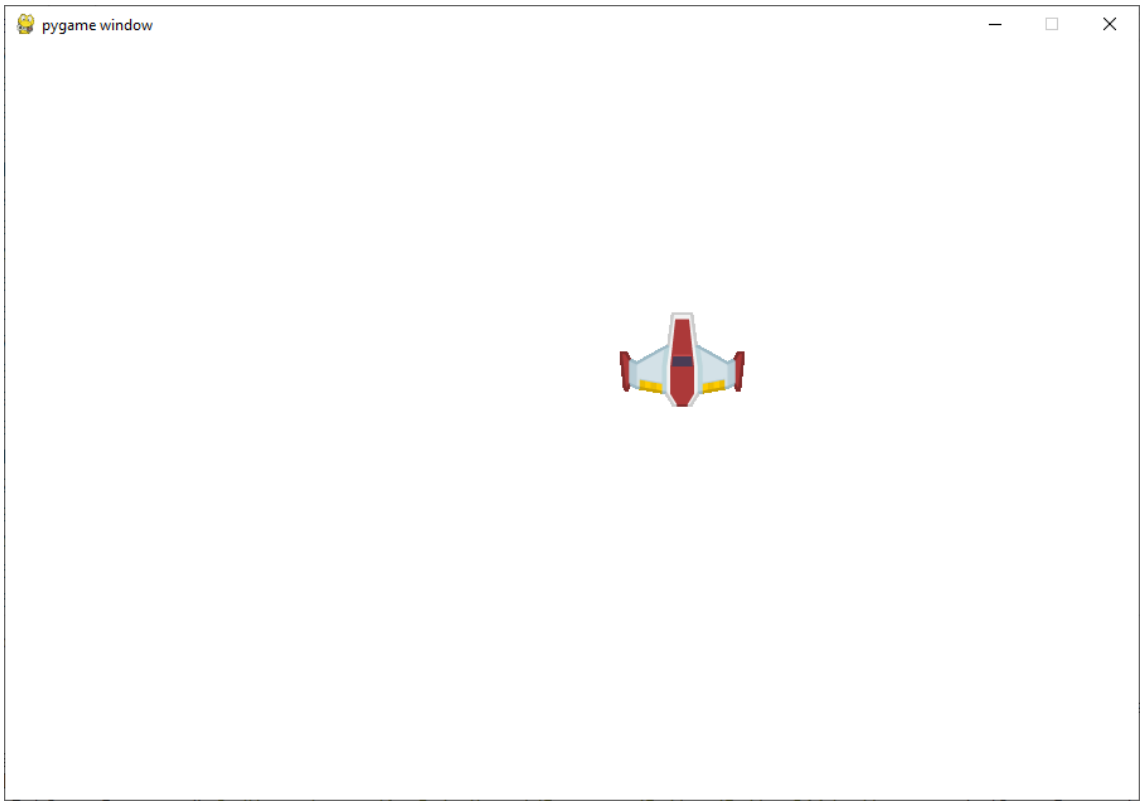
Reproducir el sonido.

```
        meteor_hit_list = pygame.sprite.spritecollide(laser, meteor_list,
True)
        for meteor in meteor_hit_list:
            all_sprite_list.remove(laser)
            laser_list.remove(laser)
            score += 1
            print(score)
        if laser.rect.y < -10:
            all_sprite_list.remove(laser)
            laser_list.remove(laser)

    screen.fill(WHITE)
    all_sprite_list.draw(screen)

    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```

16.- Clase Juego

Este capítulo nos va a servir para hacer nuestro GAME OVER.





Cuando hemos recogido todos los objetos no pasa nada.

Adjunto código de este capítulo:

```
import pygame, random
SCREEN_WIDTH = 900
SCREEN_HEIGHT = 600
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
```

Clase Meteor

```
class Meteor(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("meteor.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()

    def update(self):
        self.rect.y += 1

        if self.rect.y > SCREEN_HEIGHT:
            self.rect.y = -10
            self.rect.x = random.randrange(SCREEN_WIDTH)
```

Clase Player

```
class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("player.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()

    def update(self):
        mouse_pos = pygame.mouse.get_pos()
        self.rect.x = mouse_pos[0]
        self.rect.y = mouse_pos[1]
```

Clase Game

