Para seguir los videotutoriales en YouTube en el canal de Manuel González llamado Programar es como un juego.

PYGAME EN PYTHON

¿CÓMO CREAR JUEGOS?

PERE MANEL VERDUGO ZAMORA

pereverdugo@gmail.com

Contenido

1.- Instalando pygame

Modo texto → Consola del sistema
Consola del IDLE

Modo gráfico → Interface gráfica (GUI)

Empezamos un nuevo tutorial en que vamos a intentar hacer nuestros primeros juegos en modo gráfico.

Utilizando una interface gráfica, una ventana con la que poder interactuar con el juego.

Hasta ahora los juegos que hemos hecho han sido en modo texto usando la consola del sistema o la consola del IDLE.

Vamos a ver como podemos hacer juegos que tengan gráficos, imágenes, que podemos utilizar el ratón, que podamos utilizar todas las teclas del teclado en definitiva un juego tal y como lo entendemos hoy con componentes gráficos.

Para ello necesitaremos de una librería de Python que nos permita acceder y manejar los recursos del hardware del ordenador: Tarjeta gráfica, tarjeta de sonido, ratón, etc.

De esta forma podremos crear una ventana donde aparezcan los gráficos, las imágenes, los sonidos y desde la cual tengamos control sobre los eventos del juego.

Librerías orientadas a aplicaciones de escritorio:

Tkinter, wxPython, PyQt, Pyside, PyGTK, …

Librerías orientadas a juegos:

Pygame, Pyglet, Arcade, …

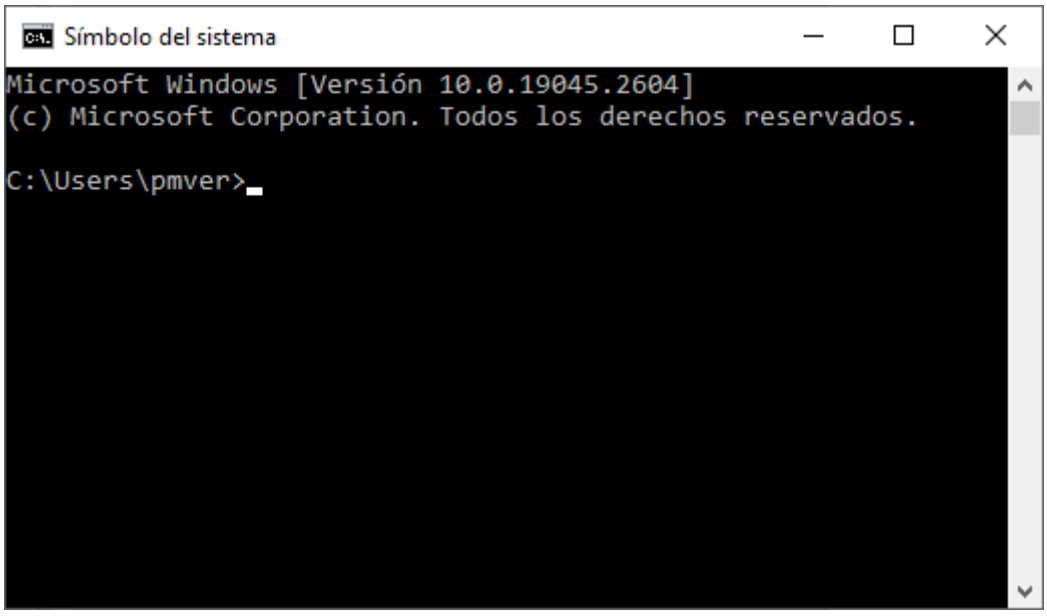
Nosotros vamos a utilizar Pygame.

La única que viene instalada por defecto es Tkinter, las demás no vienen por defecto en la instalación de Python y que hay que instalar.

La instalación de estos paquetes va a ser muy sencilla, a partir de la versión de Python 3.4 viene por defecto un sistema para la gestión de paquetes Pip que hace muy sencilla la instalación de cualquier paquete.

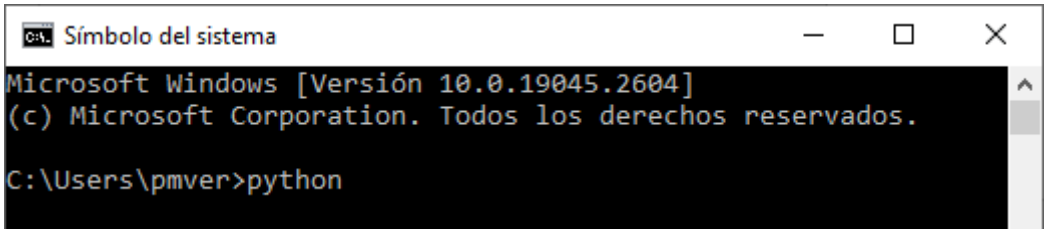
Esta aplicación se conecta a Python Package index (PyPI), que es un repositorio donde están guardadas la mayor parte de paquetes y aplicaciones e terceros como dice la Wikipedia del lenguaje de programación Python.

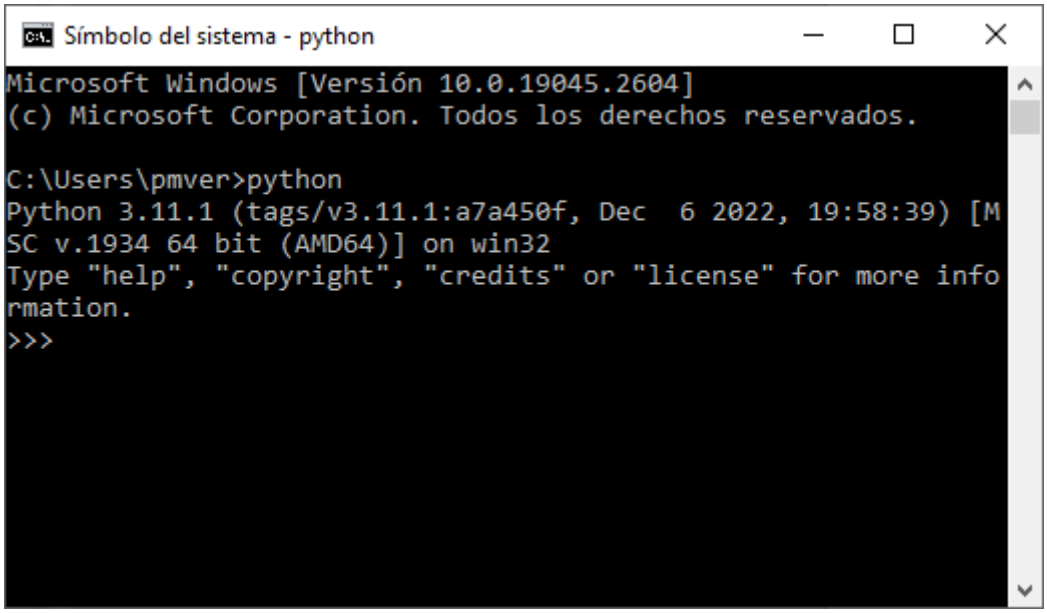
Vamos a abrir Cmd del sistema:



Lo primero que vamos a comprobar si tenemos definidas las variables del entorno del sistema para Python.

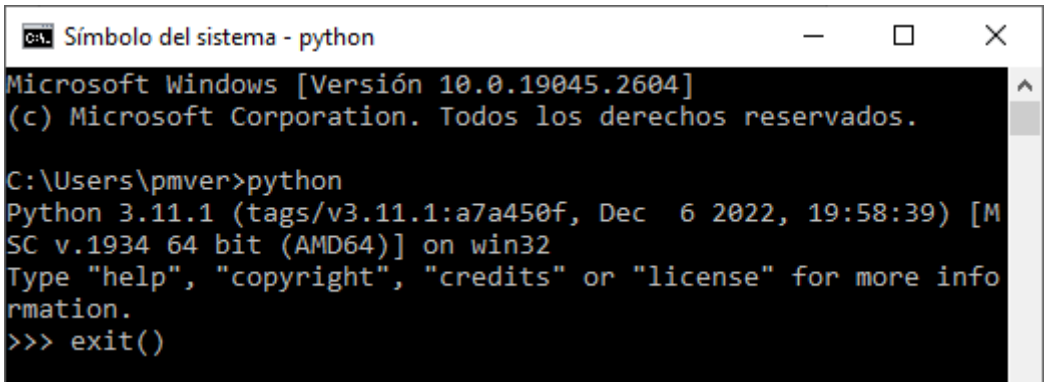
Vamos a escribir python.

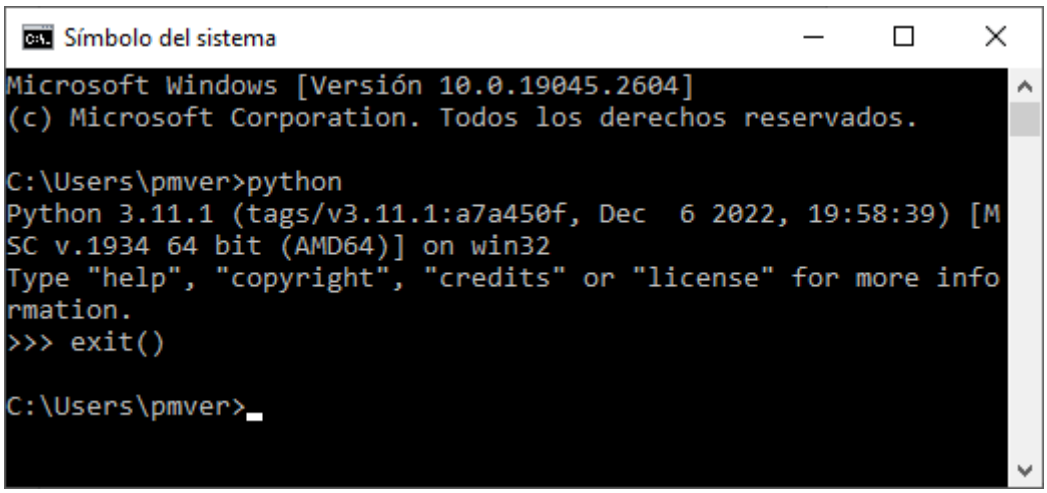




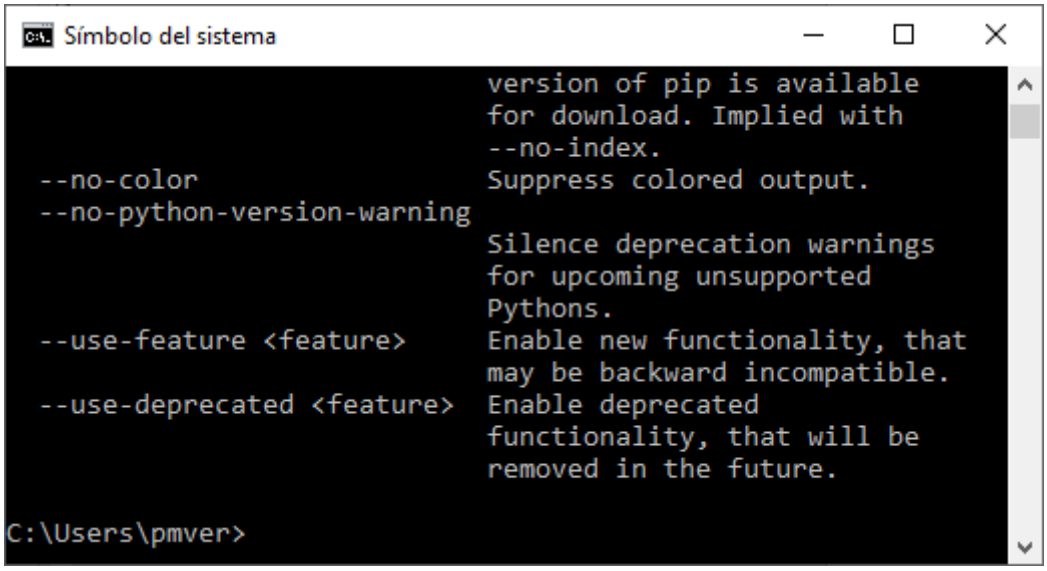
Tenemos que entrar en el intérprete de Python.

Para salir escribiremos exit()





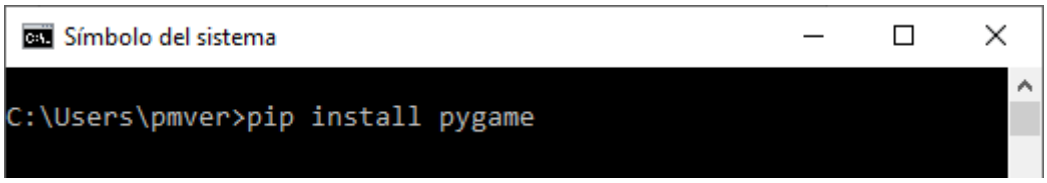
Vamos a ver si tenemos definidas las variables para el entorno pip.

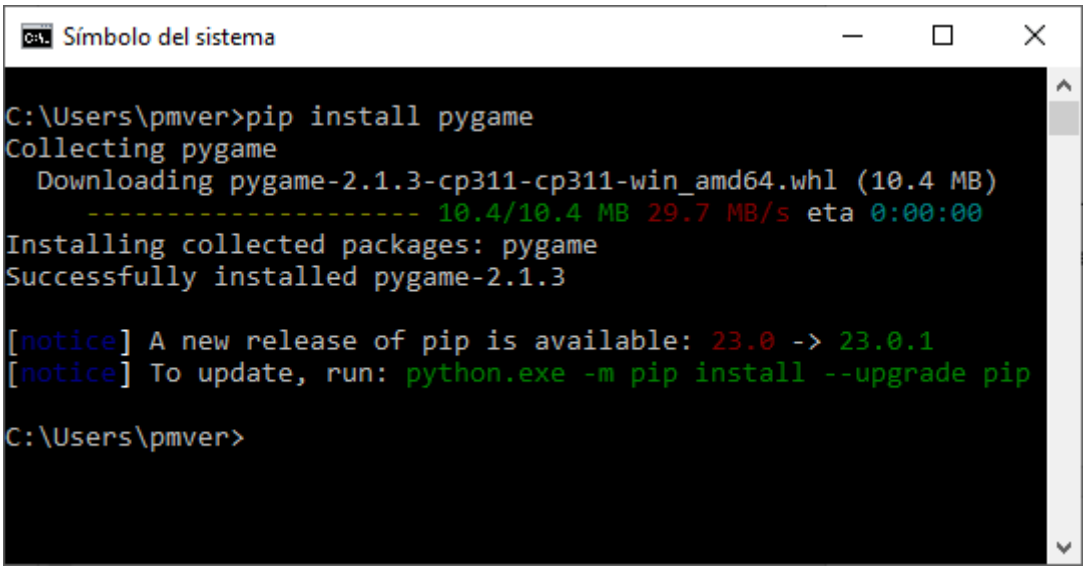


Muestra una información general de este comando.

Si escribimos **pip list** nos mostrará los paquetes que tenemos instalados.

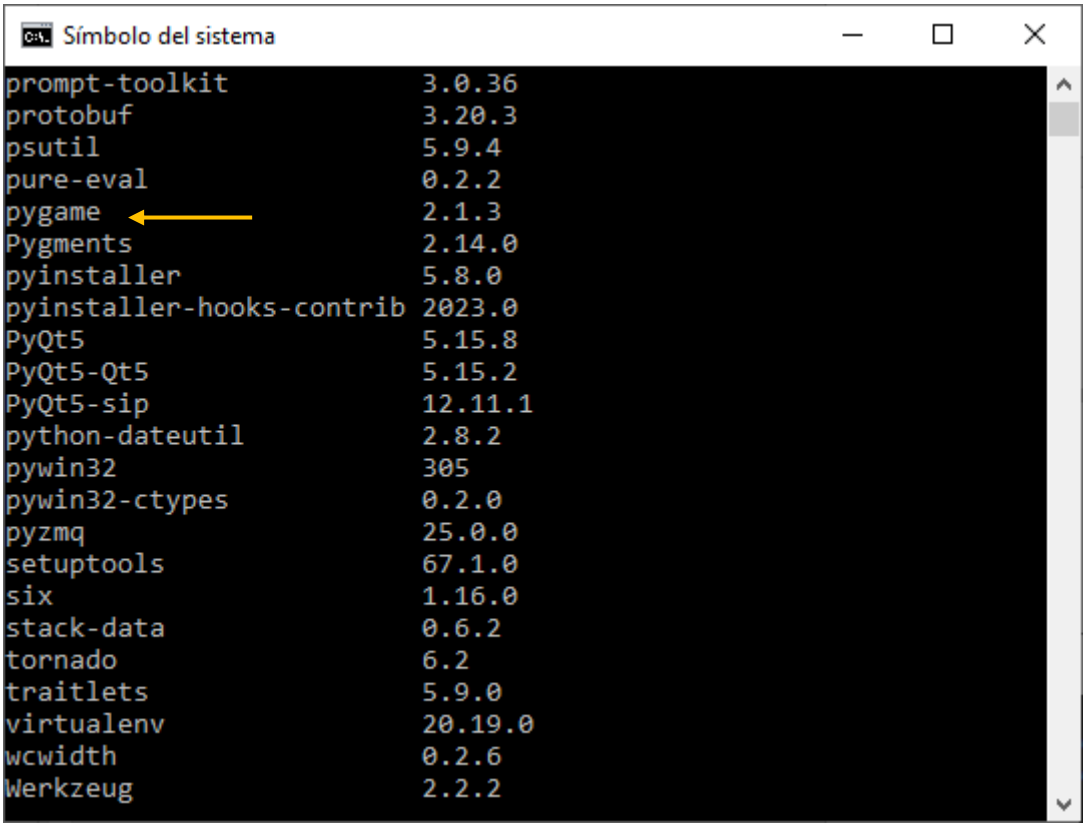
Para instalar vamos a escribir pip install pygame.





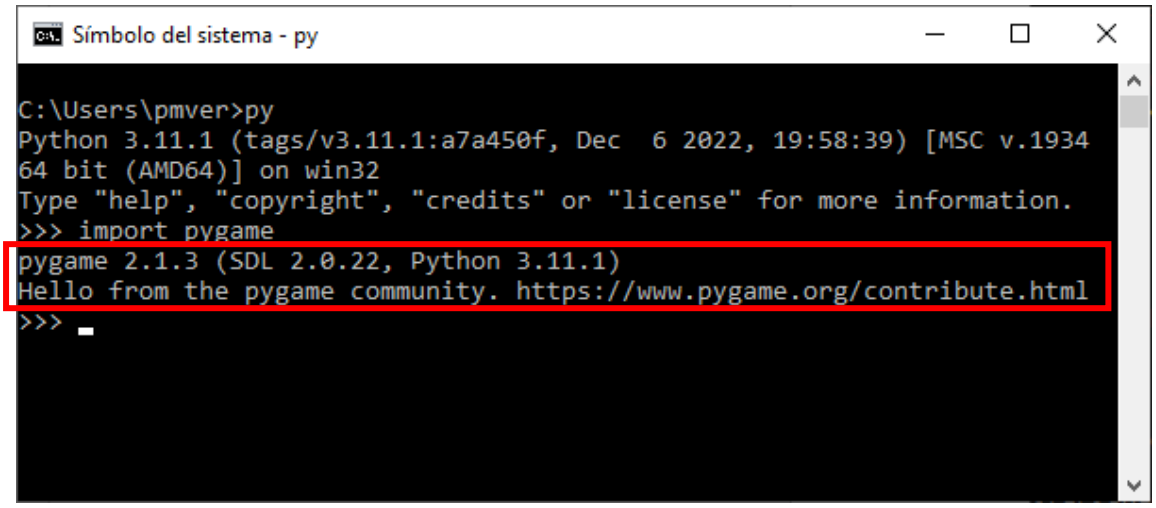
Ya lo ha instalado.

Vamos a realizar un pip list para ver si ya lo tenemos.



Ya lo tenemos.

Vamos al intérprete de Python y escribimos import pygame.

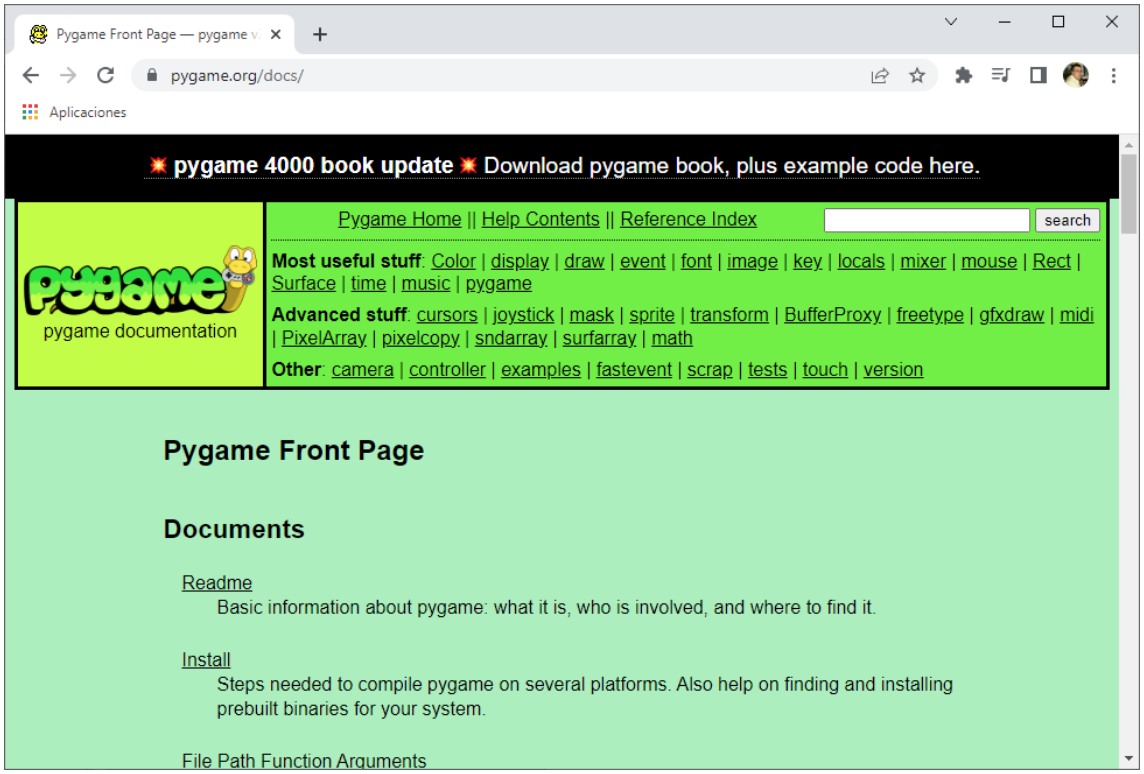


Hemos instalado correctamente esta librería y vamos a poder utilizarla.

2.- Primera ventana en pygame

Antes de empezar vamos a ver algunos recursos para los que les interese ir más deprisa.

https://www.pygame.org/docs/



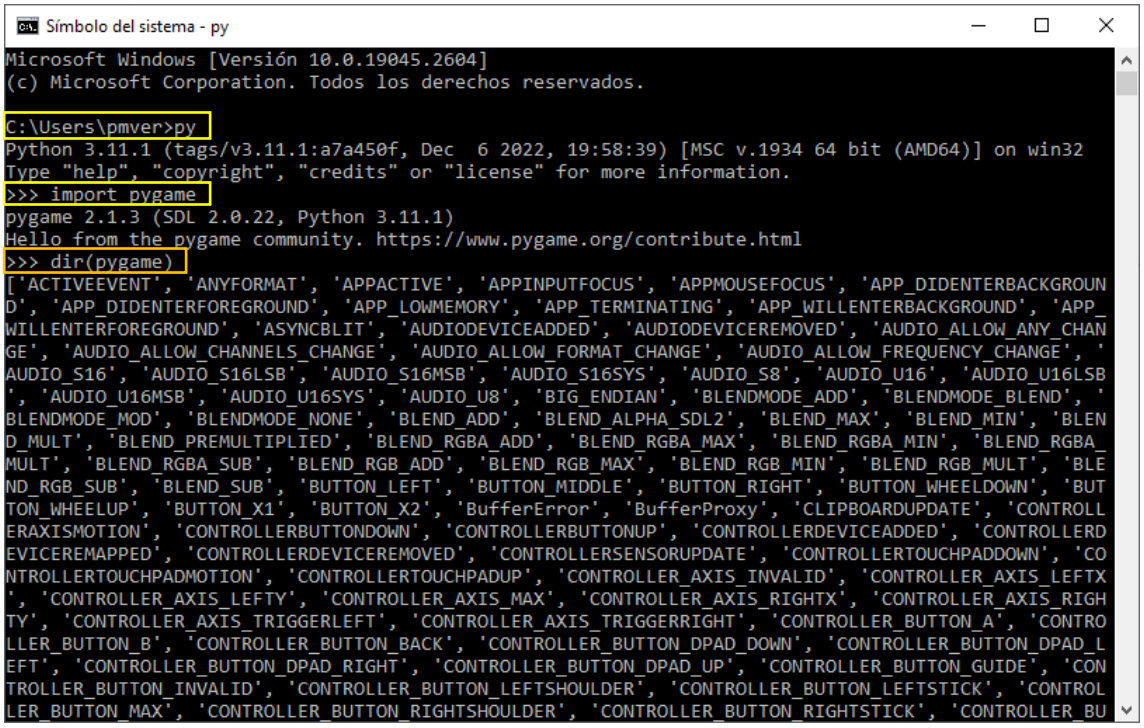
Hay muchos tutoriales, pero están en inglés.

http://programarcadegames.com/

https://inventwithpython.com/es/

Vamos a empezar, una vez hallamos creado una carpeta creamos el archivo ejemplo1.py

Desde la cmd de Windows:



Podemos ver todo los elementos que contiene y los módulos que a su vez contiene esta librería.

Vamos a hacer nuestro primer programa que consiste en abrir una ventana.

```
1   import pygame
2
3   pygame.init()
4
5   ventana = pygame.display.set_mode((800, 600))
6
7   pygame.quit()
```

De esta manera se inicializa los módulos.

Al ejecutar la ventana a aparecido pero de inmediato se ha cerrado, para mantener la ventana.

```
1   import pygame
2
3   pygame.init()
4
5   ventana = pygame.display.set_mode((800, 600))
6
7   pygame.time.delay(3000)
8
9   pygame.quit()
```

La ventana se muestra durante 3 segundos.

```
1   import pygame
2
3   pygame.init()
4
5   ventana = pygame.display.set_mode((800, 600))
6
7   ventana.fill((255, 255, 255))
8
9   pygame.display.update()
10
11  pygame.time.delay(3000)
12
13  pygame.quit()
```

Para cambiar el color de la pantalla, pero para que tenga efecto hay que actualizar, tal como se muestra en la línea 9.

3.- Eventos de pygame

En el capítulo anterior ya vimos como creamos una primera ventana, vamos a ir recordando lo del capítulo anterior introduciendo algunas optimizaciones para que nuestro código sea más legible.

```
1    import pygame
```

Lo primero que hacíamos era importar la librería pyame.

```
3    pygame.init()
```

Posteriormente inicializábamos con la función init los módulos de esta librería.

Ya estamos en condiciones de crear nuestra primera ventana.

```
5    ventana = pygame.display.set_mode((800, 600))
```

Creamos la ventana y a la función set_mode le tenemos que dar las dimensiones en este caso 800 x 600.

```
5    ANCHO = 800
6    ALTO = 600
7    ventana = pygame.display.set_mode((ANCHO, ALTO))
```

Es recomendable definir dos variables que las pondremos en mayúsculas para tratarlas como constantes para guardar el ANCHO y ALTO.

```
9    ventana.fill((255, 255, 255))
```

Con la función fill la rellenamos de un color, le pasamos el valor con una tupla, pero estos valores los vamos a almacenar en contantes.

Otra forma de definir el color es creando constates:

```
8    BLANCO = (255,255,255)
9    NEGRO = (0, 0, 0)
10   ROJO = (255, 0, 0)
```

Para luego asignárselo al método fill.

```
14   ventana.fill(ROJO)
```

Si dejamos así el programa la ventana se cerrará inmediatamente, casi no vamos a poderla ver.

```
14    jugando = True
15
16    while jugando:
17        ventana.fill(ROJO)
```

Mediante es bucle controlaremos cuando queremos que se cierre la ventana.

```
16    while jugando:
17        ventana.fill(ROJO)
18        pygame.display.update()
```

Tenemos que actualizar la ventana.

```
16    while jugando:
17        ventana.fill(ROJO)
18        pygame.display.update()
19
20    pygame.quit()
```

Fuera del bucle para que se cierre la ventana.

Vamos a ejecutar:

Se ejecuta la ventana pero al intentar cerrarla se reproduce un error.

Código completo:

```
import pygame

pygame.init()

ANCHO = 800
ALTO = 600
BLANCO = (255,255,255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)

ventana = pygame.display.set_mode((ANCHO, ALTO))

jugando = True

while jugando:
    ventana.fill(ROJO)
    pygame.display.update()

pygame.quit()
```

solo hemos creado la ventana pero no podemos ralizar ninguna acción sobre ella, porque no lo hemos codificado.

```
16    while jugando:
17        for event in pygame.event.get():
18            print(event)
19
20        ventana.fill(ROJO)
21        pygame.display.update()
```

Vamos a ejecutar:

```
<Event(1024-MouseMotion {'pos': (660, 4), 'rel': (9, 4), 'butto
s': (0, 0, 0), 'touch': False, 'window': None})>
<Event(1024-MouseMotion {'pos': (669, 6), 'rel': (9, 2), 'butto
s': (0, 0, 0), 'touch': False, 'window': None})>
<Event(1024-MouseMotion {'pos': (683, 10), 'rel': (14, 4), 'but
ons': (0, 0, 0), 'touch': False, 'window': None})>
<Event(1024-MouseMotion {'pos': (703, 16), 'rel': (20, 6), 'but
ons': (0, 0, 0), 'touch': False, 'window': None})>
```

Se nos muestra los eventos que recoge la ventana, en este caso como movemos el ratón por la ventana.

Puede recoger cualquier botón que presionemos del ratón o cualquier tecla del teclado.

```
14    jugando = True
15
16    while jugando:
17        for event in pygame.event.get():
18            if event.type == pygame.QUIT:
19                jugando = False
20            if event.type == pygame.KEYDOWN:
21                if event.key == pygame.K_q:
22                    jugando = False
23
24        ventana.fill(ROJO)
25        pygame.display.update()
```

En la línea 14 definimos una variable con el valor True.

En la línea 16 mientras será True no saldremos del bucle.

En la línea 17 en la variable evento recogemos un evento, este puede ser mover el ratón, pulsar el botón del ratón o utilizar alguna tecla del ratón.

En la línea 18 estamos comparando si el evento es que hemos presionado la x de la ventana para cerrarla.

En la línea 19 si es así la varible pasa a valer False, como es la variable que compara el bucle while, el bucle finaliza.

En la línea 20 queremos comparar si el evento es referente a presionar una tecla.

En la línea 21 comparamos si hemos presionado la tecla 'q' si es así en la línea 22 volvemos a asignar a la variable False para que finalice el bucle while.

Ya podemos cerrar dándole a la x.

También presionando la tecla 'q'.

Te adjunto el código completo:

```
1    import pygame
2
3    pygame.init()
4
```

```
5   ANCHO = 800
6   ALTO = 600
7   BLANCO = (255,255,255)
8   NEGRO = (0, 0, 0)
9   ROJO = (255, 0, 0)
10  ventana = pygame.display.set_mode((ANCHO, ALTO))
11
12  jugando = True
13
14  while jugando:
15      for event in pygame.event.get():
16          if event.type == pygame.QUIT:
17              jugando = False
18          if event.type == pygame.KEYDOWN:
19              if event.key == pygame.K_q:
20                  jugando = False
21
22      ventana.fill(ROJO)
23      pygame.display.update()
24
25  pygame.quit()
```

Te propongo que practiques con los eventos o por mediación de una tecla.

4.- Pantalla completa en pygame

Este es el código que hemos visto hasta ahora:

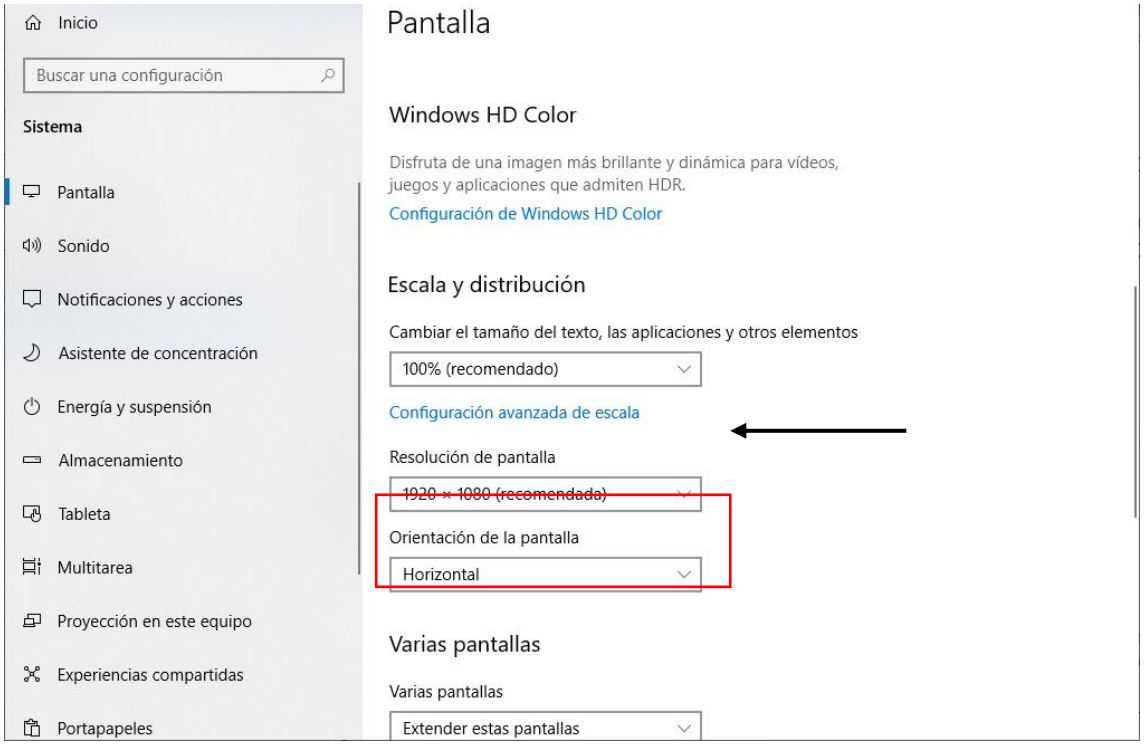
```
1   import pygame
2
3   # Inicializar
4   pygame.init()
5
6   # Medidas
7   ANCHO = 800
8   ALTO = 600
9
10  # Colores
11  BLANCO = (255,255,255)
12  NEGRO = (0, 0, 0)
13  ROJO = (255, 0, 0)
14  VERDE = (0, 255, 0)
15  AZUL = (0, 0, 255)
16
17  # Ventana
18  ventana = pygame.display.set mode((ANCHO, ALTO))
19
20  # Bucle principal
21  jugando = True
22  while jugando:
23      for event in pygame.event.get():
24          if event.type == pygame.QUIT:
25              jugando = False
26          if event.type == pygame.KEYDOWN:
27              if event.key == pygame.K_ESCAPE:
28                  jugando = False
29      # Dibujos
30      ventana.fill(NEGRO)
31
32      # Actualizar
33      pygame.display.update()
34
```

```
35    # Salir
36    pygame.quit()
```

En los eventos solamente hemos definido “Cerrara la ventana” con la x, y también como salir del bucle presionando una tecla, en este caso la tecla ‘q’, que hemos cambiado por la tecla Escape.
Esto nos puede ser muy útil si queremos configurar nuestros juegos a pantalla completa.

```
18    ventana = pygame.display.set_mode((ANCHO, ALTO),
19                                      pygame.FULLSCREEN)
```

Puede ir todo en la misma línea.
Antes de ejecutar, tenemos que saber las medidas de resolución de tu ordenador.
Botón derecho del ratón sobre el escritorio y seleccionaremos configuración de pantalla.



Y vemos la resolución en mi caso es de 1920 x 1080.

Si en cambiar el tamaño del texto, es superior al 100% sería conveniente coger una resolución menor que en nuestro caso no lo es.

Para que nuestro juego pueda valer para otros ordenadores vamos a poner 1280 x 720.

```
6     # Medidas
7     ANCHO = 1280
8     ALTO = 720
```

Vamos a ejecutar.

La ventana ocupa toda la pantalla y para salir vamos a presionar la tecla Escape.

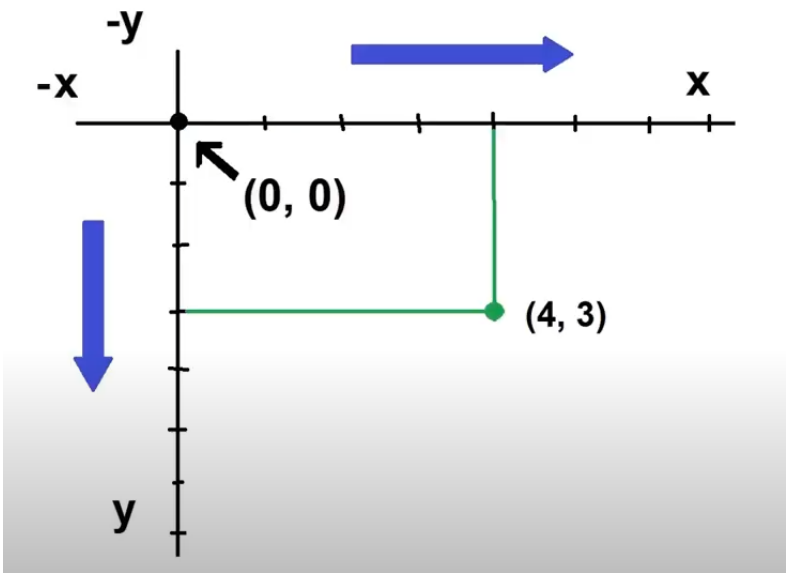
5.- Coordenadas de pygame

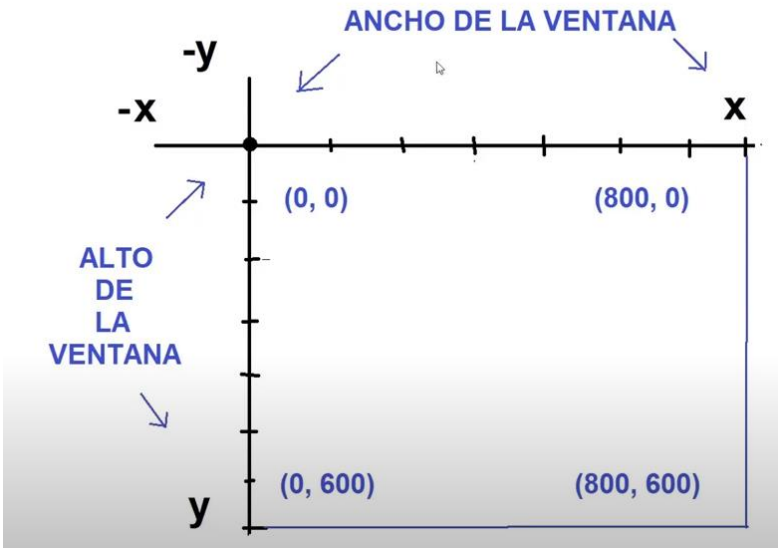
Seguimos viendo las opciones que nos ofrece la librería pygame.

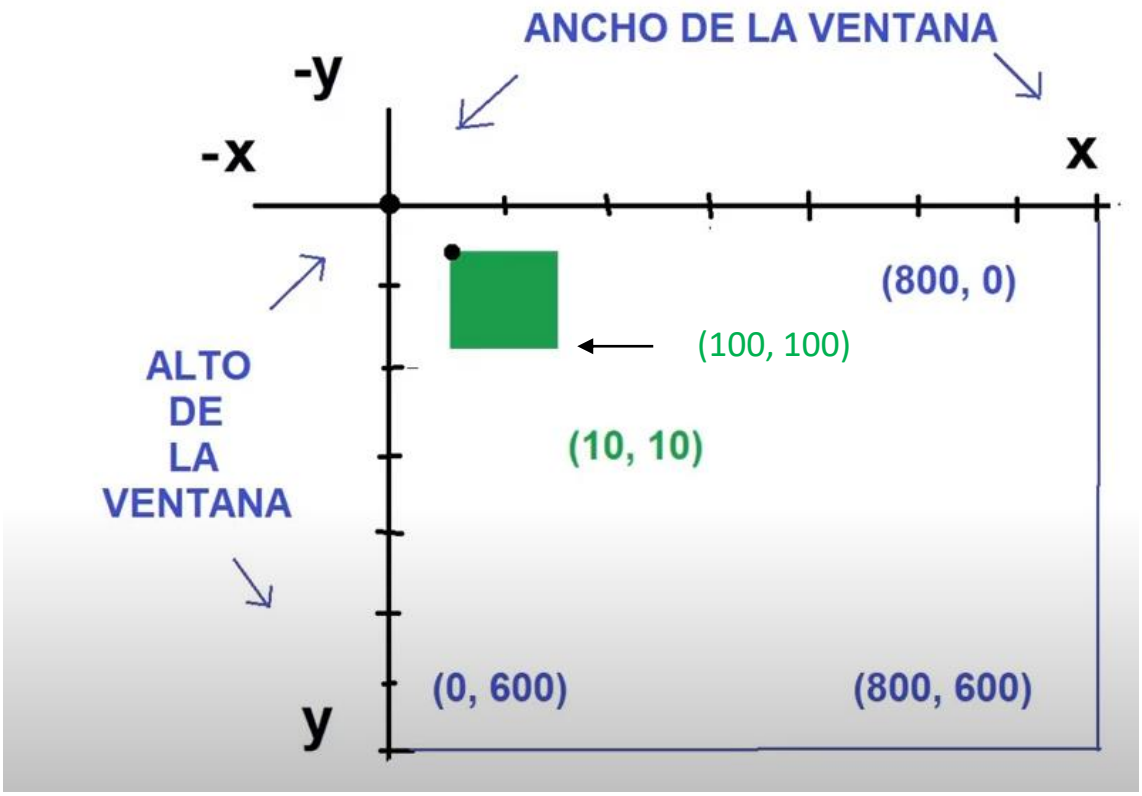
Hasta ahora hemos creado una ventana y simplemente lo que hemos hecho ha sido rellenarla de un color, vamos a ver como podemos dibujar figuras dentro de la ventana.

```
30      # Dibujos
31      ventana.fill(NEGRO)
32      pygame.draw.rect(ventana, VERDE, (10,10, 100, 100) )
```

Es este caso vamos a dibujar un cuadrado, esta función rect vale para cuadrados como rectángulos.







Este será el resultado:

```
29      # Dibujos
30      ventana.fill(NEGRO)
31      pygame.draw.rect(ventana, VERDE, (10,10, 100, 100) )
32
33      # Actualizar
34      pygame.display.update()
```

El proceso es el siguiente:

En la línea 30 le decimos que el fondo será de color negro, pero aun no se visualizará, se guardará en memoria.

En la línea 31 dibujamos un rectángulo, con los siguientes parámetros, lugar donde se dibuja (ventana), color podemos poner una tupla con los colores RGB o bien una constante con dichos valores, en nuestro caso seleccionamos VERDE, y por último las coordenadas que hemos detallado las imágenes anteriores.

Hasta que no lleguemos a la línea 34 con toda la información en memoria se mostrará la ventana con su cuadrado.

De otro modo si tuviéramos ciento de figuras o imágenes el juego no se vería adecuadamente, así cuando ya está todo en memoria mediante la función update() se plasma todo en la ventana.

Por otra parte es importante poner primero el fondo de la ventana, en este caso el rellenado con el color porque si ponemos primero el cuadrado y a continuación el fondo, el fondo taparía al cuadrado y se vería todo negro en este caso.

Por lo tanto primero rellenamos el fondo y luego el cuadrado.

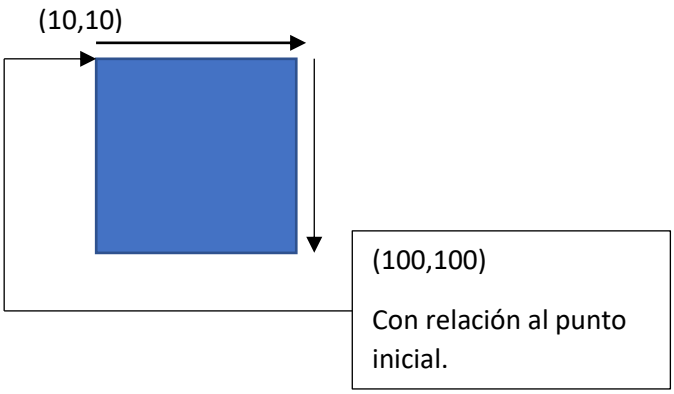
Te propongo este ejercicio:

6.- Dibujando con pygame

La solución:

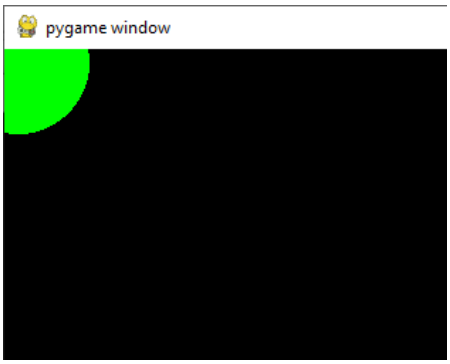
```
29      # Dibujos
30      ventana.fill(NEGRO)
31      pygame.draw.rect(ventana, VERDE, (10,10, 100, 100))
32      pygame.draw.rect(ventana, AZUL, (690,10, 100, 100))
33      pygame.draw.rect(ventana, ROJO, (10,490, 100, 100))
34      pygame.draw.rect(ventana, BLANCO, (690,490, 100, 100))
```

La primera coordenada es la parte superior izquierda del cuadrado y la segunda coordenada en con respecto a la primera.



En definitiva los parámetros son los dos primeros valores son las coordenadas x e y el tercer valor es el acho y el curto valor el alto.

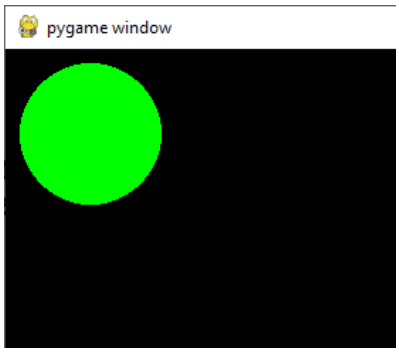
Vamos a eliminar los todos los cuadrados para dibujar un círculo.



Las coordenadas del circulo no funcionan igual que la del rectángulo, ya que en el circulo su coordenada se refiere al centro del mismo, si hacemos un radio mayor solo nos mostrará la parte inferior derecha.

```
29      # Dibujos
30      ventana.fill(NEGRO)
31      pygame.draw.circle(ventana, VERDE, (60,60),50)
```

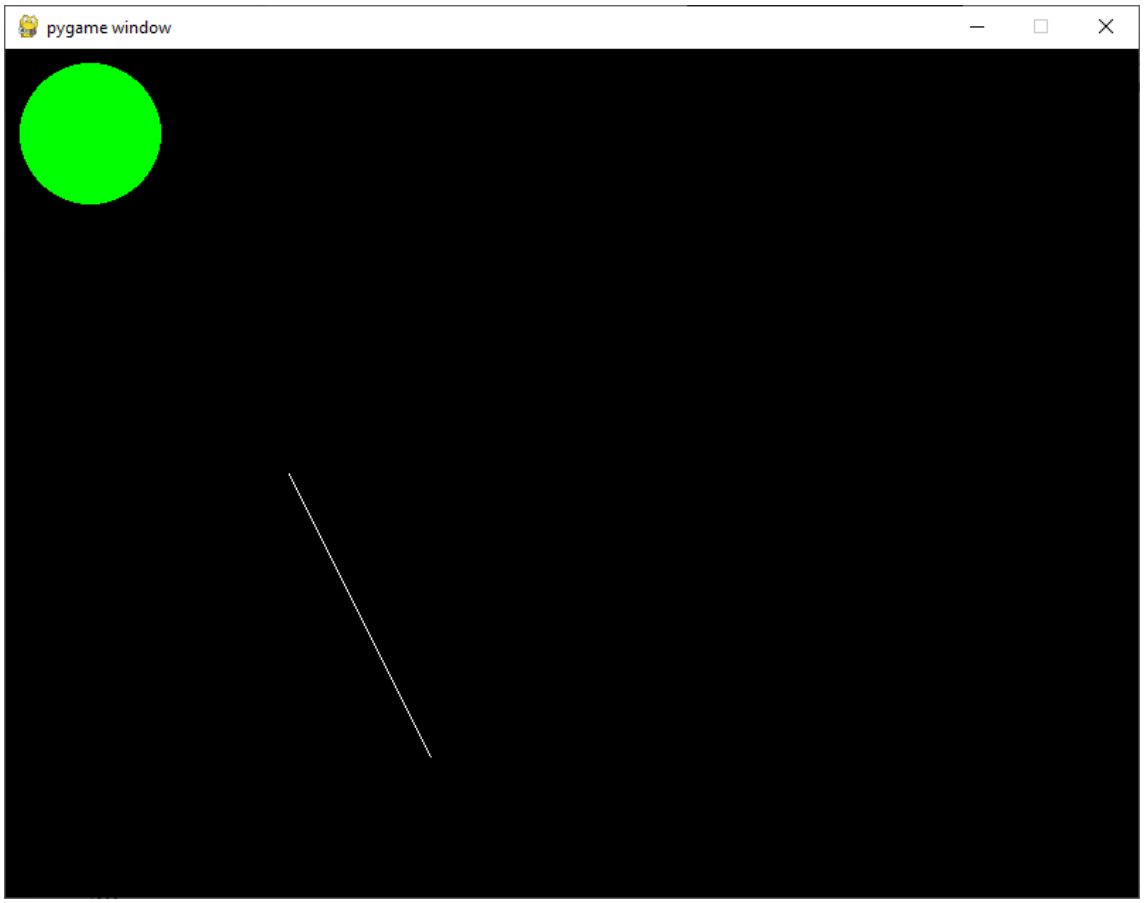
Si queremos que el circulo deje un espacio por arriba y a la izquierda de 10 pixeles tendremos que sumar a la coordenada la separación de la esquina que es 10 más el radio que es 50.



Vamos a dibujar una línea.

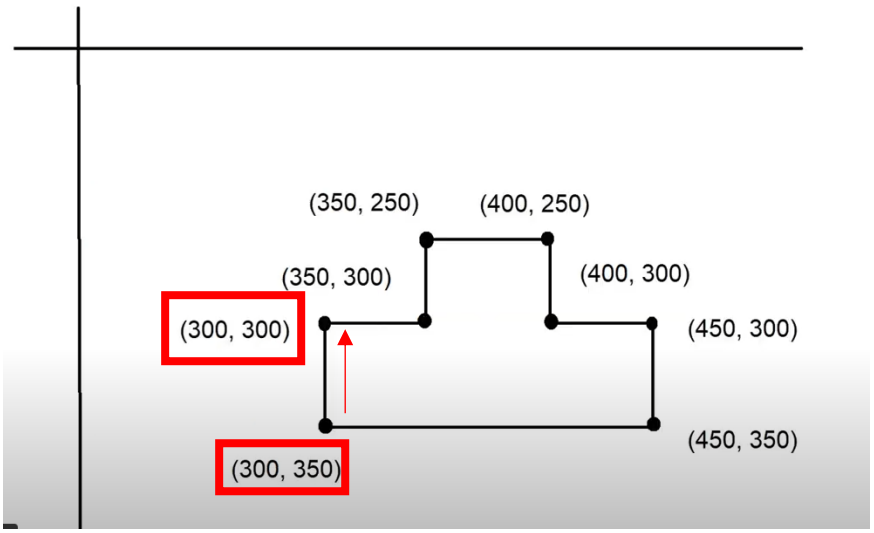
```
29      # Dibujos
30      ventana.fill(NEGRO)
31      pygame.draw.circle(ventana, VERDE, (60,60),50)
32      pygame.draw.line(ventana, BLANCO, (200, 300), (300, 500))
```

Sus parámetros son donde la dibujamos (ventana), el color, Coordenada_Inicial , Coordenada_final.