```
class Game(object):
    def __init__(self):
        self.score = 0
        self.meteor_list = pygame.sprite.Group()
        self.all_sprites_list = pygame.sprite.Group()

        for i in range(50):
            meteor = Meteor()
            meteor.rect.x = random.randrange(900)
            meteor.rect.y = random.randrange(600)
            self.meteor_list.add(meteor)
            self.all_sprites_list.add(meteor)

        self.player = Player()
        self.all_sprites_list.add(self.player)
```

```
def process_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            return True
    return False
```

Continua la Clase Game

```
    def run_logic(self):
        self.all_sprites_list.update()
        meteor_hit_list =
pygame.sprite.spritecollide(self.player,self.meteor_list, True )

        for meteor in meteor_hit_list:
            self.score += 1
            print(self.score)
```

```
def display_frame(self, screen):
    screen.fill(WHITE)
    self.all_sprites_list.draw(screen)
    pygame.display.flip()
```

```
def main():
    pygame.init()
    screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
    done = False
    clock = pygame.time.Clock()

    game = Game()

    while not done:
        done = game.process_events()
        game.run_logic()
        game.display_frame(screen)
        clock.tick(60)

    pygame.quit()

if __name__ == "__main__":
    main()
```

Creamos un objeto de la clase Game

17.- Implementando Game Over

En este capítulo aprenderemos dos cosas, la primer a realizar un GAME OVER y cómo podemos poner textos en nuestros juegos.

Vamos a agregar el código que falta:

```
import pygame, random
SCREEN_WIDTH = 900
SCREEN_HEIGHT = 600
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)

class Meteor(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("meteor.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()

    def update(self):
        self.rect.y += 1

        if self.rect.y > SCREEN_HEIGHT:
            self.rect.y = -10
            self.rect.x = random.randrange(SCREEN_WIDTH)

class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("player.png").convert()
        self.image.set_colorkey(BLACK)
        self.rect = self.image.get_rect()

    def update(self):
        mouse_pos = pygame.mouse.get_pos()
        self.rect.x = mouse_pos[0]
        self.rect.y = mouse_pos[1]

class Game(object):
    def __init__(self):
        self.game_over = False
        self.score = 0
        self.meteor_list = pygame.sprite.Group()
        self.all_sprites_list = pygame.sprite.Group()

        for i in range(50):
            meteor = Meteor()
            meteor.rect.x = random.randrange(900)
            meteor.rect.y = random.randrange(600)
```

Definimos una variable de tipo bolean, para controlar cuando se termina el juego.

```
            self.meteor_list.add(meteor)
            self.all_sprites_list.add(meteor)

        self.player = Player()
        self.all_sprites_list.add(self.player)

    def process_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                return True
            if event.type == pygame.MOUSEBUTTONDOWN:
                if self.game_over:
                    self.__init__()

        return False

    def run_logic(self):
        if not self.game_over:
            self.all_sprites_list.update()
            meteor_hit_list =
pygame.sprite.spritecollide(self.player,self.meteor_list, True )

            for meteor in meteor_hit_list:
                self.score += 1
                print(self.score)

            if len(self.meteor_list) == 0:
                self.game_over = True

    def display_frame(self, screen):
        screen.fill(WHITE)

        if self.game_over:
            font = pygame.font.SysFont("serif", 25) # Fuente
            text = font.render("Game Over, Click to Continue", True,
BLACK) # Texto
            center_x = (SCREEN_WIDTH // 2) - (text.get_width() // 2) #
Coordenadas x
            center_y = (SCREEN_HEIGHT // 2) - (text.get_height() // 2) #
Coordenadas y
            screen.blit(text, [center_x, center_y]) # Ponerlo en pantalla

        if not self.game_over:
            self.all_sprites:list.draw(screen)

        self.all_sprites_list.draw(screen)
        pygame.display.flip()

def main():
```

Cuando hacemos clic con el botón del ratón se reinicia el juego.

Si no has llegado al final del juego llama al método update() desde el grupo all_sprites_list.update().

Comprueba si hay colisiones, por colisión incrementa a uno los puntos (score) y los muestra por consola.

Cuando el grupo meteor_list este a 0, significa que has terminado el juego.

Para dibujar el texto “Game Over” cuando finaliza e juego.

```
    pygame.init()
    screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
    done = False
    clock = pygame.time.Clock()

    game = Game()

    while not done:
        done = game.process_events()
        game.run_logic()
        game.display_frame(screen)
        clock.tick(60)

    pygame.quit()

if __name__ == "__main__":
    main()
```