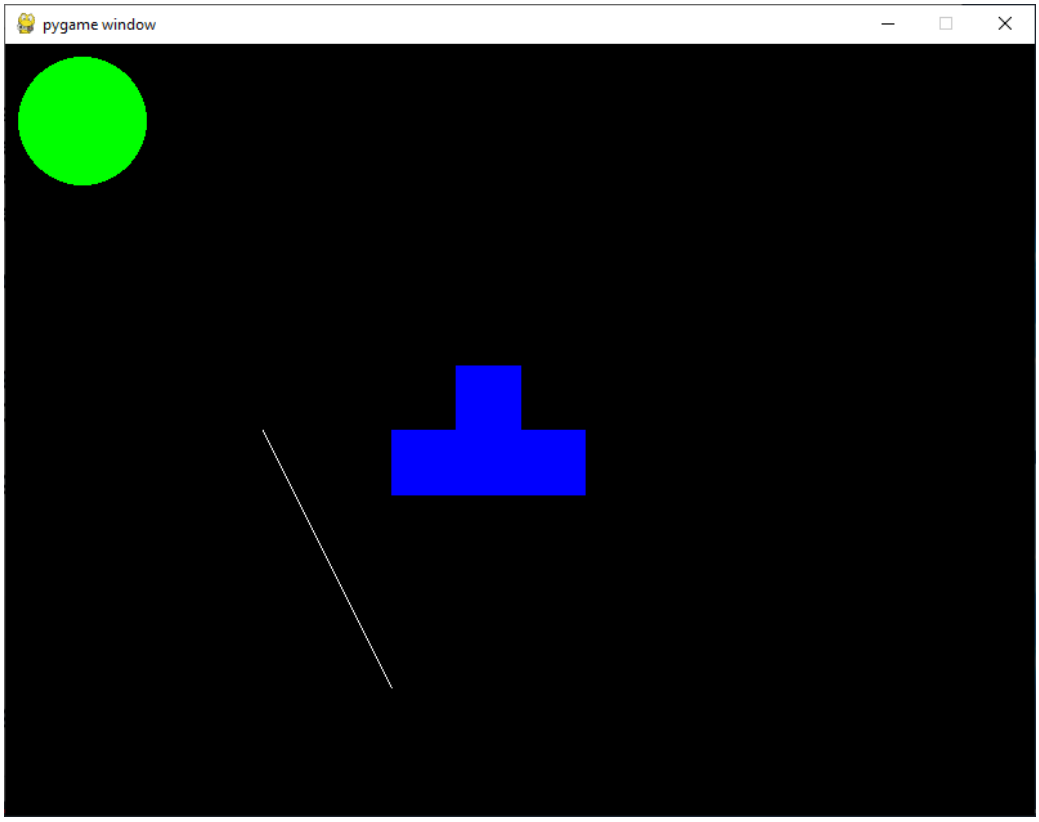
Vamos a dibujar figuras poligonales.

Como la siguiente figura:




```
29      # Dibujos
30      ventana.fill(NEGRO)
31      pygame.draw.circle(ventana, VERDE, (60,60),50)
32      pygame.draw.line(ventana, BLANCO, (200, 300), (300, 500))
33      pygame.draw.polygon(ventana, AZUL, ((300,300), (350,300),
34                                         (350, 250), (400, 250),
35                                         (400, 300), (450, 300),
36                                         (450, 350), (300, 350)))
```

El polygon cerrará el polígono, como indica la flecha.



Te propongo que practiques con las funciones rect, circle, line y polygon.

7.- Movimiento en pygame

En este vídeo vamos a empezar a ver el movimiento. El movimiento va a ser algo importante en muchos juegos. Movimiento de figuras o imágenes para intentar conseguir un objetivo.

Vamos a partir en una versión de programa que hemos realizado en capítulos anteriores en la que solo tendremos un fondo negro y un cuadrado.

```
1   import pygame
2
3   # Inicializar
4   pygame.init()
5
6   # Medidas
7   ANCHO = 800
8   ALTO = 600
9
10  # Colores
11  BLANCO = (255,255,255)
12  NEGRO = (0, 0, 0)
13  ROJO = (255, 0, 0)
14  VERDE = (0, 255, 0)
15  AZUL = (0, 0, 255)
16
17  # Ventana
18  ventana = pygame.display.set_mode((ANCHO, ALTO))
19
20  # Bucle principal
21  jugando = True
22  while jugando:
23      for event in pygame.event.get():
24          if event.type == pygame.QUIT:
25              jugando = False
26          if event.type == pygame.KEYDOWN:
27              if event.key == pygame.K_ESCAPE:
28                  jugando = False
29      # Dibujos
30      ventana.fill(NEGRO)
31      pygame.draw.rect(ventana, VERDE, (100,100, 50, 50))
```

```
32
33          # Actualizar
34          pygame.display.update()
35      # Salir
36      pygame.quit()
```

Si le damos a ejecutar:

Vamos a ver un cuadrado de color verde en un fondo negro.

```
1   import pygame
2
3   # Inicializar
4   pygame.init()
5
6   # Medidas
7   ANCHO = 800
8   ALTO = 600
9
10  # Colores
11  BLANCO = (255,255,255)
```

```
12    NEGRO = (0, 0, 0)
13    ROJO = (255, 0, 0)
14    VERDE = (0, 255, 0)
15    AZUL = (0, 0, 255)
16
17    # Ventana
18    ventana = pygame.display.set_mode((ANCHO, ALTO))
19
20    # Datos
21    pos_x = 100
22    pos_y = 100
23
24    # Bucle principal
25    jugando = True
26    while jugando:
27        for event in pygame.event.get():
28            if event.type == pygame.QUIT:
29                jugando = False
30            if event.type == pygame.KEYDOWN:
31                if event.key == pygame.K_ESCAPE:
32                    jugando = False
33
34        # Lógica
35        pos_x += 1
36
37        # Dibujos
38        ventana.fill(NEGRO)
39        pygame.draw.rect(ventana, VERDE, (pos_x,pos_y, 50, 50))
40
41        # Actualizar
42        pygame.display.update()
43    # Salir
44    pygame.quit()
```

En las línea 21 y 22 definimos dos variables para las coordenadas x e y.
En la línea 35 hacemos que la coordenada pos_x en cada ciclo incremente en 1.
En la línea 39 hemos cambiado los valores por las variables, ya que la variable pos_x se irá incrementando.
Este será el resultado:

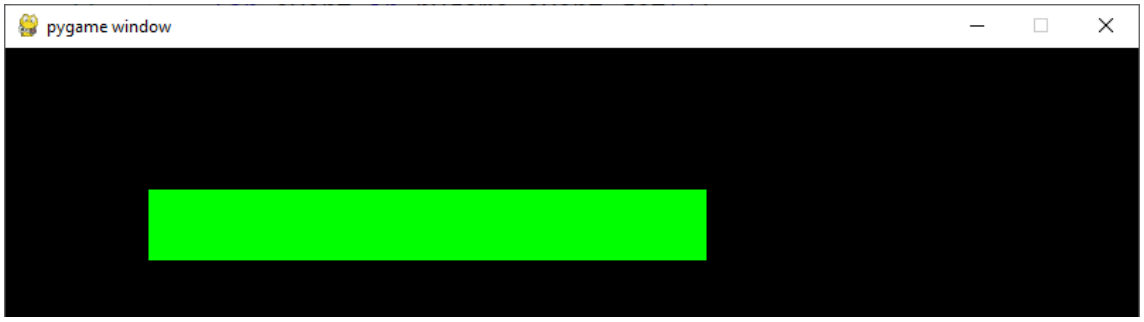
Observamos como el cuadro se desplaza de izquierda a derecha hasta salir de la ventana.

```
41      # Actualizar
42      pygame.display.update()
43      pygame.time.delay(5)
```

En la línea 43 hacemos una demora de 5 milisegundos para que el desplazamiento no sea tan rápido.

```
37      # Dibujos
38      # ventana.fill(NEGRO)
39      pygame.draw.rect(ventana, VERDE, (pos_x,pos_y, 50, 50))
```

Que pasa si en cada iteración no volvemos a pintar el fondo de la pantalla:

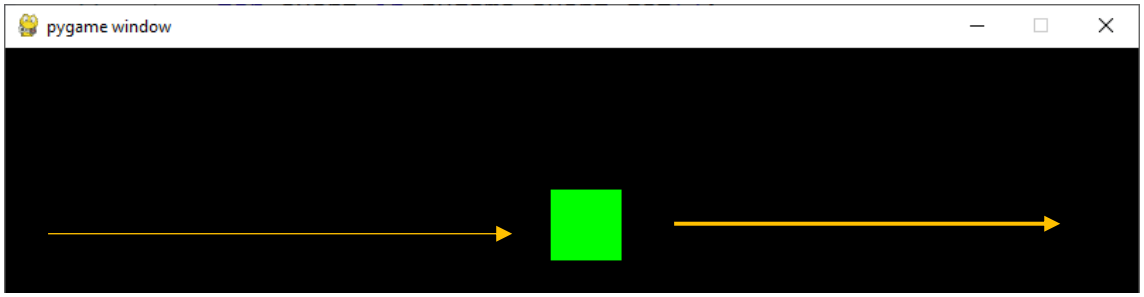


Se iría pintando el cuadrado pero dejando rastro.

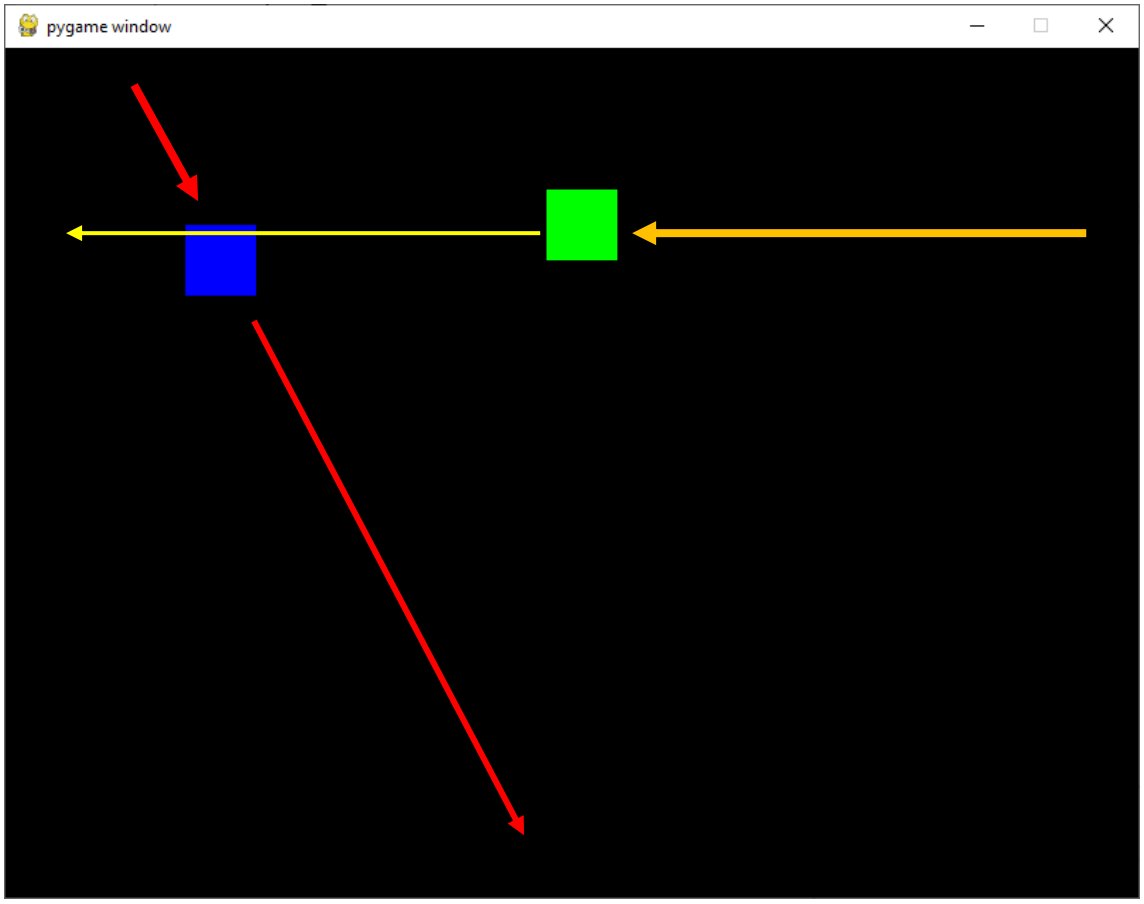
Como queremos que el cuadrado no se salga de la ventana vamos a realizar lo siguiente:

```
34      # Lógica
35      pos_x += 1
36      if pos_x > ANCHO:
37          pos_x = 0
```

Ahora queremos controlar que cuando el cuadro sale por la derecha que vuelva a empezar entrando por la izquierda.



Te propongo que realices este proyecto:



Para mi este será el resultado:

```
1   import pygame
2
3   # Inicializar
4   pygame.init()
5
6   # Medidas
7   ANCHO = 800
8   ALTO = 600
9
10  # Colores
11  BLANCO = (255,255,255)
12  NEGRO = (0, 0, 0)
13  ROJO = (255, 0, 0)
14  VERDE = (0, 255, 0)
15  AZUL = (0, 0, 255)
16
```

```
17  # Ventana
18  ventana = pygame.display.set_mode((ANCHO, ALTO))
19
20  # Datos
21  pos_x = 800
22  pos_y = 100
23
24  pos1_x = 10
25  pos1_y = 10
26
27  # Bucle principal
28  jugando = True
29  while jugando:
30      for event in pygame.event.get():
31          if event.type == pygame.QUIT:
32              jugando = False
33          if event.type == pygame.KEYDOWN:
34              if event.key == pygame.K_ESCAPE:
35                  jugando = False
36
37      # Lógica
38      pos_x -= 1
39      pos1_x += 1
40      pos1_y += 1
41      if pos1_y > 600:
42          pos1_y = 0
43      if pos1_x > 800:
44          pos1_x = 0
45      if pos_x < - 100:
46          pos_x = ANCHO
47
48      # Dibujos
49      ventana.fill(NEGRO)
50      pygame.draw.rect(ventana, VERDE, (pos_x,pos_y, 50, 50))
51      pygame.draw.rect(ventana, AZUL, (pos1_x,pos1_y, 50, 50))
```

```
52
53      # Actualizar
54      pygame.display.update()
55      pygame.time.delay(5)
56
57  # Salir
58  pygame.quit()
```

8.- Más movimiento de figuras

Este es el codigo con pequeñas rectificaciones.

```
import pygame

# Inicializar
pygame.init()

# Medidas
ANCHO = 800
ALTO = 600

# Colores
BLANCO = (255,255,255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)

# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))

# Datos
pos_x = 800
pos_y = 100

pos1_x = 200
pos1_y = 200

# Bucle principal
jugando = True
while jugando:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                jugando = False

    # Lógica
    pos_x -= 1
    pos1_x += 1
```

```
    pos1_y += 1
    if pos1_y > 600:
        pos1_y = -50
    if pos1_x > ANCHO:
        pos1_x = -50
    if pos_x > ANCHO:
        pos_x = -50

    # Dibujos
    ventana.fill(NEGRO)
    pygame.draw.rect(ventana, VERDE, (pos_x,pos_y, 50,
50))
    pygame.draw.rect(ventana, AZUL, (pos1_x,pos1_y, 50,
50))

    # Actualizar
    pygame.display.update()
    pygame.time.delay(5)

# Salir
pygame.quit()
```

9.- Poniendo texto en pygame

Vamos a escribir el siguiente código:

```
import pygame

# Inicializar
pygame.init()

# Medidas
ANCHO = 800
ALTO = 600

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)

# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))

fuente = pygame.font.SysFont("segoe print", 30)
texto = fuente.render("CUADRADOS", True, BLANCO)

# Datos
r_x = 100
r_y = 300

# Bucle principal
jugador = True
while jugador:
    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugador = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                jugador = False

    # Lógica
    r_x += 1
```

Creamos un objeto de tipo font.SysFont con una fuente de nuestro sistema "segoe print" y un tamaño de 30.

A esta función le pasamos el texto, pondremos True si queremos que se aplique un suavizado de borde y a continuación el color por ejemplo BLANCO.

```
        if r_x > ANCHO:
            r_x = -100

        # Dibujos
        ventana.fill(NEGRO)
        pygame.draw.rect(ventana, VERDE, (r_x, r_y, 100,
100))
        ventana.blit(texto, (10, 10))

         # Actualizar
        pygame.display.update()
        pygame.time.delay(5)

# Salir
pygame.quit()
```

Mediante al función blit pegamos el texto a la ventana con las correspondientes coordenadas.

Este será el resultado:

Vamos a cambiar:

```
20    fuente = pygame.font.SysFont("segoe print", 60)
21    texto = fuente.render("CUADRADOS", False, BLANCO)
```

Cambiamos de nuevo a True.

Para colocar un texto tenemos que realizar 3 pasos, primero definir una fuente, segundo crear un objeto de tipo superficie donde escribir el texto y asignarle color y tercero insertar esta superficie en la ventana.

```
20    fuente = pygame.font.SysFont("arial black", 32)
```

Cambiamos la fuente.

```
46        ventana.blit(texto, (260, 20))
```

Vamos a cambiar las coordenadas.

10.- Añadiendo puntuación

Vamos a seguir con el código del capítulo anterior:

```
import pygame

# Inicializar
pygame.init()

# Medidas
ANCHO = 800
ALTO = 600

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)

# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))

fuente1 = pygame.font.SysFont("arial black", 32)
fuente2 = pygame.font.SysFont("consolas", 24)

texto1 = fuente1.render("CUADRADOS", True, BLANCO)
puntos = 0
vueltas = 0

# Datos
r_x = 100
r_y = 300

# Bucle principal
jugador = True
while jugador:
    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugador = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_p:
```

Definimos otro tipo de fuente.

Creamos dos variables para el recuento de puntos y vueltas.

```
                puntos += 1
```

Cada vez que presionemos la tecla p la variable puntos se incrementará en 1.

```
    # Lógica
    r_x += 1
    if r_x > ANCHO:
        r_x = -100
        vueltas += 1
```

Cada vez que el cuadrado se salga de la ventana, la variable vueltas se incrementará en 1.

```
    texto_puntos = fuente2.render("Puntos: " +
str(puntos), True, BLANCO)
    texto_vueltas = fuente2.render("Vueltas: " +
str(vueltas), True, BLANCO)

    # Dibujos
    ventana.fill(NEGRO)
    pygame.draw.rect(ventana, VERDE, (r_x, r_y, 100,
100))
    ventana.blit(texto1, (260, 20))
    ventana.blit(texto_puntos, (30, 20))
    ventana.blit(texto_vueltas, (640, 20))

     # Actualizar
    pygame.display.update()
    pygame.time.delay(5)

# Salir
pygame.quit()
```

Texto_puntos y texto_vueltas tendrán su respectivo texto más su respectiva variable con funciones de contador, borde de la fuente suavizado y de color blanco.

El marco texto_puntos lo colocamos en las coordenada (30, 30).

El marco texto_vueltas lo colocamos en lsa coordenadas (640, 20)

Este será el resultado:

11.- Cuadros (frames) por segundo

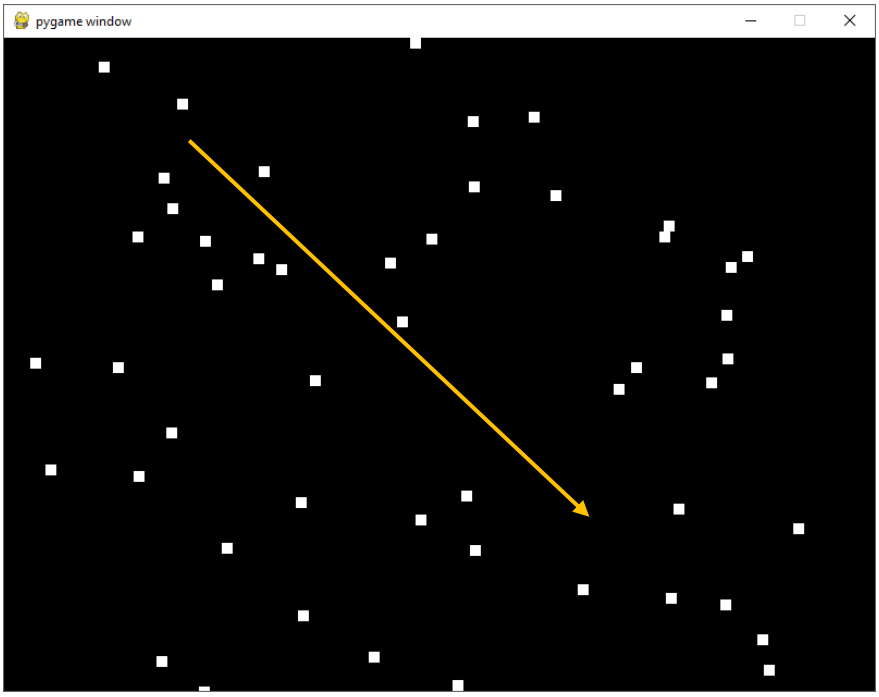
En esta capítulo vamos a ver como manejar la velocidad a la que se va mostrando las imágenes de un juego.

La frecuencia con las que se van actualizando las imágenes en un juego, es decir los frames por segundo.

```
1   import pygame
2   import random
3
4   # Medidas
5   ANCHO = 800
6   ALTO = 600
7
8   # Colores
9   BLANCO = (255, 255, 255)
10  NEGRO = (0, 0, 0)
11  ROJO = (255, 0, 0)
12  VERDE = (0, 255, 0)
13  AZUL = (0, 0, 255)
14
15  # Inicializar
16  pygame.init()
17  ventana = pygame.display.set_mode((ANCHO, ALTO))
18  fuente = pygame.font.SysFont("arial", 64)
19
20  # Datos
21  cuadrados = []
22  for i in range(50):
23      x = random.randint(1, 799)
24      y = random.randint(1, 599)
25      c = [x, y]
26      c = cuadrados.append(c)
27
28  jugando = True
29  while jugando:
30
```

```
31          # Eventos
32          for event in pygame.event.get():
33              if event.type == pygame.QUIT:
34                  jugando = False
35
36      # Lógica
37      for c in cuadrados:
38          c[0] += 1
39          c[1] += 2
40          if c[0] > 800:
41              c[0] = 0
42          if c[1] > 600:
43              c[1] = 0
44
45      # Imagenes
46      ventana.fill(NEGRO)
47
48      for c in cuadrados:
49          pygame.draw.rect(ventana, BLANCO, (c[0], c[1],
50                                             10, 10))
51
52      # Update
53      pygame.display.update()
54
55  pygame.quit()
```

Este será el resultado:



Tenemos 50 cuadrados que se mueven de arriba hacia abajo en modo diagonal.

Podrás observar que se mueven muy rápido, si quisiéramos simular unos asteroides.

```
52        # Update
53        pygame.display.update()
54        pygame.time.delay(10)
```

Hasta ahora lo que hemos hecho ha sido pausar el juego mediante la función delay del módulo time a la que le pasábamos 10 milisegundos.

De esta forma después de cada actualización de la pantalla el tiempo se pausa durante 10 milisegundos antes de volverse a actualizar la pantalla.

Vamos a ver otro forma de llevar esta tarea que además va a poder repercutir en otro aspectos de los juegos.

Eliminamos la línea 54.

```
15    # Inicializar
16    pygame.init()
17    ventana = pygame.display.set_mode((ANCHO, ALTO))
18    fuente = pygame.font.SysFont("arial", 64)
19
20    reloj = pygame.time.Clock()
```

Creamos un objeto de tipo reloj.

```
32    jugando = True
33    while jugando:
34
35        tiempo = reloj.tick()
```

El objeto de tipo reloj con la porpiedad tick puede medir el tiempo.

```
22    # Datos
23    cuadrados = []
24    for i in range(50):
25        x = random.randint(1, 799)
26        y = random.randint(1, 599)
27        c = [x, y]
28        c = cuadrados.append(c)
29
30    frames = 0
```

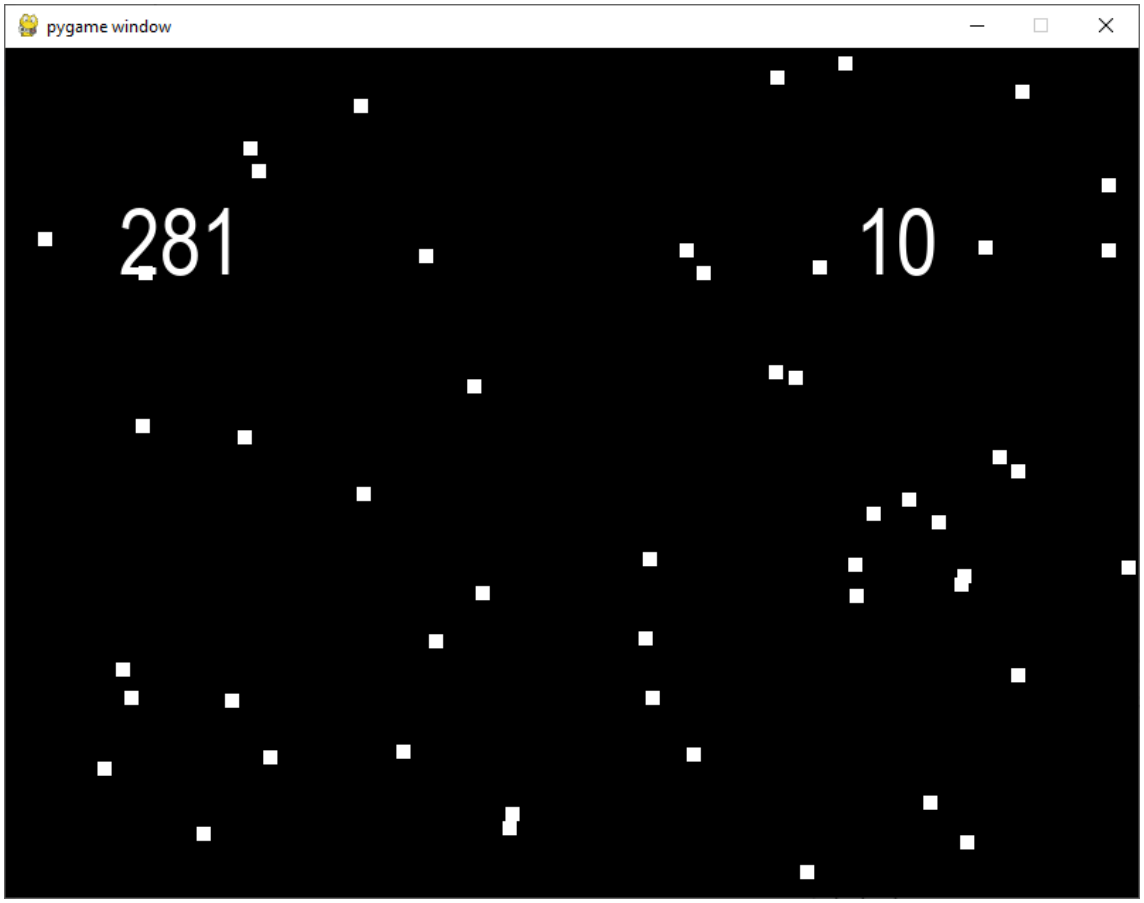
Creamos una variable llamada frames de tipo int asignándole el valor de 0.

```
32    jugando = True
33    while jugando:
34
35        tiempo = reloj.tick()
36        frames +=1
```

A la variable frames a su valor se le incrementa 1.

```
55        for c in cuadrados:
56            pygame.draw.rect(ventana, BLANCO, (c[0], c[1],
57                                               10, 10))
58
59        texto1 = fuente.render(str(frames), True, BLANCO)
60        texto2 = fuente.render(str(tiempo), True, BLANCO)
61        ventana.blit(texto1, (80,100))
62        ventana.blit(texto2, (600, 100))
```

Creamos dos textos para poder visualizar en la pantalla los frames y el tiempo.



Esta información no es muy útil.

```
import pygame
import random

# Medidas
ANCHO = 800
ALTO = 600

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)

# Inicializar
pygame.init()
ventana = pygame.display.set_mode((ANCHO, ALTO))
fuente = pygame.font.SysFont("arial", 64)
```

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

Creamos un objeto de tipo reloj, la clase time tiene un método llamado Clock() que nos va a permitir medir el tiempo que transcurre entre una actualización a otra.

```
# Datos
cuadrados = []
for i in range(50):
    x = random.randint(1, 799)
    y = random.randint(1, 599)
    c = [x, y]
    c = cuadrados.append(c)
frames = 0
transcurrido = 0
fps = 0
segundos = 0
```

Creamos las siguientes variables.

Transcurrido será el tiempo trancurrido. fps los cuadros por segundo

```
# Bucle principal
jugando = True
while jugando:
    if transcurrido >= 1000:
        fps = frames
        frames = 0
        segundos += 1
        transcurrido = 0
```

Con esta condición controlamos que si transcurrido es mayor o igual a 1000, es decir han transcurrido 1 segundo, la variable fps asumirá el valor frames, frames pasará a valer 0, se incrementa segundos en 1 y la variable transcurrido pasa a valer 0.

```
tiempo = reloj.tick()
transcurrido += tiempo
frames +=1

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False

    # Lógica
    for c in cuadrados:
        c[0] += 1
        c[1] += 2
        if c[0] > 800:
            c[0] = 0
        if c[1] > 600:
            c[1] = 0

    # Imagenes
    ventana.fill(NEGRO)

    for c in cuadrados:
        pygame.draw.rect(ventana, BLANCO, (c[0], c[1],
                                           10, 10))

    texto1 = fuente.render(str(fps), True, BLANCO)
    texto2 = fuente.render(str(segundos), True, BLANCO)
    ventana.blit(texto1, (80,100))
    ventana.blit(texto2, (600, 100))


    # Update
    pygame.display.update()
    pygame.time.delay(10)

pygame.quit()
```

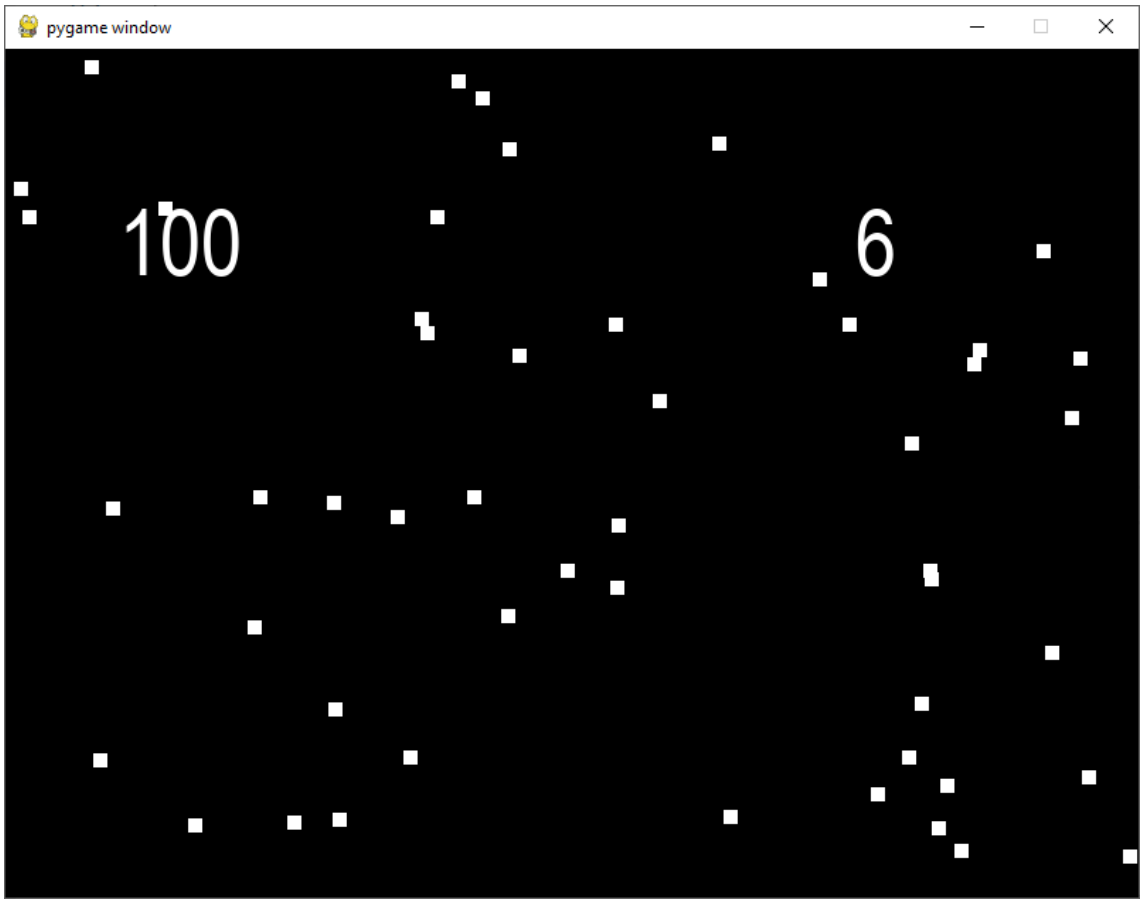
El método reloj.tick() nos va a devolver el tiempo que transcurre por cada cambio de la ventana que se lo pasaremos a la variable tiempo.

transcurrido + = tiempo, para que se vaya sumando el tiempo que vaya transcurriendo, entre actualización y actualización.

A frames su valor se le incrementa en 1.

Insertamos dos textos, en uno muestra los fotogramas por segundo y en el segundo los segundos transcurridos.

Pausamos el juego 10 milisegundos.



A la izquierda los cuadros por segundo y a la derecha los segundos transcurridos.

Cuadros por segundo – Se mide en FPS – Depende de la CPU o GPU (Frames)

Frecuencia de refresco – Se mide en HZ – Depende del monitor

Si el monitor es de 60 Hz se refrescará 60 veces por segundo, mostrando 60 imágenes nuevas por segundo.

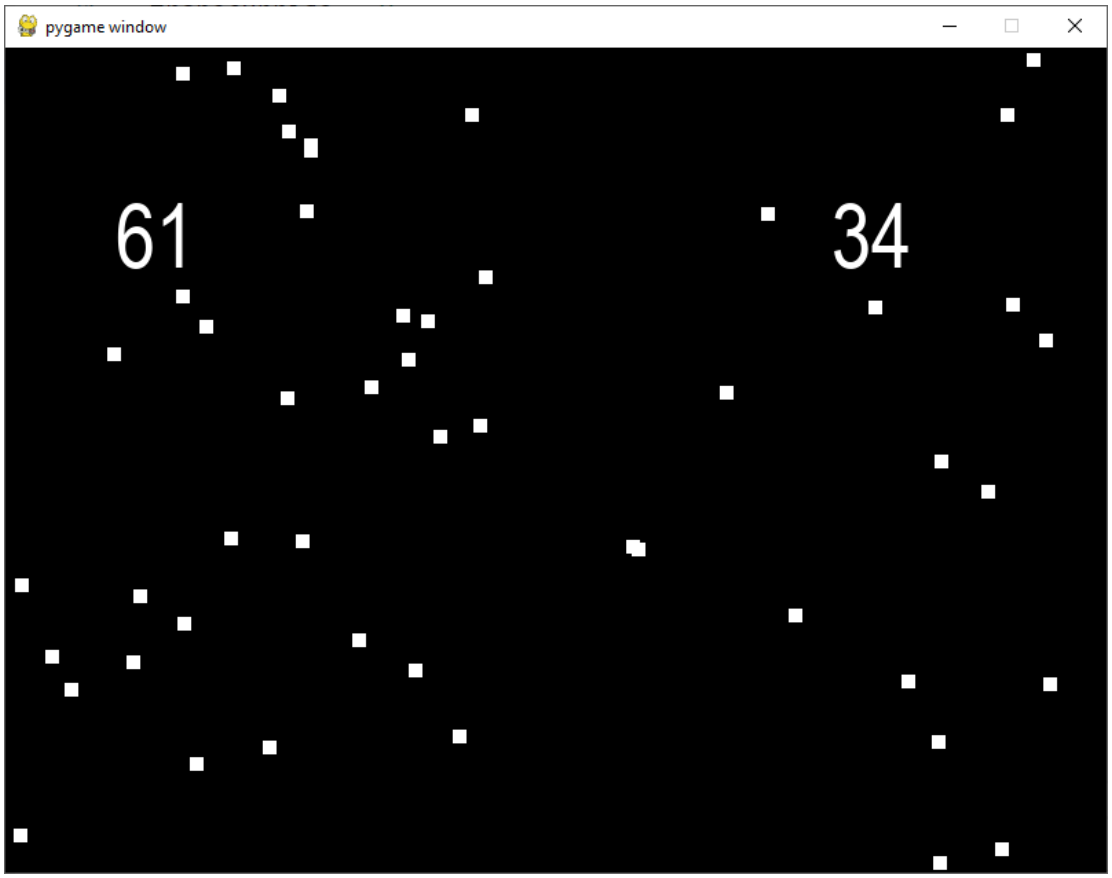
Si la CPU permite 100 0 150 FPS, las imágenes del juego cambiarán esas veces por segundo en la memoria del ordenador, pero sólo se mostrarán las que permita el monitor.

No nos interesa que un juego vaya a tantos frames por segundo, por una parte porque va a ser potencia desperdiciada ya que el monitor no va a poder mostrar.

Y por otra parte el juego no iría a la misma velocidad en dos ordenadores diferentes dependiendo de la capacidad de la CPU de cada uno.

Vamos a poder controlar tasa de frames por segundo de un juego y de esta forma hacer que el juego se vea igual en cualquier ordenador. Por lo tanto en el método reloj.tick() podemos pasarle como argumento la cantidad de frames por segundo que queremos que tenga el juego.

```
45          tiempo = reloj.tick(60)
```

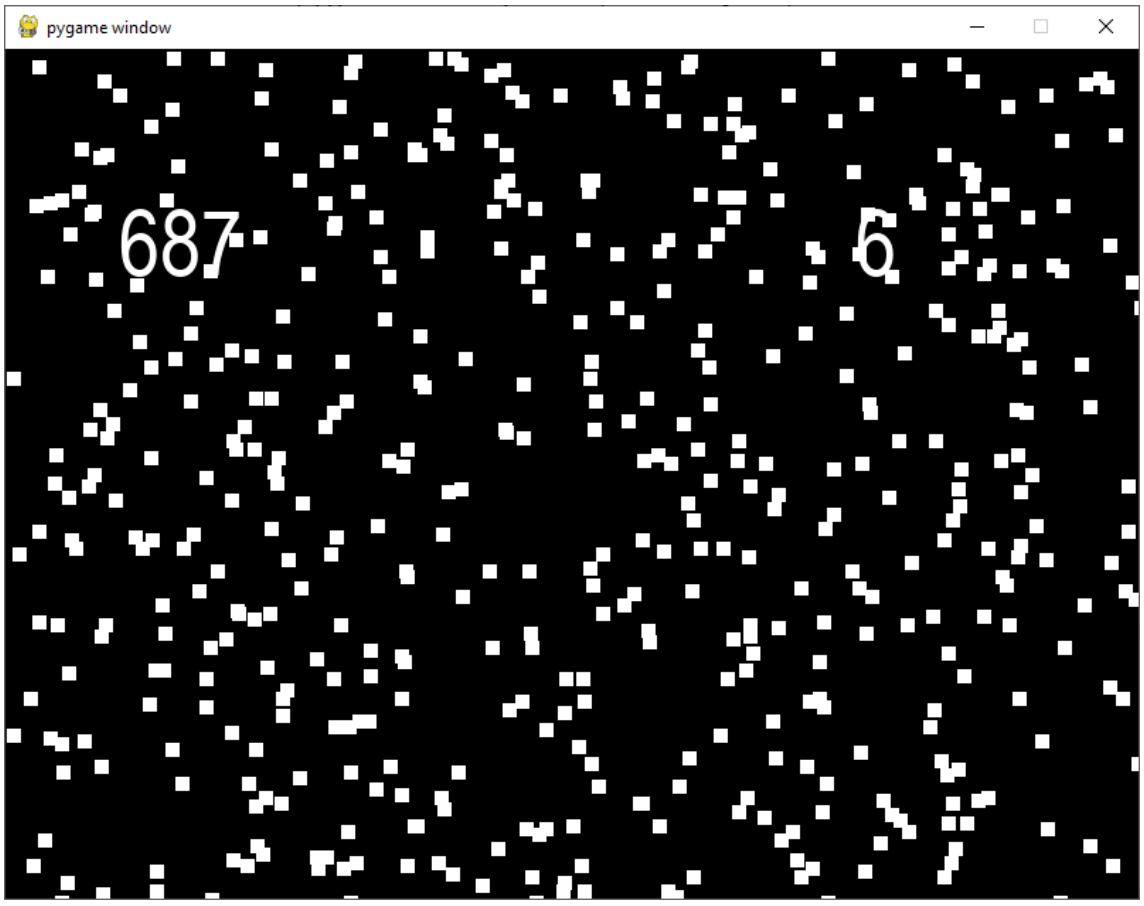


Vamos a cambiarlo a 30 y verás que va más despacio.

Vamos a quitar los 30.

```
45      tiempo = reloj.tick()

22  # Datos
23  cuadrados = []
24  for i in range(500):
25      x = random.randint(1, 799)
26      y = random.randint(1, 599)
27      c = [x, y]
28      c = cuadrados.append(c)
```

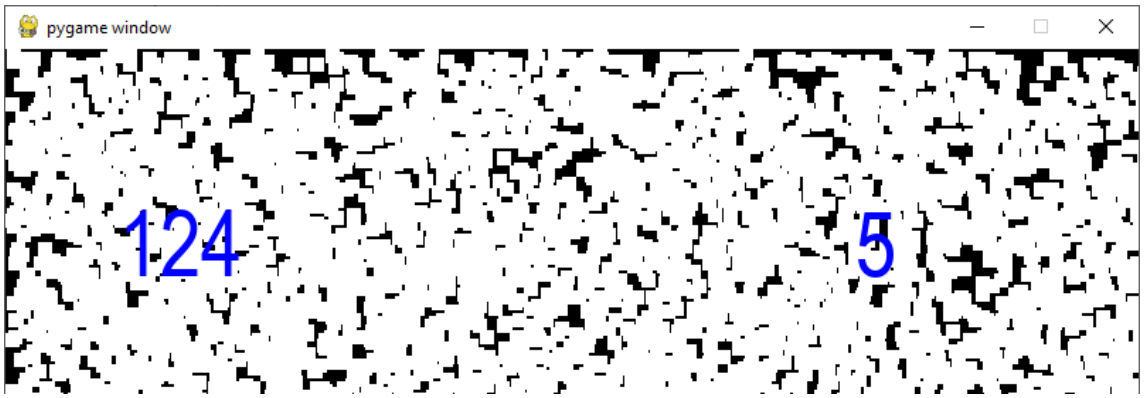
Cambiamos a 500 cuadrados.



Ya han bajado de 700 cuando mi CPU con 50 iba a 1400.

Si ponemos 5000 irá a unos 230.

Vamos a por 10000 pero vamos a cambiar el color del texto en azul para poderlo ver.



Por lo tanto cuando tenemos más imágenes el ordenador le cuesta más moverlas y bajan lo fps.

Nosotros lo dejaremos a 60 para aprovechar al máximo las capacidades del monitor y para no desperdiciar recursos de la CPU.

```
20    reloj = pygame.time.Clock(60)
```

Vamos a eliminar todo lo que no necesitaremos para el siguiente proyecto, para quedar de la siguiente manera:

```
import pygame
import random

# Medidas
ANCHO = 800
ALTO = 600

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)

# Inicializar
pygame.init()
ventana = pygame.display.set_mode((ANCHO, ALTO))
fuente = pygame.font.SysFont("arial", 64)
reloj = pygame.time.Clock()

# Datos
cuadrados = []
for i in range(50):
    x = random.randint(1, 799)
    y = random.randint(1, 599)
```

```
        c = [x, y]
        c = cuadrados.append(c)

# Bucle principal
jugando = True
while jugando:

    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False

    # Lógica
    for c in cuadrados:
        c[0] += 1
        c[1] += 2
        if c[0] > 800:
            c[0] = 0
        if c[1] > 600:
            c[1] = 0

    # Imagenes
    ventana.fill(NEGRO)

    for c in cuadrados:
        pygame.draw.rect(ventana, BLANCO, (c[0], c[1],
                                           10, 10))

    # Update
    pygame.display.update()

pygame.quit()
```

Dentro de modulo time tenemos la función pygame.time.get_ticks() que nos devuelve el tiempo en milisegundos.

Dentro de la clase pygame.time.Clock tenemos el método get_fps() que nos devuelve los fps directamente sin tener que calcularlos.

12.- Moviendo personajes en pygame

Ya hemos visto como hacer que una figura se mueva por la ventana en una dirección determinada.

Vamos hacer ahora como se puede hacer para que sea el usuario el que mueva una figura mediante las teclas.

Recogemos la plantilla de videos anteriores y realizamos las siguientes modificaciones:

```
import pygame
import random

# Medidas
ANCHO = 800
ALTO = 600

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)
NARANJA = (255, 128, 0)

# Inicializar
pygame.init()
ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

# Datos
aste_pos_x = -60
aste_pos_y = 70
nave_pos_x = 400
nave_pos_y = 400

# Bucle principal
jugando = True
while jugando:

    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
```

Definimos el color Naranja.

Definimos las variables de las coordenadas de los dos cuadrados que agregaremos a la ventana.

```
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                jugando = False
            if event.key == pygame.K_RIGHT:
                nave_pos_x += 10
            if event.key == pygame.K_LEFT:
                nave_pos_x -= 10
            if event.key == pygame.K_DOWN:
                nave_pos_y +=10
            if event.key == pygame.K_UP:
                nave_pos_y -= 10
```

Con las teclas derecha, izquierda, arriba y abajo controlaremos el movimiento de un cuadrado que se desplazará 10 pixeles en dirección a la flecha que seleccionemos

```
    # Lógica
    aste_pos_x += 5

    if aste_pos_x > ANCHO:
        aste_pos_x = - 60
```

El cuadro de color Naranja de desplaza de izquierda a derecha hasta desaparecer por la ventana y vuelve aparecer por la parte izquierda, continuamente.

```
    # Dibujos
    ventana.fill(NEGRO)
```

Dibujamos los dos cuadrados con sus respectivas coordenadas y dimensiones.

```
    pygame.draw.rect(ventana, NARANJA, (aste_pos_x,
aste_pos_y, 60, 60))
    pygame.draw.rect(ventana, VERDE, (nave_pos_x,
nave_pos_y, 60, 60))

    # Update
    pygame.display.update()

pygame.quit()
```

El inconveniente es que para mover el cuadrado tenemos que apretar y soltar la tecla ya que si la mantenemos apretada el cuadrado no se mueve, además el movimiento del cuadrado es poco suave.

Para ello vamos a realizar algunas modificaciones.

Vamos a simular las leyes de la física del movimiento.

E = V x T → Espacio es igual a Velocidad por Tiempo.

O Espacio_Final es igual Espacio_Inicial más la Velocidad por el Tiempo

$$Ef = Ei + V \times T$$

Vamos a mejorar el código:

```
import pygame
import random

# Medidas
ANCHO = 800
ALTO = 600

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)
NARANJA = (255, 128, 0)

# Inicializar
pygame.init()
ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

# Datos
aste_pos_x = -60
aste_pos_y = 70
aste_vel_x = 5
nave_pos_x = 400
nave_pos_y = 400
nave_vel_x = 0
nave_vel_y = 0

# Bucle principal
jugando = True
while jugando:

    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
```

Variables de velocidad de la nave.

```
            jugando = False
        if event.key == pygame.K_RIGHT:
            nave_vel_x = 10
        if event.key == pygame.K_LEFT:
            nave_vel_x = -10
        if event.key == pygame.K_DOWN:
            nave_vel_y = 10
        if event.key == pygame.K_UP:
            nave_vel_y = -10
    if event.type == pygame.KEYUP:
        if event.key == pygame.K_RIGHT:
            nave_vel_x = 0
        if event.key == pygame.K_LEFT:
            nave_vel_x = 0
        if event.key == pygame.K_DOWN:
            nave_vel_y = 0
        if event.key == pygame.K_UP:
            nave_vel_y = 0
```

Cuando presionamos las flechas.

KEYDOWN

Cuando dejamos de presionar las flechas.

KEYUP

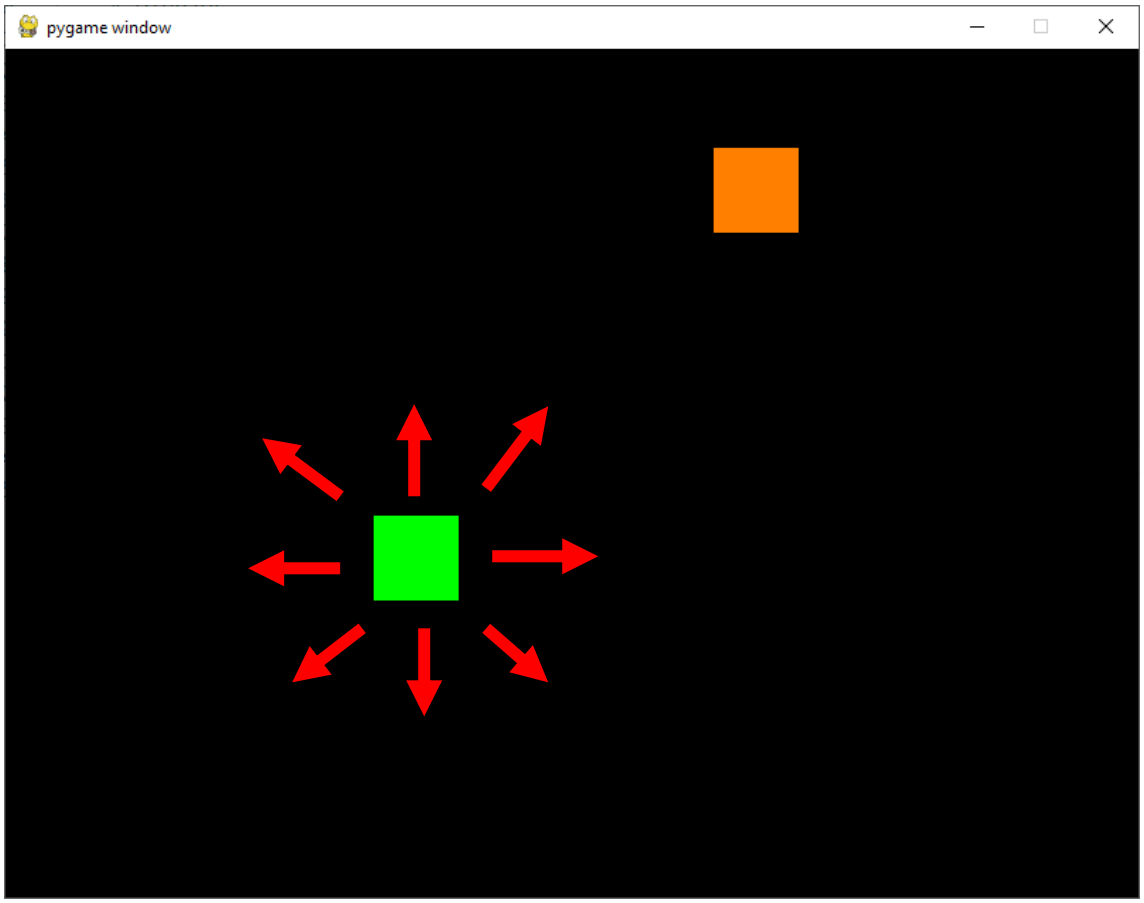
```
    # Lógica
    aste_pos_x += aste_vel_x
    if aste_pos_x > ANCHO:
        aste_pos_x = - 60

    nave_pos_x += nave_vel_x
    nave_pos_y += nave_vel_y

    # Dibujos
    ventana.fill(NEGRO)

    pygame.draw.rect(ventana, NARANJA, (aste_pos_x,
aste_pos_y, 60, 60))
    pygame.draw.rect(ventana, VERDE, (nave_pos_x,
nave_pos_y, 60, 60))

    # Update
    pygame.display.update()

pygame.quit()
```

La nave que se desplaza sola de izquierda a derecha.

El valor de las variables nave_vel_x y nave_vel_y según el usuario mantenga o no presionada las flechas del teclado.

Los valores de las variable aste_pos_x y aste_pos_y las controla el programa.

Los valores de las variables nave_pos_x y nave_pos_y las controla el usuario con las teclas de dirección del teclado.


pygame window

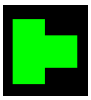
13.- Cambiando orientación de las figuras

Este será el código:

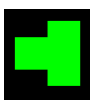
```
import pygame
import random

# Medidas
ANCHO = 800
ALTO = 600

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)
NARANJA = (255, 128, 0)

# Funciones
```

```
def nave_arriba(superficie, x ,y):
    pygame.draw.rect(superficie, VERDE, (x, y, 60, 60))
    pygame.draw.rect(superficie, NEGRO, (x, y, 15, 30))
    pygame.draw.rect(superficie, NEGRO, (x + 45, y, 15,
30))
```

```
def nave_abajo(superficie, x ,y):
    pygame.draw.rect(superficie, VERDE, (x, y, 60, 60))
    pygame.draw.rect(superficie, NEGRO, (x, y+30, 15,
30))
    pygame.draw.rect(superficie, NEGRO, (x + 45, y+30,
15, 30))
```



```
def nave_derecha(superficie, x ,y):
    pygame.draw.rect(superficie, VERDE, (x, y, 60, 60))
    pygame.draw.rect(superficie, NEGRO, (x+30, y, 30,
15))
    pygame.draw.rect(superficie, NEGRO, (x+30, y+45,
30, 15))
```



```
def nave_izquierda(superficie, x, y):
    pygame.draw.rect(superficie, VERDE, (x, y, 60, 60))
    pygame.draw.rect(superficie, NEGRO, (x, y, 30, 15))
```

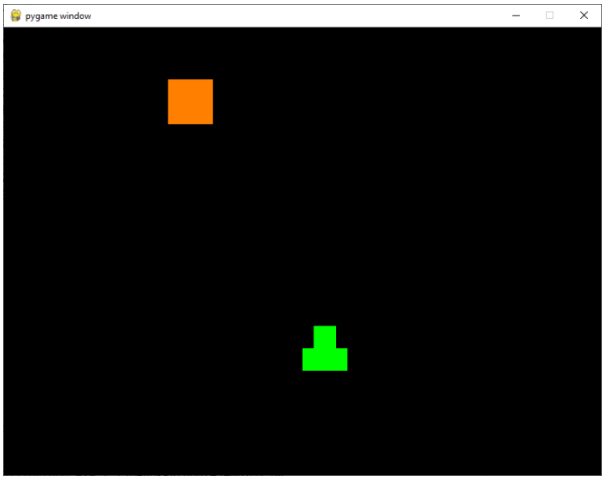
```
    pygame.draw.rect(superficie, NEGRO, (x, y+45, 30,
15))


# Inicializar
pygame.init()
ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

# Datos
aste_pos_x = -60
aste_pos_y = 70
aste_vel_x = 5

nave_pos_x = 400
nave_pos_y = 400
nave_vel_x = 0
nave_vel_y = 0

direccion = "arriba"

# Bucle principal
jugando = True
while jugando:

    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                jugando = False
            if event.key == pygame.K_RIGHT:
                direccion = "derecha"
                nave_vel_x = 10
            if event.key == pygame.K_LEFT:
                direccion = "izquierda"
                nave_vel_x = -10
            if event.key == pygame.K_DOWN:
                direccion = "abajo"
```

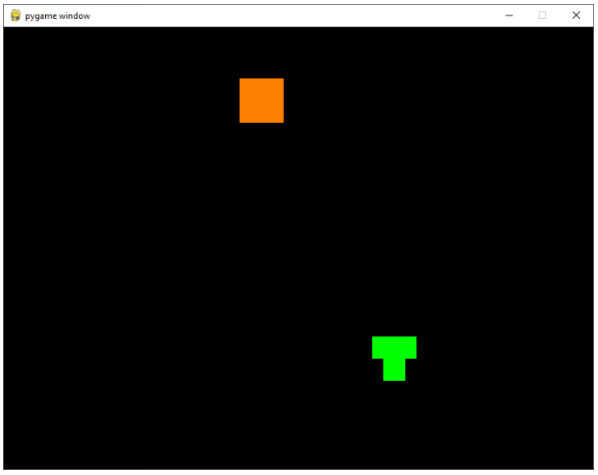
La variable direccion guarda la dirección que le asignaremos según la tecla de dirección que presionemos del teclado.

```
                nave_vel_y = 10
            if event.key == pygame.K_UP:
                direccion = "arriba"
                nave_vel_y = -10
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_RIGHT:
                nave_vel_x = 0
            if event.key == pygame.K_LEFT:
                nave_vel_x = 0
            if event.key == pygame.K_DOWN:
                nave_vel_y = 0
            if event.key == pygame.K_UP:
                nave_vel_y = 0

    # Lógica
    aste_pos_x += aste_vel_x
    if aste_pos_x > ANCHO:
        aste_pos_x = - 60

    nave_pos_x += nave_vel_x
    nave_pos_y += nave_vel_y


    # Dibujos
    ventana.fill(NEGRO)

    pygame.draw.rect(ventana, NARANJA, (aste_pos_x,
aste_pos_y, 60, 60))
    if direccion == "arriba":
        nave_arriba(ventana, nave_pos_x, nave_pos_y)
    elif direccion == "abajo":
        nave_abajo(ventana, nave_pos_x, nave_pos_y)
    elif direccion == "derecha":
        nave_derecha(ventana, nave_pos_x, nave_pos_y)
    elif direccion == "izquierda":
        nave_izquierda(ventana, nave_pos_x, nave_pos_y)


    # Update
    pygame.display.update()

pygame.quit()
```

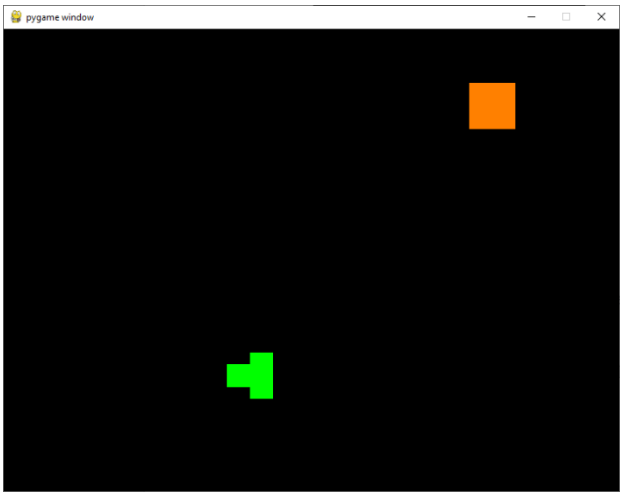
Según el valor que asuma dirección, esta llamará a la correspondiente función.

nave_arriba(), nave_abajo(), nave_derecha() o nave_izquierda()


pygame window


pygame window


pygame window


pygame window

14.- Animación de figuras en pygame

Continuando con el proyecto anterior vamos a hacer que nuestro asteroide vaya rotando.

Este será el código final:

```
import pygame
import random

# Medidas
ANCHO = 800
ALTO = 600

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)
NARANJA = (255, 128, 0)

# Funciones
def nave_arriba(superficie, x ,y):
    pygame.draw.rect(superficie, VERDE, (x, y, 60, 60))
    pygame.draw.rect(superficie, NEGRO, (x, y, 15, 30))
    pygame.draw.rect(superficie, NEGRO, (x + 45, y, 15,
30))

def nave_abajo(superficie, x ,y):
    pygame.draw.rect(superficie, VERDE, (x, y, 60, 60))
    pygame.draw.rect(superficie, NEGRO, (x, y+30, 15,
30))
    pygame.draw.rect(superficie, NEGRO, (x + 45, y+30,
15, 30))

def nave_derecha(superficie, x ,y):
    pygame.draw.rect(superficie, VERDE, (x, y, 60, 60))
    pygame.draw.rect(superficie, NEGRO, (x+30, y, 30,
15))
    pygame.draw.rect(superficie, NEGRO, (x+30, y+45,
30, 15))

def nave_izquierda(superficie, x, y):
    pygame.draw.rect(superficie, VERDE, (x, y, 60, 60))
```

```
    pygame.draw.rect(superficie, NEGRO, (x, y, 30, 15))
    pygame.draw.rect(superficie, NEGRO, (x, y+45, 30,
15))
```

```
def asteroide_1(superficie, x, y):
    pygame.draw.rect(superficie, NARANJA, (x, y, 60,
60))
    pygame.draw.rect(superficie, NEGRO, (x, y, 20, 20))
```

```
def asteroide_2(superficie, x, y):
    pygame.draw.rect(superficie, NARANJA, (x, y, 60,
60))
    pygame.draw.rect(superficie, NEGRO, (x+40, y, 20,
20))
```

```
def asteroide_3(superficie, x, y):
    pygame.draw.rect(superficie, NARANJA, (x, y, 60,
60))
    pygame.draw.rect(superficie, NEGRO, (x+40, y+40,
20, 20))
```

```
def asteroide_4(superficie, x, y):
    pygame.draw.rect(superficie, NARANJA, (x, y, 60,
60))
    pygame.draw.rect(superficie, NEGRO, (x, y+40, 20,
20))
```

```
# Inicializar
pygame.init()
ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

# Datos
aste_pos_x = -60
aste_pos_y = 70
aste_vel_x = 5

nave_pos_x = 400
nave_pos_y = 400
nave_vel_x = 0
nave_vel_y = 0
```

```
direccion = "arriba"

contador = 0

# Bucle principal
jugando = True
while jugando:

    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                jugando = False
            if event.key == pygame.K_RIGHT:
                direccion = "derecha"
                nave_vel_x = 10
            if event.key == pygame.K_LEFT:
                direccion = "izquierda"
                nave_vel_x = -10
            if event.key == pygame.K_DOWN:
                direccion = "abajo"
                nave_vel_y = 10
            if event.key == pygame.K_UP:
                direccion = "arriba"
                nave_vel_y = -10
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_RIGHT:
                nave_vel_x = 0
            if event.key == pygame.K_LEFT:
                nave_vel_x = 0
            if event.key == pygame.K_DOWN:
                nave_vel_y = 0
            if event.key == pygame.K_UP:
                nave_vel_y = 0

    # Lógica
    aste_pos_x += aste_vel_x
    if aste_pos_x > ANCHO:
```

Según el valor de al variable contador el asteroide tendrá un determinado giro.

```
        aste_pos_x = - 60

    nave_pos_x += nave_vel_x
    nave_pos_y += nave_vel_y


    # Dibujos
    ventana.fill(NEGRO)
```

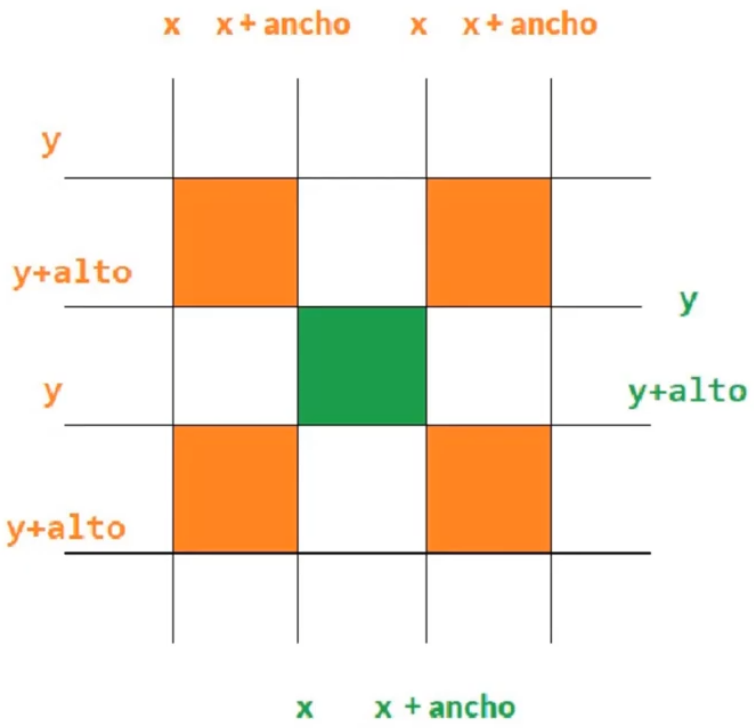
Según el valor de contador según las siguientes condiciones llamará a una función u otra.

Si contador mayor o igual a 41 lo volvemos a reiniciar a 0.

```
    contador += 1
    if contador >= 41:
        contador = 1
    if contador < 11:
        asteroide_1(ventana, aste_pos_x, aste_pos_y)
    elif  contador < 21:
        asteroide_2(ventana, aste_pos_x, aste_pos_y)
    elif contador < 31:
        asteroide_3(ventana, aste_pos_x, aste_pos_y)
    elif contador < 41:
        asteroide_4(ventana, aste_pos_x, aste_pos_y)


    if direccion == "arriba":
        nave_arriba(ventana, nave_pos_x, nave_pos_y)
    elif direccion == "abajo":
        nave_abajo(ventana, nave_pos_x, nave_pos_y)
    elif direccion == "derecha":
        nave_derecha(ventana, nave_pos_x, nave_pos_y)
    elif direccion == "izquierda":
        nave_izquierda(ventana, nave_pos_x, nave_pos_y)


   # Update
    pygame.display.update()

pygame.quit()
```

15.- Colisiones en pygame

En este capítulo vamos a ver como hacer que dos objetos choquen.



Para entender las colisiones vamos a analizar este esquema en que vamos a tener el cuadrado naranja y el cuadrado verde.

Las coordenadas del cuadrado naranja x + x + ancho y de la y hasta la y + ancho, igualmente las coordenadas del cuadrado verde.

Vamos a trabajar sobre este código:

```
import pygame

# Inicializar
pygame.init()
reloj = pygame.time.Clock()

# Medidas
ANCHO = 800
ALTO = 600

# Colores
BLANCO = (255,255,255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)
```

```
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)
NARANJA = (255, 128, 0)

# Funciones
def nave(superficie, x, y, ancho, alto):
    pygame.draw.rect(superficie, VERDE, (x, y, ancho,
alto))

def asteroide(superficie, x, y, ancho, alto):
    pygame.draw.rect(superficie, NARANJA, (x, y, ancho,
alto))

# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))

# Datos
aste_ancho = 60
aste_alto = 60
aste_pos_x = 100
aste_pos_y = 100
aste_vel_x = 5

nave_ancho = 60
nave_alto = 60
nave_pos_x = 600
nave_pos_y = 500
nave_vel_x = 0
nave_vel_y = 0

# Bucle principal
reloj.tick(60)
jugando = True
while jugando:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                jugando = False
            if event.key == pygame.K_RIGHT:
                nave_vel_x = 10
```

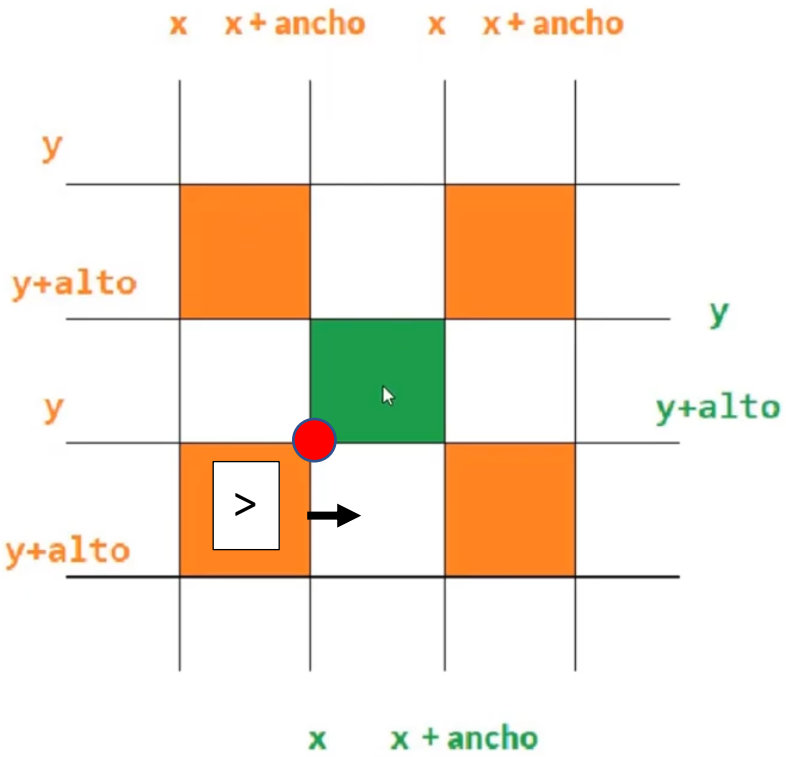
```
            if event.key == pygame.K_LEFT:
                nave_vel_x = -10
            if event.key == pygame.K_DOWN:
                nave_vel_y = 10
            if event.key == pygame.K_UP:
                nave_vel_y = -10
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_RIGHT:
                nave_vel_x = 0
            if event.key == pygame.K_LEFT:
                nave_vel_x = 0
            if event.key == pygame.K_DOWN:
                nave_vel_y = 0
            if event.key == pygame.K_UP:
                nave_vel_y = 0

    # Lógica
    aste_pos_x += aste_vel_x
    if aste_pos_x > ANCHO:
        aste_pos_x =- aste_ancho

    nave_pos_x += nave_vel_x
    nave_pos_y += nave_vel_y



    # Dibujos
    ventana.fill(NEGRO)
    asteroide(ventana, aste_pos_x, aste_pos_y,
aste_ancho, aste_alto)
    nave(ventana, nave_pos_x, nave_pos_y, nave_ancho,
nave_alto)

    # Actualizar
    pygame.display.update()
    pygame.time.delay(10)

# Salir
pygame.quit()
```

Vamos a realizar las siguientes modificaciones:

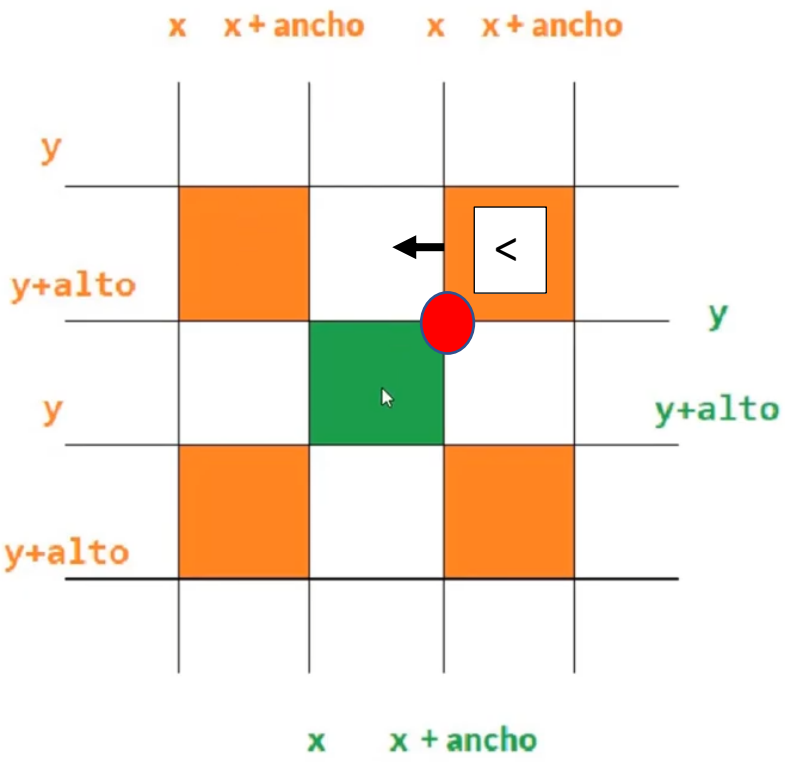
Este es el código que hemos agreagado:

```
71      # Lógica
72      aste_pos_x += aste_vel_x
73      if aste_pos_x > ANCHO:
74          aste_pos_x =- aste_ancho
75
76      nave_pos_x += nave_vel_x
77      nave_pos_y += nave_vel_y
78
79      if aste_pos_x + aste_ancho > nave_pos_x and \
80          aste_pos_x < nave_pos_x + nave_ancho and \
81          aste_pos_y + aste_alto > nave_pos_y and \
82          aste_pos_y < nave_pos_y + nave_alto:
83          pygame.time.delay(1000)
84          aste_pos_x = 100
85          aste_pos_y = 100
```

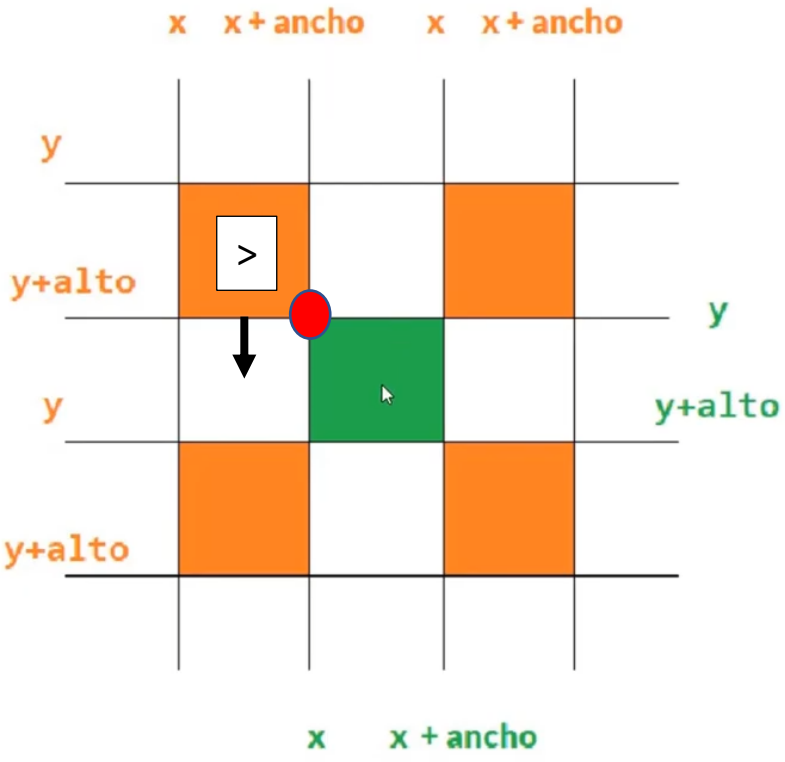
Línea 79



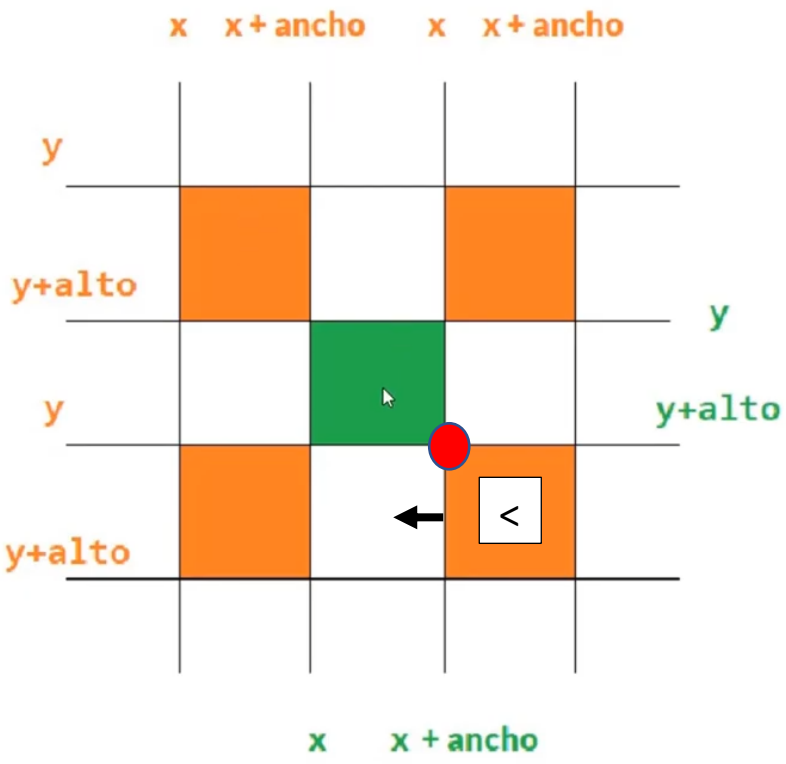
Línea 80



Línea 81



Línea 82



Las líneas 84 y 85 hacer que el asteroide retorne a un punto de partida.

```
79      if aste_pos_x + aste_ancho > nave_pos_x + 20 and \
80          aste_pos_x + 20 < nave_pos_x + nave_ancho and \
81          aste_pos_y + aste_alto > nave_pos_y + 20 and \
82          aste_pos_y + 20 < nave_pos_y + nave_alto:
83          pygame.time.delay(1000)
84          aste_pos_x = 100
85          aste_pos_y = 100
```

Sumamos + 20 para que los dos objetos se interpongan 20 pixeles entre ellos.

16.- Cargando imágenes

Hasta ahora solamente hemos estado utilizando figuras y dibujos, pero si queremos dar una estética más interesante a nuestro juegos no va hacer falta utilizar imágenes, que hayamos diseñador nosotros o las que tengamos a nuestra disposición para poder ponerlas en los juegos.

Vamos a ver como podemos cargar las imágenes y utilizarlas en juego con pygame.

Partiendo del siguiente código:

```
import pygame

# Inicializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
NARANJA = (255, 128,0)
VERDE = (0, 255, 0)


ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

# Bucle principal
jugando = True
while jugando:

    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                jugando = False
```

```
    # Imagenes
    ventana.fill(NEGRO)
    pygame.draw.rect(ventana, NARANJA, (200, 200, 60,
60))
    pygame.draw.rect(ventana, VERDE, (600, 500, 60,
60))
    pygame.display.update()

pygame.quit()
```

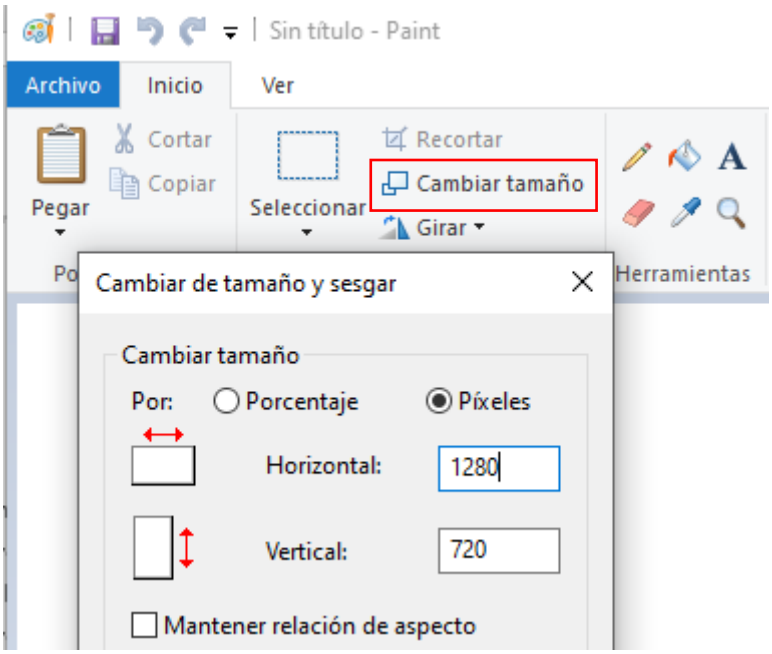
Este será el resultado:

Ahora lo que queremos hacer es en vez de rellenar de negro la ventana queremos poner un fondo.

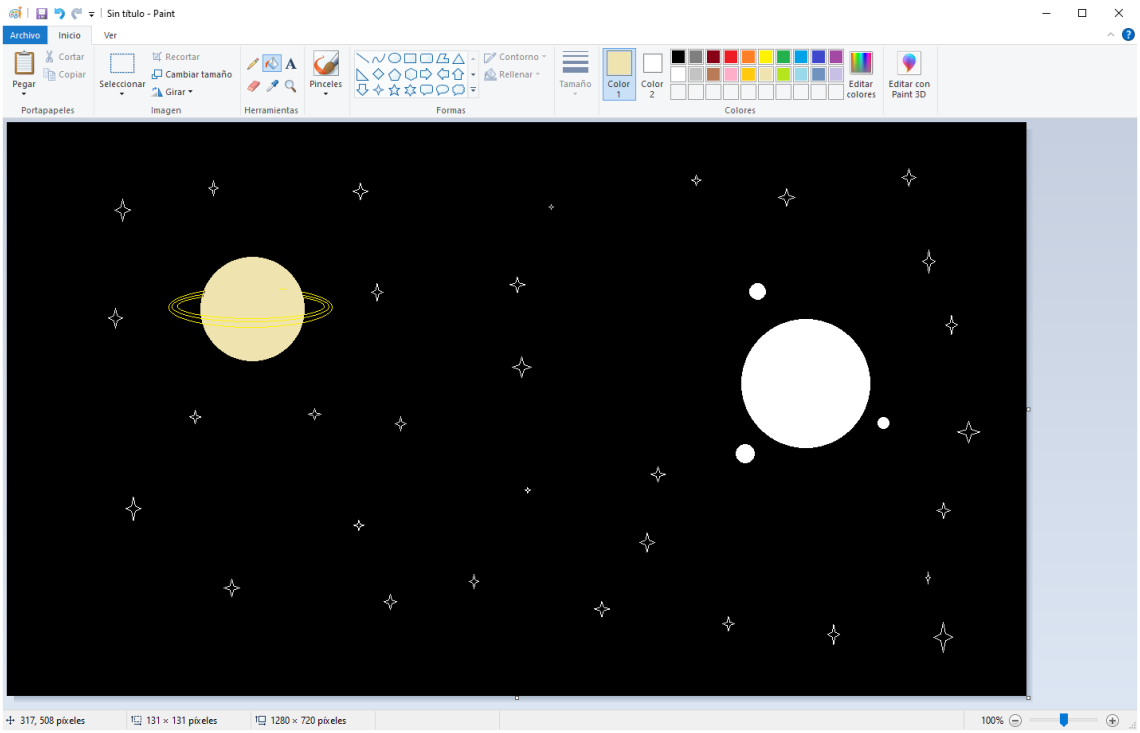
En lugar del cuadrado naranja un asteroide y en lugar del cuadro verde una nave.

Vamos a buscar los archivos con los dibujos.

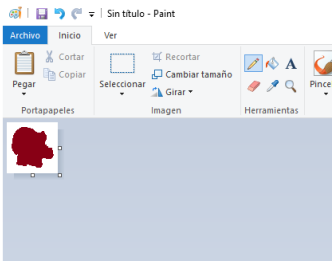
En este primer ejemplo lo vamos a dibujar con el Paint de Windows.



Vamos a cambiar las dimensiones que tienen que coincidir con las dimensiones de la ventana.

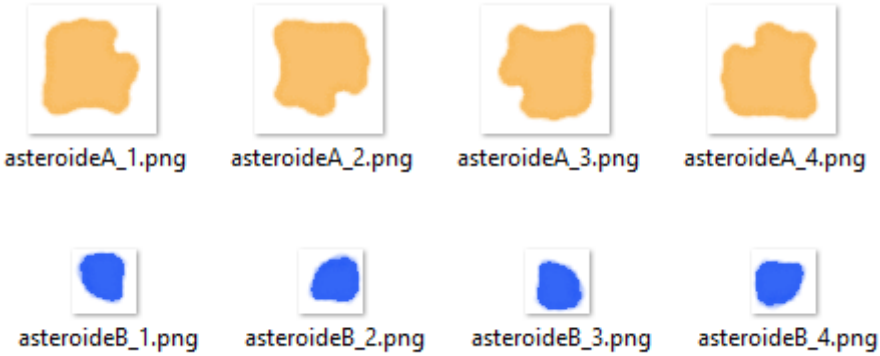


Realizaremos el siguiente dibujo.
Ahora lo vamos a guardar con el formato PNG que para los juegos es el más recomendado.
Vamos a realizar el asteroide de 64 x 64 pixeles.



También lo guardamos como PNG.

Otros programas que podemos utilizar que son totalmente gratuitos son el Gimp y el Krita.



Ya tenemos los dos asteroides.

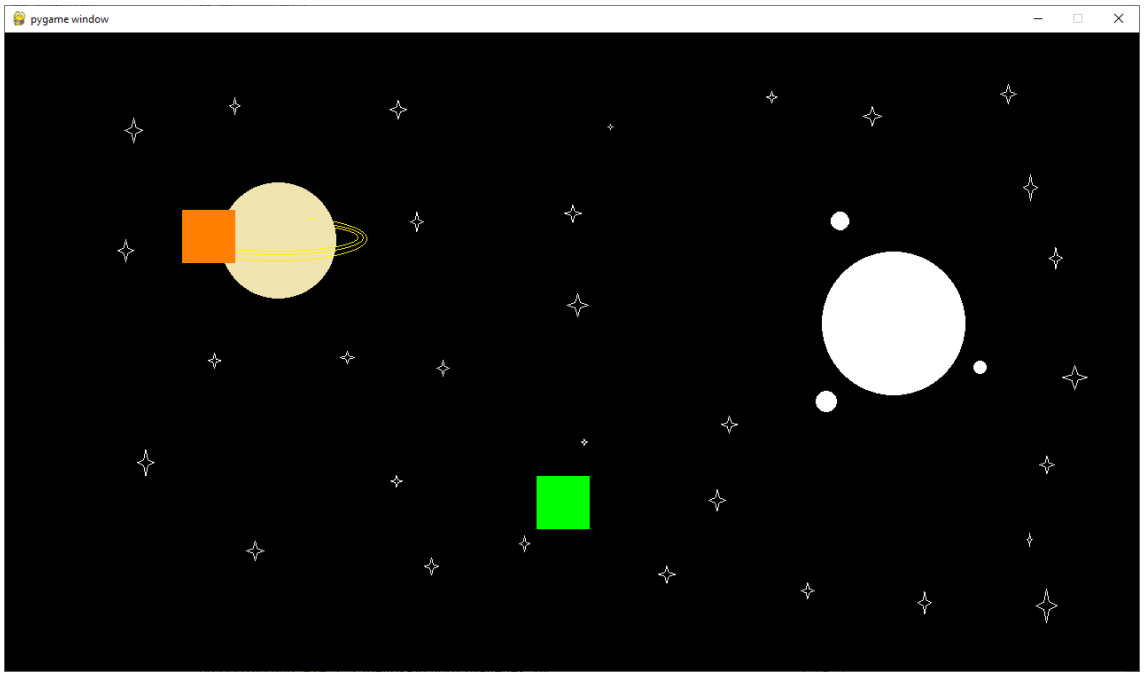
```
import pygame

# Inicializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
NARANJA = (255, 128,0)
VERDE = (0, 255, 0)

ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

#  Imágenes
fondo = pygame.image.load("fondo.png")

# Bucle principal
jugando = True
while jugando:

    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
```

Creamos un objeto de tipo imagen llamado fondo, para cargar la imagen desde nuestro ordenador, si esta se encontrarse en una carpeta, habría que indicar la ruta.

```
            if event.type == pygame.QUIT:
                jugando = False
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    jugando = False

    # Imagenes
    ventana.blit(fondo,(0,0))

    pygame.draw.rect(ventana, NARANJA, (200, 200, 60,
60))
    pygame.draw.rect(ventana, VERDE, (600, 500, 60,
60))
    pygame.display.update()

pygame.quit()
```

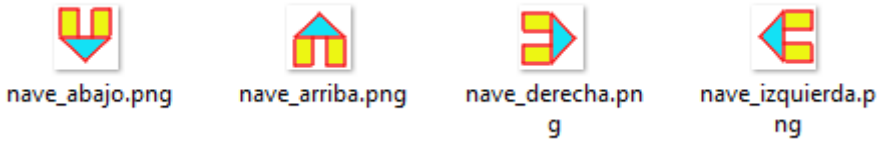
Para insertar el fondo en la ventana utilizaremos el método blit, llamando al objeto y dándole las cooredendas.

Este será el resultado:



Ahora temenos que instertar los meteoritos.

Vamos a diseñar la nave:



Vamos a eliminar los dos cuadrados.

Este serán las modificaciones:

```
import pygame

# Inicializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
NARANJA = (255, 128,0)
VERDE = (0, 255, 0)

ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

#  Imágenes
fondo = pygame.image.load("fondo.png")
asteroide_1A = pygame.image.load("asteroideA_1.png")
# Si el dibujo muestra los bordes blancos.
asteroide_1A.set_colorkey(BLANCO)
# los demas los hemos realizado con Krita
asteroide_2A = pygame.image.load("asteroideB_1.png")
nave_arriba = pygame.image.load("nave_arriba.png")

# Bucle principal
jugando = True
while jugando:

    reloj.tick(60)

    # Eventos
```

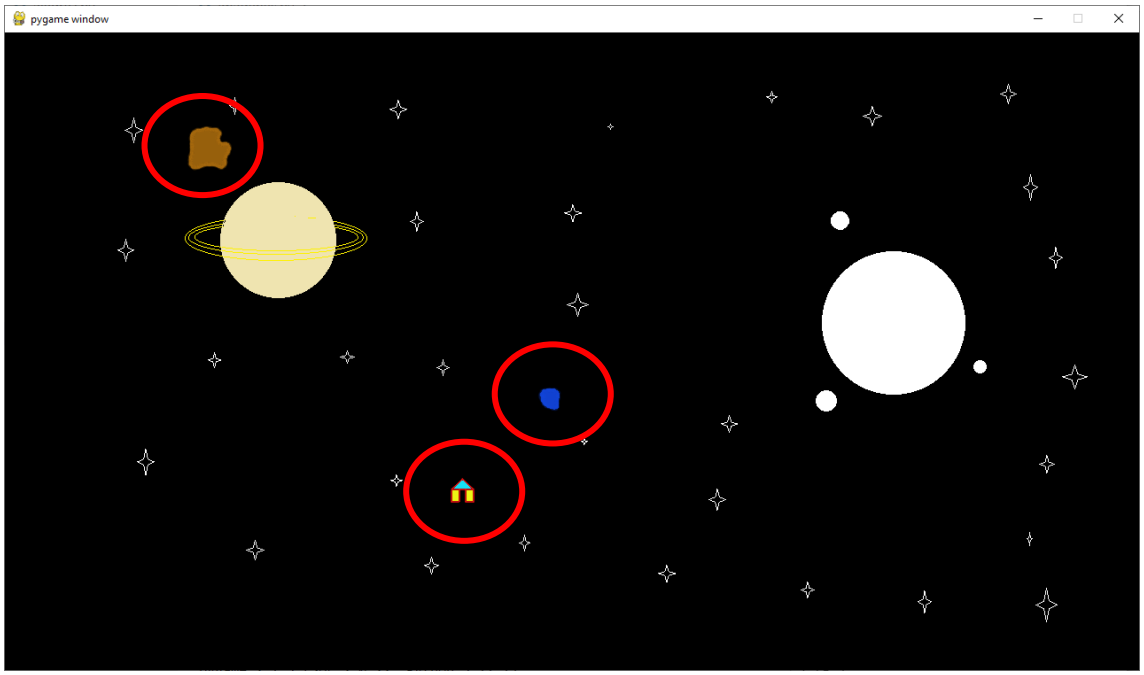
Con la variable fondo cargamos en memoria el archivo fondo.png.
Con la variable asteroide_A1 cargamos la imagen asteroidea_1.png.
Si las imágenes se han realizado con el Paint con la sentencia set_color.key le dedimos que color queremos eliminar.

Con la variable asteroide_A2 cargamos la imagen ateroideB_1.png y por último con la variable nave_arriba cargamos nave_arriba.png.

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                jugando = False


    # Imagenes
    ventana.blit(fondo,(0,0))
    ventana.blit(asteroide_1A, (200, 100))
    ventana.blit(asteroide_2A, (600, 400))
    ventana.blit(nave_arriba, (500, 500))

    pygame.display.update()

pygame.quit()
```

Cargamos el fondo y las correspondientes imágenes en la ventana, con sus correspondientes coordenadas.

Este será el resultado:



17.- Usar método convert con las imágenes en pygame

En el capitulo anterior vimos como cargar imagen y mostrarlas en la ventana de un juego.

```
19    #  Imágenes
20    fondo = pygame.image.load("fondo.png")
21    asteroide_1A = pygame.image.load("asteroideA_1.png")
22    # Si el dibujo muestra los bordes blancos.
23    asteroide_1A.set_colorkey(BLANCO)
24    # los demas los hemos realizado con Krita
25    asteroide_2A = pygame.image.load("asteroideB_1.png")
26    nave_arriba = pygame.image.load("nave_arriba.png")
```

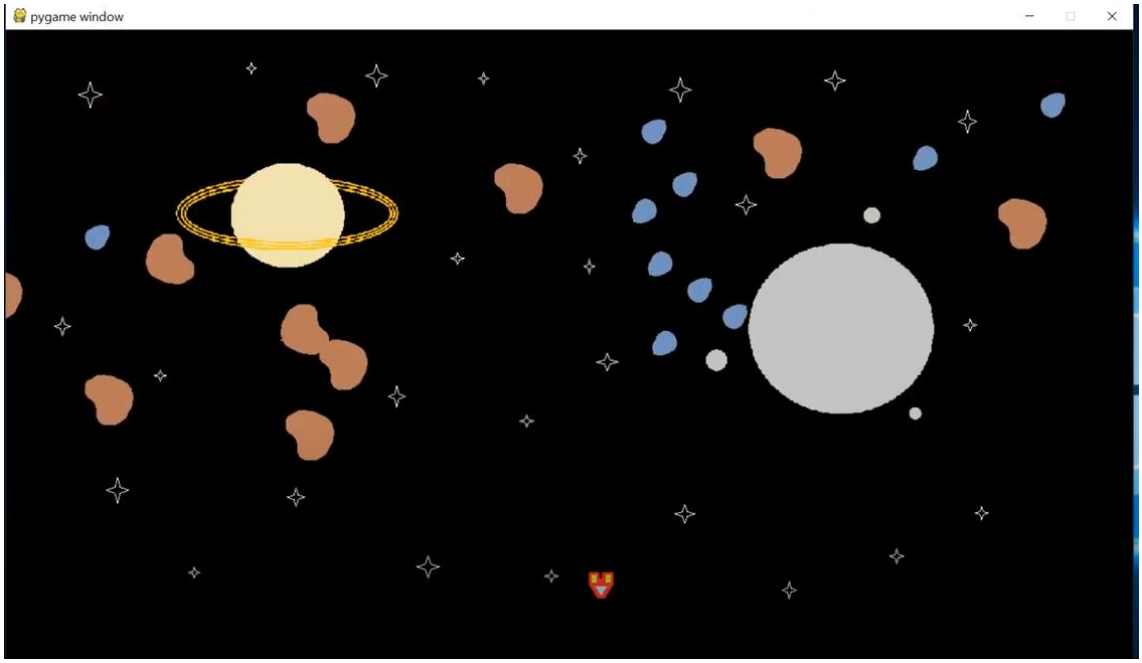
Lo que hicimos fue cargar una imagen para el fondo, mediante la función load e hicimos lo mismo para un asteroide grande y otro pequeño, además de la nave.

Si la imagen la realizamos con el Paint estas no tienen transparencias, tendremos que utilizar el método .set_colorkey(Color que se convierte a transparencia).

Las imágenes realizadas con Kryta permiten las transparencias, ello implica que nos podemos saltar este paso.

El resultado se muestra en la imagen anterior.

Para el próximo capítulo vamos utilizando esta plantilla para ir haciendo el siguiente juego:

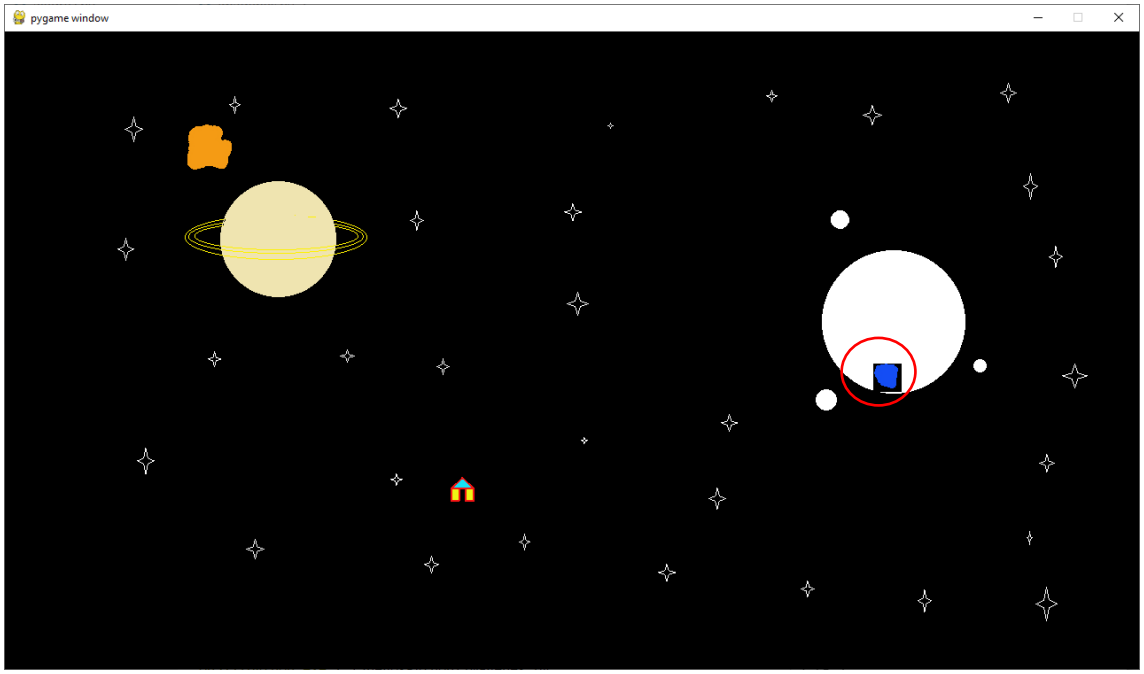


Una nave que podemos mover y que tiene que traspasar una serie de cinturones de asteroides, pero antes de llevarlo a cabo vamos a ver unos métodos que nos va a permitir dar más eficiencia a los juegos a la hora de poner las imágenes en la ventana y vamos a realizar algunas pruebas para ver hasta que punto la fluidez de los juegos depende de que la imagen contengan o no transparencia.

El método convert, que nos va a permitir y convertir el formato de píxeles con el que se guarda las imágenes en memoria de tal forma que luego para el interprete de Python sea más rápido mostrarlas en la ventana.

Como dice la documentación de pygame este va a ser el formato con el que más rapidez se van a colocar las imágenes. Por lo tanto es una buena idea convertir las superficies antes de que se vaya a poner en la ventana muchas veces, y también nos indica que si la superficie tiene pixeles alpha, tienen transparencia habrá que usar el método convert_alpha() para preservar esos pixeles.

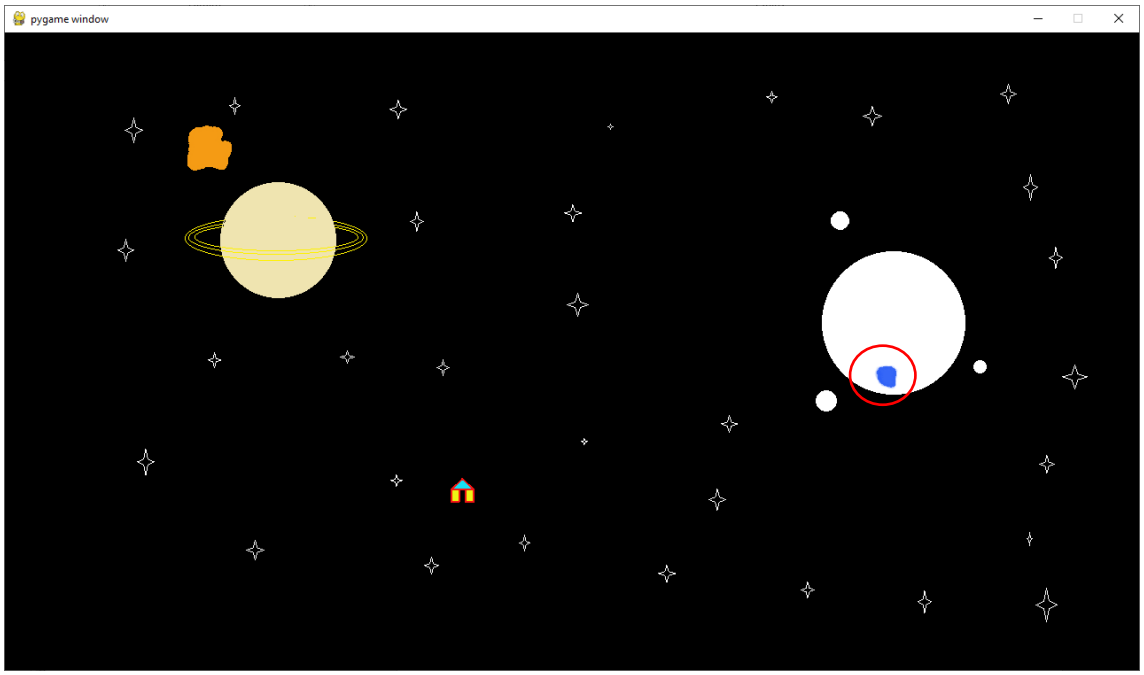
```
19    #  Imágenes
20    fondo = pygame.image.load("fondo.png").convert()
21    asteroide_1A = pygame.image.load("asteroideA_1.png").convert()
22    # Si el dibujo muestra los bordes blancos.
23    asteroide_1A.set_colorkey(BLANCO)
24    # los demas los hemos realizado con Krita
25    asteroide_2A = pygame.image.load("asteroideB_1.png").convert()
26    nave_arriba = pygame.image.load("nave_arriba.png").convert()
```



Las transparencias las convierte en color negro.

```
19    #  Imágenes
20    fondo = pygame.image.load("fondo.png").convert()
21    asteroide_1A = pygame.image.load("asteroideA_1.png").convert_alpha()
22    # Si el dibujo muestra los bordes blancos.
23    asteroide_1A.set_colorkey(BLANCO)
24    # los demas los hemos realizado con Krita
25    asteroide_2A = pygame.image.load("asteroideB_1.png").convert_alpha()
26    nave_arriba = pygame.image.load("nave_arriba.png").convert_alpha()
```

Este será el resultado:



	Sin método convert	Con método convert
Imágenes sin transparencia	40	80
Imágenes con transparencia	30	120

```
32      reloj.tick()
33      fps = reloj.get_fps()
```

Hemos quitado el límite de fps y a continuación vamos a comprobar los fps con el método ger_fps(), sin tener que hacer cálculos.

18.- Añadiendo figuras

Seguimos con el juego de los asteroides, vamos a ver como podemos poner distintos asteroides en al ventana del juego cada uno con su velocidad, cada uno con su orden de rotación y si la nave choca con uno se produzca la colisión.

Adjuntamos código:

```
import pygame
import random

# Inicializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)

# Funciones
def colision(x1, y1, a1, b1, x2, y2, a2, b2, ex=0):
    if  x1 + a1 > x2 + ex and \
        x1 + ex < x2 + a2 and \
        y1 + b1 > y2 + ex and \
        y1 + ex < y2 + b2:
        return True
    else:
        return False

ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()
fuente = pygame.font.SysFont("arial black", 20)

#  Carga Imágenes
fondo = pygame.image.load("fondo.png").convert()

nave_arr =
pygame.image.load("nave_arriba.png").convert_alpha()
nave_abj =
pygame.image.load("nave_abajo.png").convert_alpha()
```

Cuando llamemos a esta función con los parámetros (Ancho, Alto del arteroide, Ancho y alto de la nave, posición x y posición y de la nave, tamaño x e y de la nave, cuando choca la profundidad del choque.

```
nave_izq =
pygame.image.load("nave_izquierda.png").convert_alpha()
nave_der =
pygame.image.load("nave_derecha.png").convert_alpha()

aste_1a =
pygame.image.load("asteroideA_1.png").convert_alpha()
aste_1b =
pygame.image.load("asteroideA_2.png").convert_alpha()
aste_1c =
pygame.image.load("asteroideA_3.png").convert_alpha()
aste_1d =
pygame.image.load("asteroideA_4.png").convert_alpha()

aste_1 = [aste_1a, aste_1b, aste_1c, aste_1d]
aste_3 = [aste_1b, aste_1c, aste_1d, aste_1a]
aste_4 = [aste_1c, aste_1d, aste_1a, aste_1b]
aste_5 = [aste_1d, aste_1a, aste_1b, aste_1c]

asteroides_grandes = []

for i in range(10):
    x = random.randint(0, ANCHO)
    y = random.randint(50, ALTO-120)
    v = random.randint(1, 3)
    f = random.choice([aste_1, aste_3, aste_4, aste_5])
    a = [f, x, y, v]
    asteroides_grandes.append(a)

# Datos

vidas = 3
nivel = 1

nave_pos_x = 600
nave_pos_y = 670
nave_vel_x = 0
nave_vel_y = 0
direccion = "arriba"
frames_asteroides = 0

# Bucle principal
```

Cada asteroide realizará la rotación independiente.

Lista vacía

En el ciclo for se almacenan ancho, alto, velocidad posición de rotación del asteroide con valores aleatorios que al final se almacenan en la lista aterorides_grandes.

Un total de 10 asteroides.

Posición de la nave, con coordenadas x e y.

Velocidad en x como en y a 0.

Dirección mirando hacia arriba.

Frames = 0 (controlará la rotación del asteroide.

```
jugando = True
while jugando:
    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                jugando = False
            if event.key == pygame.K_RIGHT:
                direccion = "derecha"
                nave_vel_x = 2
            if event.key == pygame.K_LEFT:
                direccion = "izquierda"
                nave_vel_x = -2
            if event.key == pygame.K_DOWN:
                direccion = "abajo"
                nave_vel_y = 2
            if event.key == pygame.K_UP:
                direccion = "arriba"
                nave_vel_y = -2

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_RIGHT:
                nave_vel_x = 0
            if event.key == pygame.K_LEFT:
                nave_vel_x = 0
            if event.key == pygame.K_DOWN:
                nave_vel_y = 0
            if event.key == pygame.K_UP:
                nave_vel_y = 0
    # Lógica
    for a in asteroides_grandes:
        a[1]+= a[3]
        if a[1] > ANCHO:
            a[1] = -64

    nave_pos_x += nave_vel_x
    nave_pos_y += nave_vel_y
```

Este ciclo recorre todos los asteroides , suma la velocidad, es para que se vaya desplazando hacia la derecha.

Con el if controlamos que cuando llega al final del acho de al ventana retorne al principio.

Cuando presionamos las teclas de dirección, las variables nave_vel_x y nave_vel_y cambian sus valores.

```
if nave_pos_x > ANCHO - 32:
    nave_pos_x = ANCHO - 32
if nave_pos_x < 0:
    nave_pos_x = 0
if nave_pos_y > ALTO -32:
    nave_pos_y = ALTO - 32
```

Impide que la nave por la derecha, izquierda y abajo salga de la ventana.

```
    if nave_pos_y < 10:
        jugando = False
```

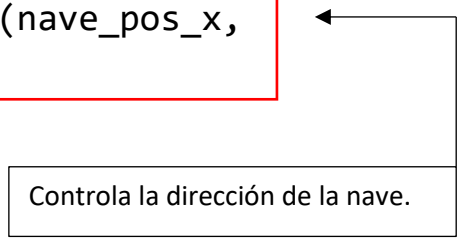
Controla si la nave toca a algún asteroide.

```
    for a in asteroides_grandes:
        if colision(a[1], a[2], 64, 64, nave_pos_x,
nave_pos_y, 32, 32, 15):
            jugando = False

    # Imagenes
    ventana.blit(fondo,(0,0))
    texto1 = fuente.render("Nivel: " + str(nivel),
True, BLANCO)
    texto2 = fuente.render("Vidas: " + str(vidas),
True, BLANCO)
    ventana.blit(texto1, (20,10))
    ventana.blit(texto2, (1150, 10))
```

Controla la rotación de los asteroides

```
    frames_asteroides += 1
    if frames_asteroides >= 41:
        frames_asteroides = 1

    if frames_asteroides < 11:
        for a in asteroides_grandes:
            ventana.blit(a[0][0], (a[1], a[2]))
    elif frames_asteroides < 21:
        for a in asteroides_grandes:
            ventana.blit(a[0][1], (a[1], a[2]))
    elif frames_asteroides < 31:
        for a in asteroides_grandes:
            ventana.blit(a[0][2], (a[1], a[2]))
    elif frames_asteroides < 41:
        for a in asteroides_grandes:
            ventana.blit(a[0][3], (a[1], a[2]))
```

```
    if direccion == "arriba":
        ventana.blit(nave_arr, (nave_pos_x,
nave_pos_y))
    if direccion == "abajo":
        ventana.blit(nave_abj, (nave_pos_x,
nave_pos_y))
    if direccion == "izquierda":
        ventana.blit(nave_izq, (nave_pos_x,
nave_pos_y))
    if direccion == "derecha":
        ventana.blit(nave_der, (nave_pos_x,
nave_pos_y))

    pygame.display.update()

pygame.quit()
```

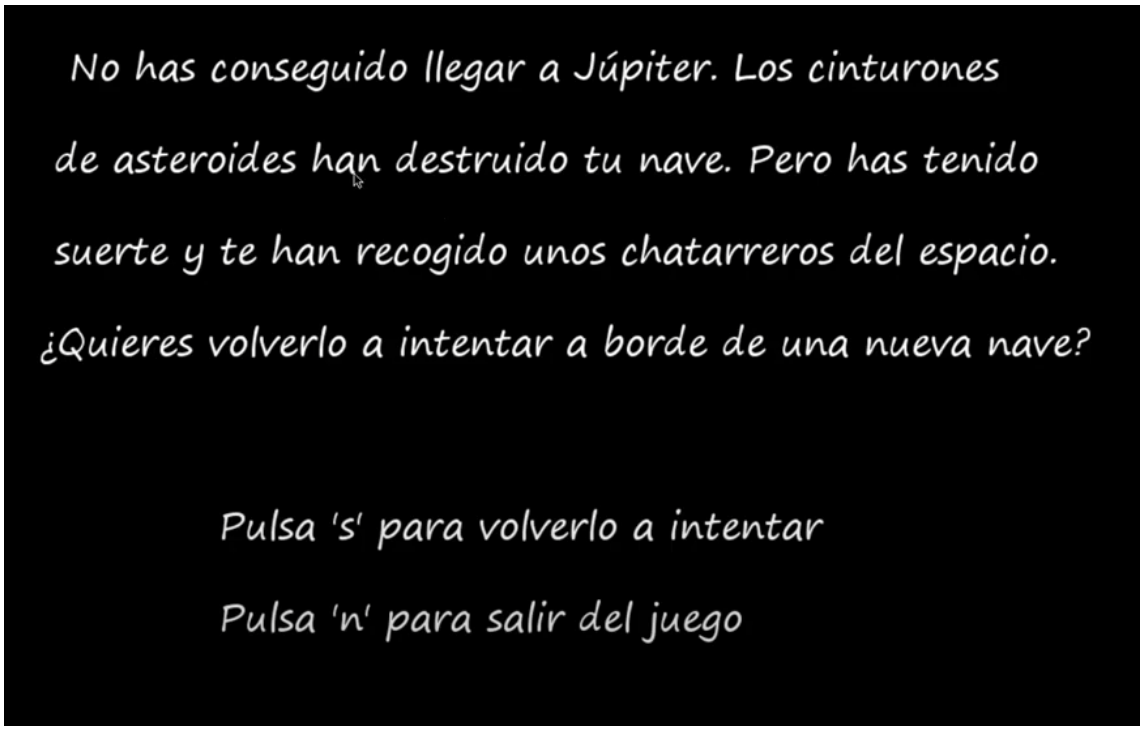


19.- Poniendo niveles

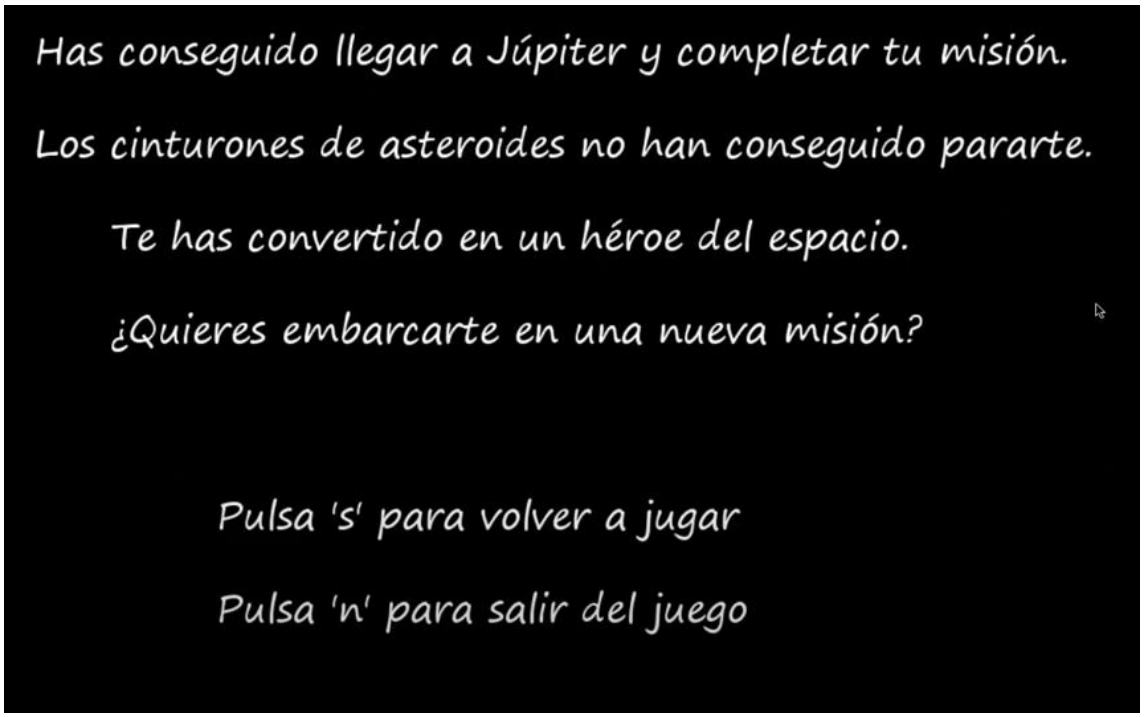
En este capítulo vamos a implementar una ventana de introducción y varios niveles de juego.

Cuando ejecutamos el programa nos aparece la pantalla de inicio

Pantalla final cuando pierdes.



Pantalla final habiendo ganado el juego:



Antes de seguir con el juego vamos a ver un esquema para entender un poco la lógica del juego, antes de ver el código.

```
ESQUEMA DEL JUEGO ASTEROIDES CON NIVELES

while en_juego:

    (while en_inicio)

    while en_partida:

        (while en_nivel)

        (while en_final)
```

Vamos a tener dos bucles lógicos y dentro de los bucles lógicos para mantener las ventanas de pygame.

Una vez empezamos el juego entramos en while en_juego donde entraremos a un segundo bucle llamado while en_inicio donde veremos la presentación y a continuación entraremos en un tercer bucle donde empieza la partida, donde entramos a otro bucle para los niveles del juego y por último el bucle final.

A continuación vamos a crear un nuevo archivo llamado boceto_bucle.py y escribiremos el siguiente código:

```
en_juego = True
while en_juego:
    num_vidas = 3
    num_nivel = 1

    en_partida = False
    en_inicio = True

    while en_inicio:
        # Evento Entrar: en_partida = True
        # Evento Cerrar ventana:
            # en_inicio = False
            # en_juego = False
            pass
    while en_partida:
        en_final = False
        # Se cran los asteroides según el nivel

        en_nivel = True
        while en_nivel:

            # Evento Cerrar ventana:
                # en_nivel = Falsa
                # en_partida = False
                # en_juego = False
            # Eventos Moverse

            if # colision:
                num_vidas -= 1
                en_nivel = False

            if # llegar arriba:
                num_nivel += 1
                en_nivel = False

            if num_vidas == 0:
                en_final = True

            if num_nivel > 3:
                en_final = True

        while en_final:
```

```
            # Evento Cerrar ventana o no jugar más:
                 # en_final = False
                 # en_partida = False
                # en_juego = False
            # Evento jugar de nuevo:
                # en_final = False
                # en_partida = False
            if ganando:
                # Mensaje: Se ha ganado
            else:
                # Mensaje: Se ha perdido
```

Esta es la estructura donde tiene que ir el juego.

Ahora vamos a implementar los bucles al código del ejercicio anterior.

```
import pygame
import random

# Inicializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)


# Funciones

def colision(x1, y1, a1, b1, x2, y2, a2, b2, ex=0):
    if x1 + a1 > x2 + ex and \
       x1 + ex < x2 + a2 and \
       y1 + b1 > y2 + ex and \
       y1 + ex < y2 + b2:
        return True
    else:
        return False
```

```
# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()
fuente1 = pygame.font.SysFont("arial black", 24)
fuente2 = pygame.font.SysFont("Segoe print", 32)

# Carga imágenes
fondo = pygame.image.load("fondo.png").convert()

nave_arr =
pygame.image.load("nave_arriba.png").convert_alpha()
nave_abj =
pygame.image.load("nave_abajo.png").convert_alpha()
nave_izq =
pygame.image.load("nave_izquierda.png").convert_alpha()
nave_der =
pygame.image.load("nave_derecha.png").convert_alpha()

aste_1a =
pygame.image.load("asteroideA_1.png").convert_alpha()
aste_1b =
pygame.image.load("asteroideA_2.png").convert_alpha()
aste_1c =
pygame.image.load("asteroideA_3.png").convert_alpha()
aste_1d =
pygame.image.load("asteroideA_4.png").convert_alpha()

aste_1 = [aste_1a, aste_1b, aste_1c, aste_1d]
aste_3 = [aste_1b, aste_1c, aste_1d, aste_1a]
aste_4 = [aste_1c, aste_1d, aste_1a, aste_1b]
aste_5 = [aste_1d, aste_1a, aste_1b, aste_1c]

aste_2a =
pygame.image.load("asteroideB_1.png").convert_alpha()
aste_2b =
pygame.image.load("asteroideB_2.png").convert_alpha()
aste_2c =
pygame.image.load("asteroideB_3.png").convert_alpha()
aste_2d =
pygame.image.load("asteroideB_4.png").convert_alpha()

aste_2 = [aste_2a, aste_2b, aste_2c, aste_2d]
```

```
aste_6 = [aste_2b, aste_2c, aste_2d, aste_2a]
aste_7 = [aste_2c, aste_2d, aste_2a, aste_2b]
aste_8 = [aste_2d, aste_2a, aste_2b, aste_2c]


# Bucle principal

en_juego = True
while en_juego:

    num_vidas = 3
    num_nivel = 1

    en_partida = False
    en_inicio = True

    while en_inicio:

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                en_inicio = False
                en_juego = False

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_RETURN:
                    en_inicio = False
                    en_partida = True

        ventana.fill(NEGRO)

        historia = [
            "                              MISIÓN A JUPITER",
            "     Tras meses de viaje en misión a
Júpiter ya sólo se",
            "    interponen en tu camino tres
cinturones de asteroides",
            "  que te dificultan el paso y amenazan con
no dejarte pasar.",
            "   Intenta traspasarlos para conseguir
finalizar tu misión.",
            "",
```

Variable en_juego de tipo booleana valor True

El bucle en_juego

Inicializamos las variables num:_vidas a 3 y num nivel a 1

Inicializamos las variables en_partida a False u en_incio a True

Bucle en_inicio

Evento cerrar ventana

En_inicio igual a False
en_juego igual a False

Evento entrar partida igual a True.

```
            "                    Pulsa 'ENTER' para
empezar"
            ]

        y = 80
        for frase in historia:
            texto = fuente2.render(frase, True, BLANCO)
            ventana.blit(texto, (150, y))
            y += 80

        pygame.display.update()
```

```
while en_partida:
```
← Bucle en_partida

```
en_final = False
```
← En_final igual a False

Se crean los asteroides. ↓

```
        num_aste_grandes = num_nivel * 5
        num_aste_pequenios = num_nivel * 10

        asteroides_grandes = []
        asteroides_pequenios = []

        for i in range(num_aste_grandes):
            x = random.randint(0, ANCHO)
            y = random.randint(50, ALTO-120)
            v = random.randint(1, 3)
            f = random.choice([aste_1, aste_3, aste_4,
aste_5])
            a = [f, x, y, v]
            asteroides_grandes.append(a)

        for i in range(num_aste_pequenios):
            x = random.randint(0, ANCHO)
            y = random.randint(50, ALTO-120)
            v = random.randint(1, 4)
            f = random.choice([aste_2, aste_6, aste_7,
aste_8])
            a = [f, x, y, v]
            asteroides_pequenios.append(a)
```

```
        asteroides = asteroides_grandes +
asteroides_pequenios
```

```
nave_pos_x = 600
nave_pos_y = 670
nave_vel_x = 0
nave_vel_y = 0
direccion = "arriba"

frames_asteroides = 0
```

Variable en_nivel igual a True

```
en_nivel = True
while en_nivel:
```

Bucle en_nivel

```
    reloj.tick(60)
```

Evento cerrar ventana

```
    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            en_nivel = False
            en_partida = False
            en_juego = False
```

Variables en_nivel, en_partida y en_juego a False.

Eventos de moverse

```
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_RIGHT:
            direccion = "derecha"
            nave_vel_x = 2
        if event.key == pygame.K_LEFT:
            direccion = "izquierda"
            nave_vel_x = -2
        if event.key == pygame.K_DOWN:
            direccion = "abajo"
            nave_vel_y = 2
        if event.key == pygame.K_UP:
            direccion = "arriba"
            nave_vel_y = -2

    if event.type == pygame.KEYUP:
        if event.key == pygame.K_RIGHT:
            nave_vel_x = 0
        if event.key == pygame.K_LEFT:
            nave_vel_x = 0
```

```
            if event.key == pygame.K_DOWN:
                nave_vel_y = 0
            if event.key == pygame.K_UP:
                nave_vel_y = 0

        # Lógica

        for a in asteroides_grandes:
            a[1] += a[3]
            if a[1] > ANCHO:
                a[1] = -64

        for a in asteroides_pequenios:
            a[1] += a[3]
            if a[1] > ANCHO:
                a[1] = -32


        nave_pos_x += nave_vel_x
        nave_pos_y += nave_vel_y

        if nave_pos_x > ANCHO - 32:
            nave_pos_x = ANCHO - 32
        if nave_pos_x < 0:
            nave_pos_x = 0
        if nave_pos_y > ALTO - 32:
            nave_pos_y = ALTO - 32

        if nave_pos_y < 10:
            en_nivel = False
            num_nivel += 1
```

Llegar arriba

Colisión

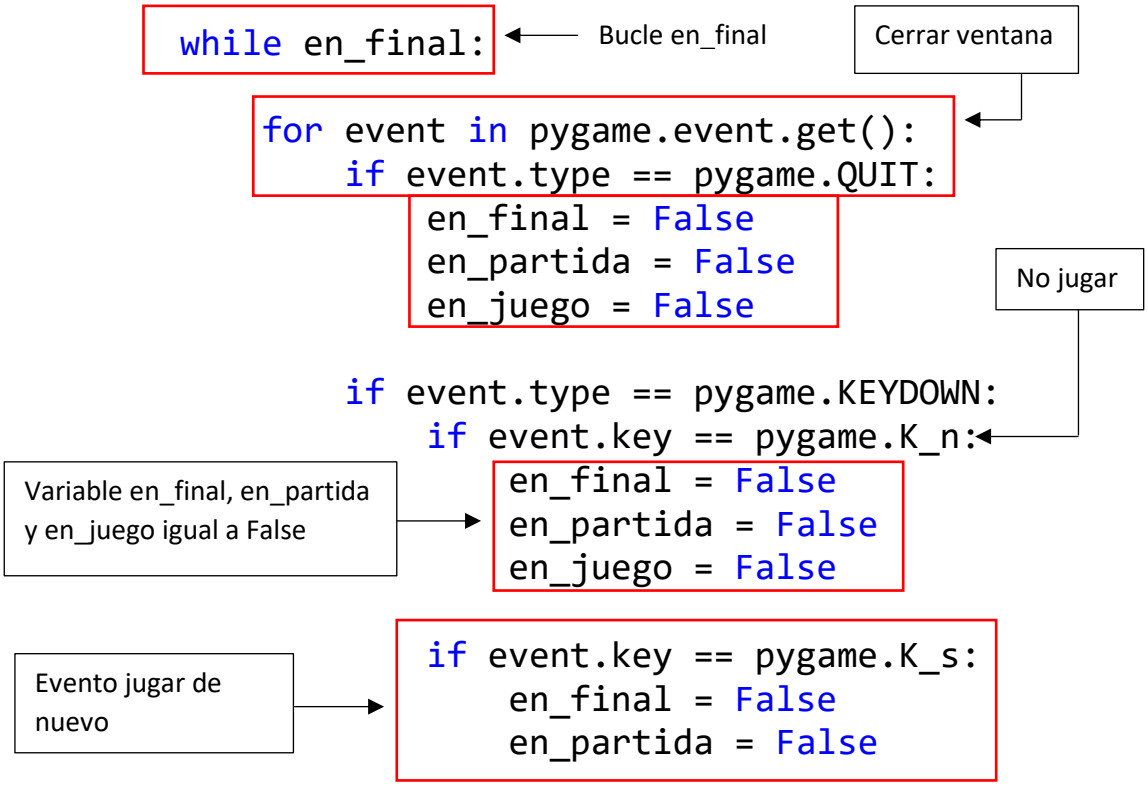
```
        for a in asteroides_grandes:
            if colision(a[1], a[2], 64, 64,
nave_pos_x, nave_pos_y, 32, 32, 15):
                en_nivel = False
                num_vidas -= 1

        for a in asteroides_pequenios:
            if colision(a[1], a[2], 32, 32,
nave_pos_x, nave_pos_y, 32, 32, 15):
```

```
                en_nivel = False
                num_vidas -= 1

            if num_vidas == 0:
                ganando = False
                en_final = True

            if num_nivel > 3:
                ganando = True
                en_final = True

            # Imágenes

            ventana.blit(fondo, (0,0))

            texto1 = fuente1.render("Nivel: " +
str(num_nivel), True, BLANCO)
            texto2 = fuente1.render("Vidas: " +
str(num_vidas), True, BLANCO)
            ventana.blit(texto1, (20,10))
            ventana.blit(texto2, (1150, 10))


            frames_asteroides += 1
            if frames_asteroides >= 41:
                frames_asteroides = 1

            if frames_asteroides < 11:
                for a in asteroides:
                    ventana.blit(a[0][0], (a[1], a[2]))
            elif frames_asteroides < 21:
                for a in asteroides:
                    ventana.blit(a[0][1], (a[1], a[2]))
            elif frames_asteroides < 31:
                for a in asteroides:
                    ventana.blit(a[0][2], (a[1], a[2]))
            elif frames_asteroides < 41:
                for a in asteroides:
                    ventana.blit(a[0][3], (a[1], a[2]))

            if direccion == "arriba":
```

Número de vidas

Número de nivel

```
                ventana.blit(nave_arr, (nave_pos_x,
nave_pos_y))
            elif direccion == "abajo":
                ventana.blit(nave_abj, (nave_pos_x,
nave_pos_y))
            elif direccion == "izquierda":
                ventana.blit(nave_izq, (nave_pos_x,
nave_pos_y))
            elif direccion == "derecha":
                ventana.blit(nave_der, (nave_pos_x,
nave_pos_y))


            pygame.display.update()
```




```
            ventana.fill(NEGRO)

            historia_perdido = [
                "    No has conseguido llegar a
Júpiter. Los cinturones",
                "   de asteroides han destruido tu
nave. Pero has tenido",
```

```
                "   suerte y te han recogido unos
chatarreros del espacio.",
                "  ¿Quieres volverlo a intentar a borde
de una nueva nave?",
                "",
                "                Pulsa 's' para volverlo
a intentar",
                "                Pulsa 'n' para salir del
juego"]

            historia_ganado = [
                "  Has conseguido llegar a Júpiter y
completar tu misión.",
                "  Los cinturones de asteroides no han
conseguido pararte.",
                "       Te has convertido en un héroe
del espacio.",
                "       ¿Quieres embarcarte en una
nueva misión?",
                "",
                "                Pulsa 's' para volver a
jugar",
                "                Pulsa 'n' para salir del
juego"]

            if ganando:
                historia = historia_ganado
            else:
                historia = historia_perdido

            y = 80
            for frase in historia:
                texto = fuente2.render(frase, True,
BLANCO)
                ventana.blit(texto, (150, y))
                y += 80

            pygame.display.update()


pygame.quit()
```

Mensaje que se muestra cuando has ganado o cuando has perdido.

20.- Clase Rect en pygame

Hasta ahora hemos visto las características más básicas de pygame, la herramientas más elementales.

Pygame dispone de herramientas más avanzadas, más optimizadas para poder llevar al cabo juegos de una forma más eficiente, entre estas herramientas están algunas clases que vienen predefinidas en la librería pygame, como la clase que vamos a ver en este capítulo.
La clase Rect ya la tenemos predefinida en la librería pygame por lo tanto será únicamente usarla.

	Atributos	(Datos)
Clase:	Métodos	(Funciones)
	r1.x = 10	(Atributos x)
r1 = Rect()	r1.x = 20	
	r1.colliderect(r2)	(Mletodo Colliderect)
22 = Rect()		

Vamos a ver mucho la programación orientada a objeto, pero si debemos de tener una nociones como que las clases constan de atributos y métodos, y para acceder a los atributos o llamar a los métodos hemos de usar la notación del punto.

```
1   # SOLO VAMOS A IMPORTAR LA CLASE RECT
2   from pygame import Rect
3
4   mi_rectangulo = Rect(400, 300, 50, 50)
5   print(mi_rectangulo.x)
6   print(mi_rectangulo.y)
7   print(mi_rectangulo)
8
9   mi_rectangulo.x += 100
10  print(mi_rectangulo)
11
12  mi_rectangulo.width += 10
13  mi_rectangulo.height += 10
14  print(mi_rectangulo)
```

Este será el resultado:

```
400
300
<rect(400, 300, 50, 50)>
<rect(500, 300, 50, 50)>
<rect(500, 300, 60, 60)>
```

Vamos con otro ejemplo:

Vamos a agregar las siguientes líneas:

```
16    otro_rectangulo = Rect(561, 300, 50, 50)
17    print(mi_rectangulo.colliderect(otro_rectangulo))
18    mi_rectangulo.x += 5
19    print(mi_rectangulo.colliderect(otro_rectangulo))
```

Este será el resultado:

```
400
300
<rect(400, 300, 50, 50)>
<rect(500, 300, 50, 50)>
<rect(500, 300, 60, 60)>
False
True
```

En este caso los dos rectángulos han colisionado.

Partiendo de esta platilla:

```
import pygame

# Inicializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Colores
NEGRO = (0, 0, 0)
AZUL = (0, 0, 255)

# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()
```

```
# Datos
nave_pos_x = 400
nave_pos_y = 500
nave_vel_x = 0
nave_vel_y = 0

# Bucle principal
jugando = True
while jugando:

    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                nave_vel_x = 5
            if event.key == pygame.K_LEFT:
                nave_vel_x = -5
            if event.key == pygame.K_DOWN:
                nave_vel_y = 5
            if event.key == pygame.K_UP:
                nave_vel_y = -5

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_RIGHT:
                nave_vel_x = 0
            if event.key == pygame.K_LEFT:
                nave_vel_x = 0
            if event.key == pygame.K_DOWN:
                nave_vel_y = 0
            if event.key == pygame.K_UP:
                nave_vel_y = 0

    # Lógica
    nave_pos_x += nave_vel_x
    nave_pos_y += nave_vel_y

    if nave_pos_x > ANCHO - 50:
        nave_pos_x = ANCHO - 50
```

```
    if nave_pos_x < 0:
        nave_pos_x = 0
    if nave_pos_y > ALTO - 50:
        nave_pos_y = ALTO - 50
    if nave_pos_y < 0:
        nave_pos_y = 0

    # Dibujos
    ventana.fill(NEGRO)
    pygame.draw.rect(ventana, AZUL,
(nave_pos_x,nave_pos_y, 50, 50))

    # Actualizar
    pygame.display.update()

# Salir
pygame.quit()
```

Ahora vamos a cambiarlo por la clase Rect.

```
18   # Datos
19   nave = pygame.Rect(600, 500, 50, 50)
20   nave_vel_x = 0
21   nave_vel_y = 0
```

En datos creamos el objeto nave, y los valores para el alto y ancho ya no son necesarios.

```
53       # Lógica
54       nave.x += nave_vel_x
55       nave.y += nave_vel_y
```

Al trabajar con una clase utilizamos el método del punto.

```
57       if nave.x > ANCHO - nave.width:
58           nave.x = ANCHO - nave.width
59       if nave.x < 0:
60           nave.x = 0
61       if nave.y > ALTO - nave.height:
62           nave.y = ALTO - nave.height
63       if nave.y < 0:
64           nave.y = 0
```

Lo mismo.

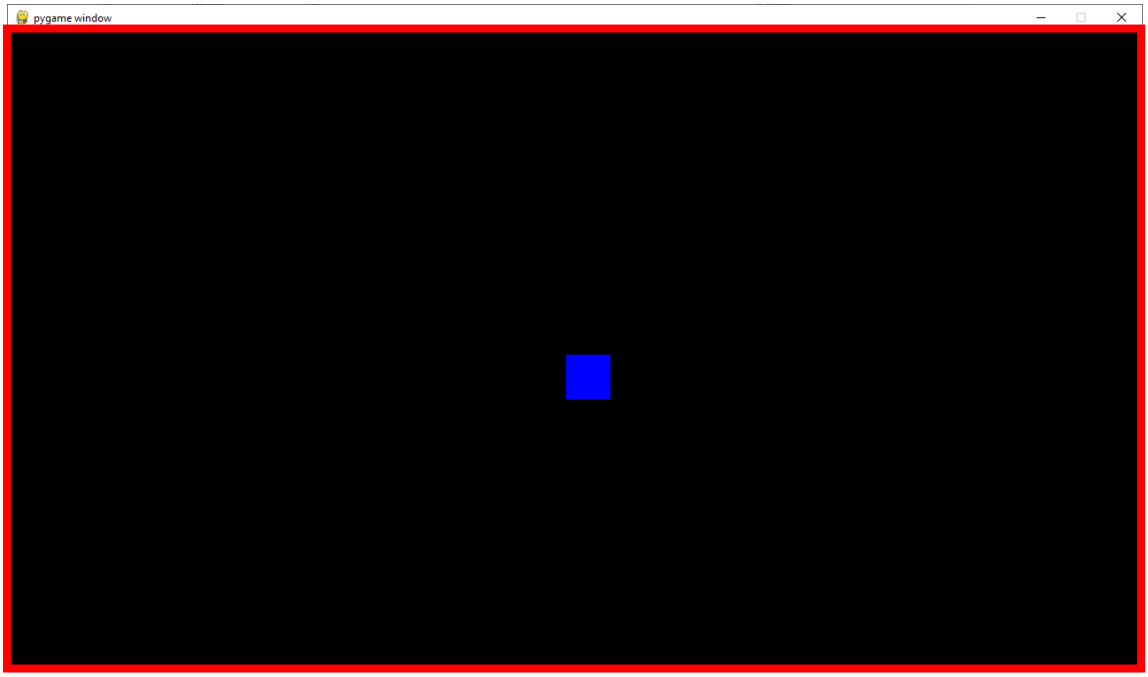
```
66      # Dibujos
67      ventana.fill(NEGRO)
68      pygame.draw.rect(ventana, AZUL, nave)
```

Al dibujar el rectángulo poniendo nave ya sabe las coordenadas y la dimensiones.

Antes necesitábamos :

```
pygame.draw.rect(ventana, AZUL, (nave_pos_x,nave_pos_y, 50, 50))
```

Le damos a ejecutar:



Ya nos podemos mover en todas las dirección, pero sin salir de la ventana, cuando llega a un extremo de ella el cuadrado no sigue.

Te propongo que realices el siguiente reto:

Añadir a este programa otro objeto de la clase Rect que pueda ser una roca y se sitúa en el centro de la ventana, de tal forma que si la nave colisiona con la roca coloque la nave en sus coordenadas iniciales y se trata de utilizar el método de la clase Rect colliderect().

21.- Método colliderect de la clase Rect.

En el apartado de Datos creamos un nuevo objeto llamado roca.

```
18    # Datos
19    roca = pygame.Rect(500, 200, 300, 100)
20    nave = pygame.Rect(600, 500, 50, 50)
21    nave_vel_x = 0
22    nave_vel_y = 0
```

Vamos a agregar el color VERDE.

```
10    # Colores
11    NEGRO = (0, 0, 0)
12    AZUL = (0, 0, 255)
13    VERDE =(0, 255, 0)
```

En el apartado lógica controlamos la colisión.

```
56        # Lógica
57        nave.x += nave_vel_x
58        nave.y += nave_vel_y
59
60        if nave.x > ANCHO - nave.width:
61            nave.x = ANCHO - nave.width
62        if nave.x < 0:
63            nave.x = 0
64        if nave.y > ALTO - nave.height:
65            nave.y = ALTO - nave.height
66        if nave.y < 0:
67            nave.y = 0
68
69        if nave.colliderect(roca):
70            pygame.time.delay(1000)
71            nave.x = 600
72            nave.y = 500
```

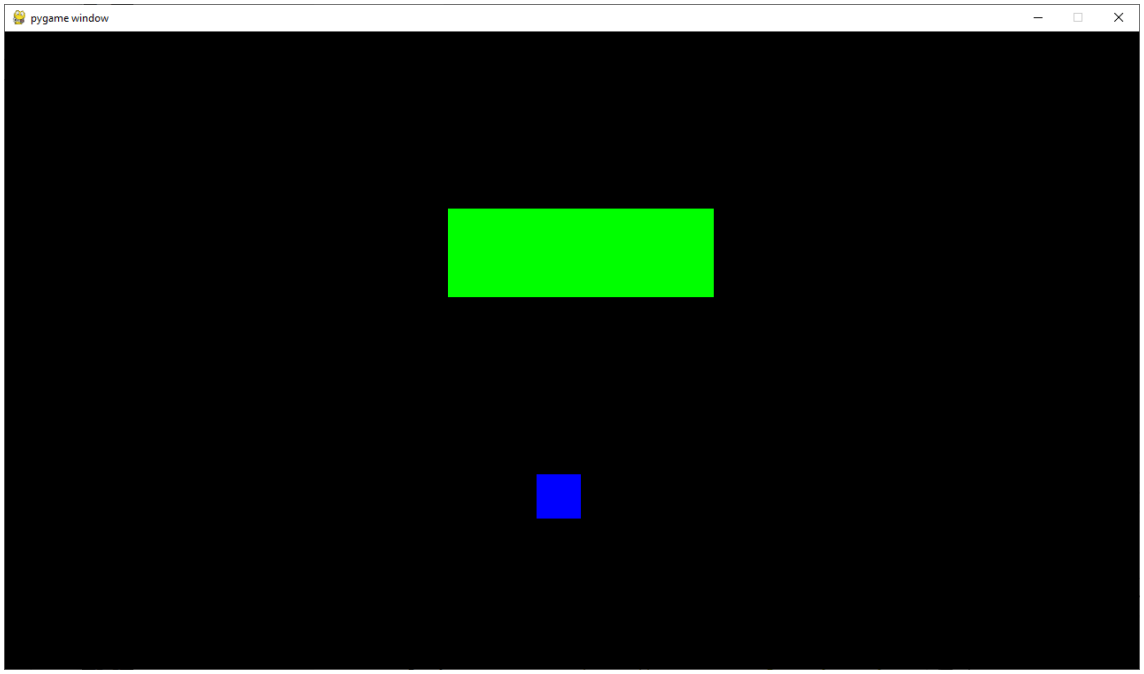
Para hacer la simulación de un muero donde el personaje no puede traspasar el muro.

```
69        if nave.colliderect(roca):
70            nave.x -= nave_vel_x
71            nave.y -= nave_vel_y
```

Las variables nave_vel_x y nave_vel_y son las que utilizamos para mover nuestro objeto.

De esta forma este método nos va a permitir implementar modos o paredes, caminos que limiten el movimiento de los objetos en los juegos.

Para el próximo capitulo vamos a poner 4 muros en uno de los lados de la ventana y que el personaje al colisionar con cualquiera de ellos se pare frente al muro y no pueda avanzar.



22.- Muros con pygame

En el capítulo anterior hemos visto cómo podemos utilizar el método colliderect() de la clase Rect para implementar muros en los juegos.

En capítulo anterior creamos un objeto de la clase Rect una nave y otro roca y si la nave colisionaba con la roca, volvíamos a restar la velocidad a la nave para que no se movise, se quedase en el mismo sitio.

Vamos a empezar:

Siguiendo con el proyecto anterior vamos a agregar las siguientes funciones:

```
15  # Funciones
16  def dibujar_muro(superficie, rectangulo):
17      pygame.draw.rect(superficie, VERDE, rectangulo)
18
19  def dibujar_personaje(superficie, rectangulo):
20      pygame.draw.rect(superficie, AZUL, rectangulo)
```

Creamos los muros:

```
26  # Datos
27  muros = [
28      pygame.Rect(500, 100, 300, 100),
29      pygame.Rect(200, 200, 100, 300),
30      pygame.Rect(500, 500, 300, 100),
31      pygame.Rect(1000, 200, 100, 300)]
32
```

Ahor agregamos el personaje:

```
26  # Datos
27  muros = [
28      pygame.Rect(500, 100, 300, 100),
29      pygame.Rect(200, 200, 100, 300),
30      pygame.Rect(500, 500, 300, 100),
31      pygame.Rect(1000, 200, 100, 300)]
32
33  personaje = pygame.Rect(600, 400, 50, 50)
34  personaje_vel_x = 0
35  personaje_vel_y = 0
```

Elimina los objetos de nave y roca.

```
    # Lógica
    personaje.x += personaje_vel_x
    personaje.y += personaje_vel_y

    if personaje.x > ANCHO - personaje.width:
        personaje.x = ANCHO - personaje.width
    if personaje.x < 0:
        personaje.x = 0
    if personaje.y > ALTO - personaje.height:
        personaje.y = ALTO - personaje.height
    if personaje.y < 0:
        personaje.y = 0

```

Lo adaptamos para el nuevo personaje.

```
    for muro in muros:
        if personaje.colliderect(muro):
            personaje.x -= personaje_vel_x
            personaje.y -= personaje_vel_y
```

Con el siguiente código controlamos que nuestro personaje no teque ninguno de los muros.

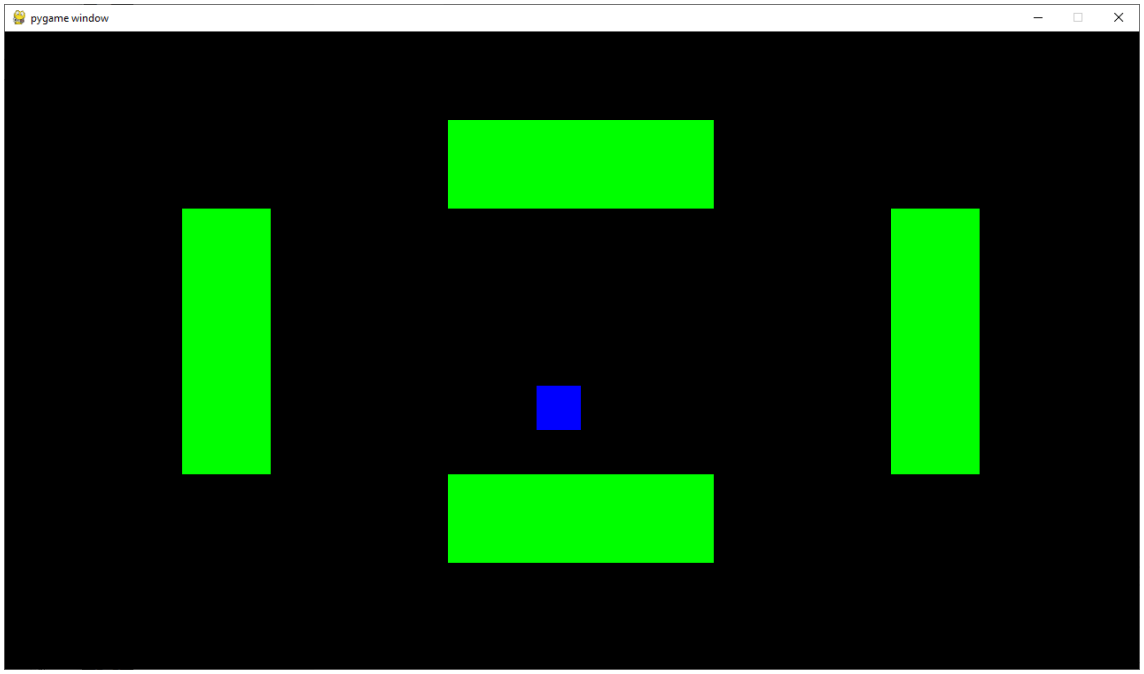
```
    # Dibujos
    ventana.fill(NEGRO)

    for muro in muros:
        dibujar_muro(ventana, muro)

    dibujar_personaje(ventana, personaje)
```

Dibujamos los muros y el personaje.

```
    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                personaje_vel_x = 5
            if event.key == pygame.K_LEFT:
                personaje_vel_x = -5
```

```
58              if event.key == pygame.K_DOWN:
59                  personaje_vel_y = 5
60              if event.key == pygame.K_UP:
61                  personaje_vel_y = -5
62
63          if event.type == pygame.KEYUP:
64              if event.key == pygame.K_RIGHT:
65                  personaje_vel_x = 0
66              if event.key == pygame.K_LEFT:
67                  personaje_vel_x= 0
68              if event.key == pygame.K_DOWN:
69                  personaje_vel_y = 0
70              if event.key == pygame.K_UP:
71                  personaje_vel_y = 0
```

Cambiamos la variable nave_vel_ por personaje_vel_ en todo este bloque de código.



Un detalle a tener en cuenta, tanto la velocidad del personaje que es 5 como la posición de los muros que todos los valores son múltiplos de 5 hace que la colisión sea perfecta, si cambiásemos la velocidad de nuestro personaje a 7 como la ubicación de los muros no serían múltiplos de 7 podemos observar que hay una pequeña separación entre el muro y el personaje.

Para evitar estas separaciones vamos a realizar las siguientes modificaciones:

```
49      # Eventos
50      for event in pygame.event.get():
51          if event.type == pygame.QUIT:
52              jugando = False
```

```
53          if event.type == pygame.KEYDOWN:
54              if event.key == pygame.K_RIGHT:
55                  direccion ="derecha"
56                  personaje_vel_x = 5
57              if event.key == pygame.K_LEFT:
58                  direccion = "izquierda"
59                  personaje_vel_x = -5
60              if event.key == pygame.K_DOWN:
61                  direccion = "abajo"
62                  personaje_vel_y = 5
63              if event.key == pygame.K_UP:
64                  direccion = "arriba"
65                  personaje_vel_y = -5
```

En la variable dirección almacenamos al tecla que estamos presionando.

```
90      for muro in muros:
91          if personaje.colliderect(muro):
92              if direccion == "derecha":
93                  personaje.right = muro.left
94              elif direccion == "izquierda":
95                  personaje.left = muro.right
96              elif direccion == "abajo":
97                  personaje.bottom = muro.top
98              elif direccion == "arriba":
99                  personaje.top = muro.bottom
```

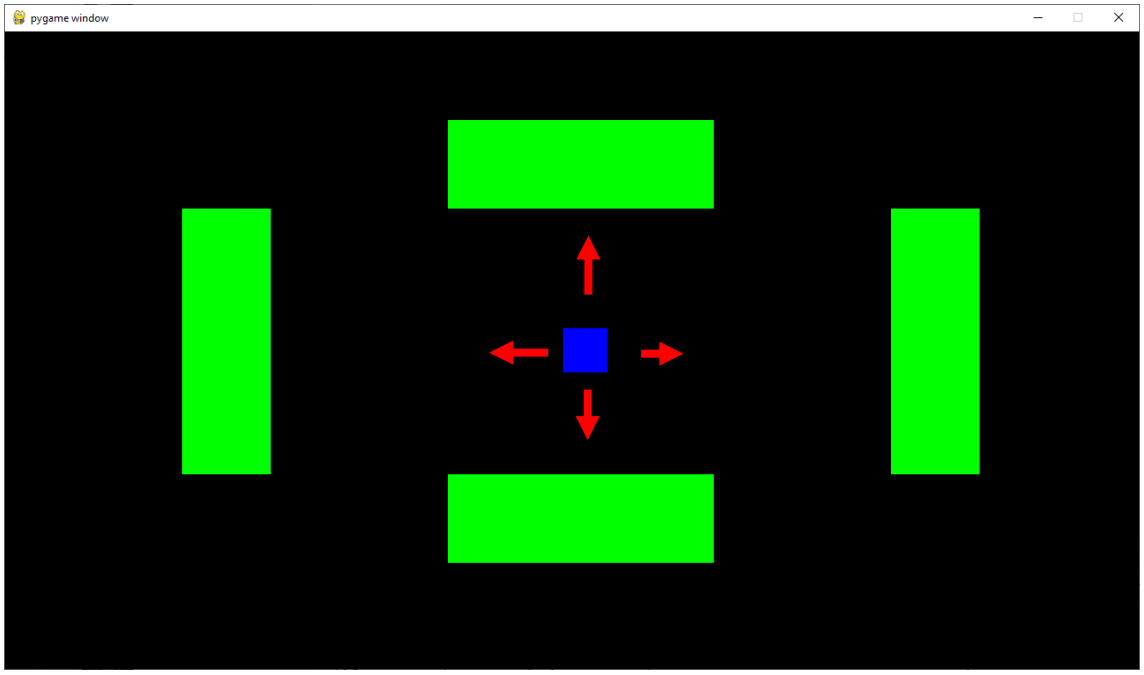
En el apartado de lógica según en la dirección que vaya nos detendremos en la parte opuesta del muro.

Pero no controlamos cuando pulsamos simultáneamente dos teclas para ir en diagonal.

```
49      # Eventos
50      for event in pygame.event.get():
51          if event.type == pygame.QUIT:
52              jugando = False
53          if event.type == pygame.KEYDOWN:
54              if event.key == pygame.K_RIGHT:
55                  direccion ="derecha"
56                  personaje_vel_x = 5
57                  personaje_vel_y = 0
58              if event.key == pygame.K_LEFT:
59                  direccion = "izquierda"
```

```
60                personaje_vel_x = -5
61                personaje_vel_y = 0
62            if event.key == pygame.K_DOWN:
63                direccion = "abajo"
64                personaje_vel_y = 5
65                personaje_vel_x = 0
66            if event.key == pygame.K_UP:
67                direccion = "arriba"
68                personaje_vel_y = -5
69                personaje_vel_x = 0
```

Ya no podemos hacer movimientos en diagonal.



23.- Mapas con pygame

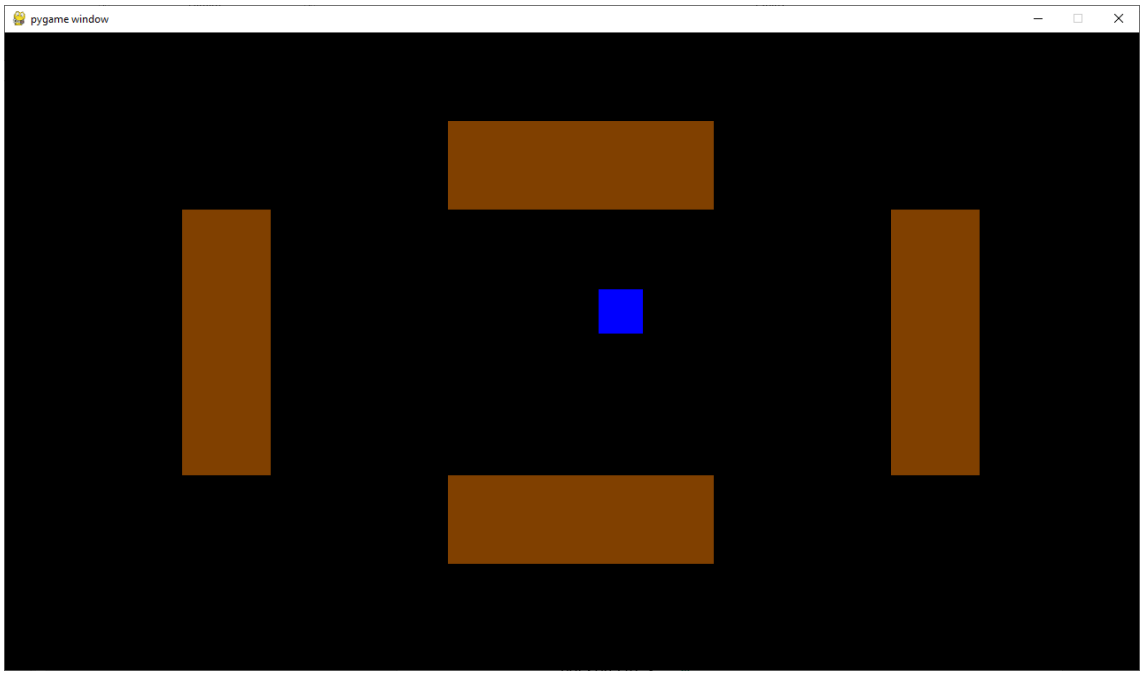
En el capítulo anterior hemos visto como poner muros en la ventana de un juego pero puede que queramos hacer un juego de plataformas o que contenga muchas habitaciones y que por lo tanto necesitemos un método mejor y más rápido para llevarlo a cabo, vamos a ver como lo podemos hacer mediante mapas.

Vamos a partir de la plantilla del capítulo anterior en la que simplemente tenemos dibujados 4 muros y tenemos un personaje que se puede ir moviendo por la ventana a la que le hemos dado velocidad 10, para que las colisiones sean más ajustadas y hemos utilizado el primer algoritmo que vimos en el capítulo anterior.

```
import pygame

# Inicializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Colores
NEGRO = (0, 0, 0)
AZUL = (0, 0, 255)
VERDE =(0, 255, 0)
MARRON =(128, 64, 0)

# Funciones
def dibujar_muro(superficie, rectangulo):
    pygame.draw.rect(superficie, MARRON, rectangulo)

def dibujar_personaje(superficie, rectangulo):
    pygame.draw.rect(superficie, AZUL, rectangulo)

# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

# Datos
muros = [
    pygame.Rect(500, 100, 300, 100),
    pygame.Rect(200, 200, 100, 300),
    pygame.Rect(500, 500, 300, 100),
    pygame.Rect(1000, 200, 100, 300)]
```

```
personaje = pygame.Rect(600, 400, 50, 50)
personaje_vel_x = 0
personaje_vel_y = 0

roca = pygame.Rect(500, 200, 300, 100)
nave = pygame.Rect(600, 500, 50, 50)
nave_vel_x = 0
nave_vel_y = 0


# Bucle principal
jugando = True
while jugando:

    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                direccion ="derecha"
                personaje_vel_x = 10
            if event.key == pygame.K_LEFT:
                direccion = "izquierda"
                personaje_vel_x = -10
            if event.key == pygame.K_DOWN:
                direccion = "abajo"
                personaje_vel_y = 10
            if event.key == pygame.K_UP:
                direccion = "arriba"
                personaje_vel_y = -10


        if event.type == pygame.KEYUP:
            if event.key == pygame.K_RIGHT:
                personaje_vel_x = 0
            if event.key == pygame.K_LEFT:
                personaje_vel_x= 0
            if event.key == pygame.K_DOWN:
                personaje_vel_y = 0
            if event.key == pygame.K_UP:
```

```
                personaje_vel_y = 0

    # Lógica
    personaje.x += personaje_vel_x
    personaje.y += personaje_vel_y

    if personaje.x > ANCHO - personaje.width:
        personaje.x = ANCHO - personaje.width
    if personaje.x < 0:
        personaje.x = 0
    if personaje.y > ALTO - personaje.height:
        personaje.y = ALTO - personaje.height
    if personaje.y < 0:
        personaje.y = 0

    for muro in muros:
        if personaje.colliderect(muro):
                personaje.x -= personaje_vel_x
                personaje.y -= personaje_vel_y
    # Dibujos
    ventana.fill(NEGRO)

    for muro in muros:
        dibujar_muro(ventana, muro)

    dibujar_personaje(ventana, personaje)

        # Actualizar
    pygame.display.update()

# Salir
pygame.quit()
```



Vamos a ver como poder poner más muros de forma más eficiente.

Eliminamos la lista muros y vamos a crear un mapa.

```
25  mapa = [
26      "XXXXXXXXXXXXXXXX",
27      "X              X",
28      "X XXXXX XXXXX X",
29      "X X            X",
30      "X XXXX XXXXXX X",
31      "X X    X       X",
32      "X XX XXXXX XXX X",
33      "X              X",
34      "XXXXXXXXXXXXXXXX"
35  ]
```

Creamos el correspondiente mapa, para que sean las baldosas de la ventana del juego.

Como la ventana tiene unas medidas de 1280 x 720

Si dividimos 1280 entre 80 nos dará 16 y si dividimos 720 entre 80 nos dará 9 es decir 16 x 9.

Si queremos poner más baldosas 1280 entre 40 serán 32 y 720 entre 40 será 18 es decir 32 x 18.

Nosotros vamos a trabajar 16 x 9 ya que el mapa en horizontal son 16 y en vertical son 9.

Vamos a definir una función:

```
23  def construir_mapa(mapa):
24      muros = []
```

```
26      y = 0
27      for fila in mapa:
28          for muro in fila:
29              if muro == "X":
30                  muros.append(pygame.Rect(x, y, 80, 80))
31              x += 80
32          x = 0
33          y += 80
34      return muros
```

Vamos a crear otra función:

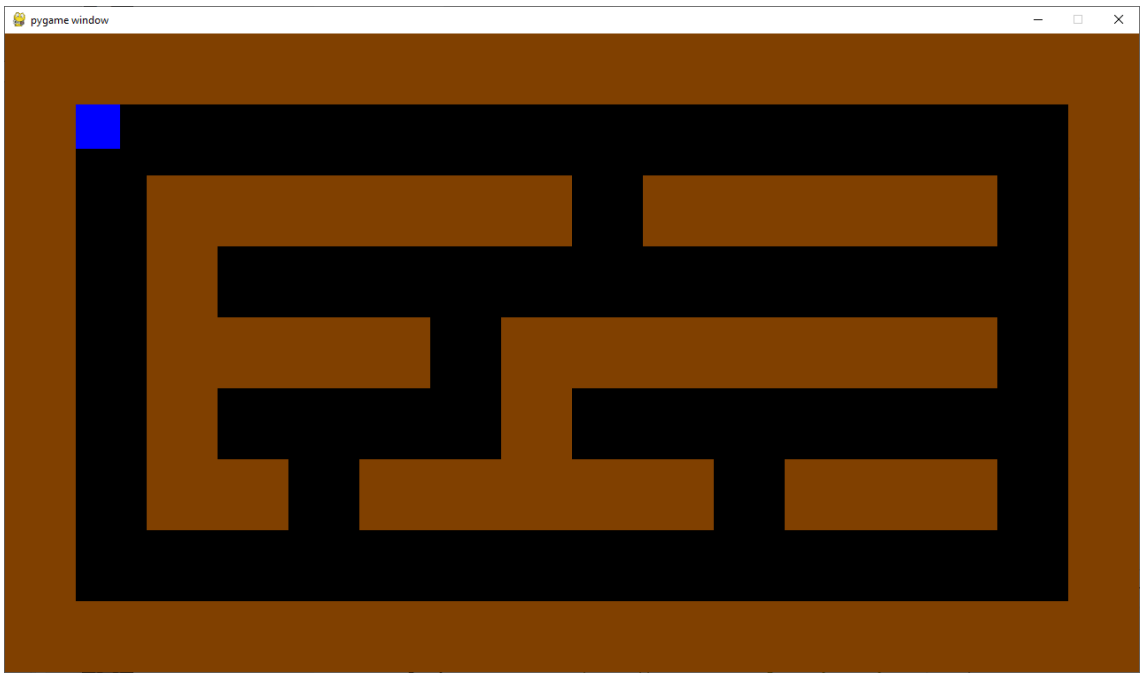
```
36  def dibujar_mapa(superficie, muros):
37      for muro in muros:
38          dibujar_muro(superficie, muro)
```

```
59  # Datos
60  muros = construir_mapa(mapa)
```

Llamamos a la función construir_mapa (con el argumento mapa).

```
123     # Dibujos
124     ventana.fill(NEGRO)
125
126     dibujar_mapa(ventana, muros)
```

Agregamos dibujar_mapa(ventana, muros).



Código completo:

```
import pygame

# Inicializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Colores
NEGRO = (0, 0, 0)
AZUL = (0, 0, 255)
VERDE =(0, 255, 0)
MARRON =(128, 64, 0)

# Funciones
def dibujar_muro(superficie, rectangulo):
    pygame.draw.rect(superficie, MARRON, rectangulo)

def dibujar_personaje(superficie, rectangulo):
    pygame.draw.rect(superficie, AZUL, rectangulo)

def construir_mapa(mapa):
    muros = []
    x = 0
    y = 0
    for fila in mapa:
        for muro in fila:
            if muro == "X":
                muros.append(pygame.Rect(x, y, 80, 80))
            x += 80
        x = 0
        y += 80
    return muros

def dibujar_mapa(superficie, muros):
    for muro in muros:
        dibujar_muro(superficie, muro)

# Mapas
```

Esta función dibuja un rectángulo de color marrón, de un objeto de tipo Rect.

Esta función con el parámetro mapa. Definimos una lista llamada muros vacía, definimos dos variables x e y con el valor 0, en este ejemplo mapa realizará un recorrido entre 0 y 8 en el primer bucle y a continuación con el siguiente bucle anidado

fila en este caso es entre 0 y 15 las 16 filas. Si muro es igual a “X” lo añades a la lista muros como un objeto Rect con los valores x e y con dimensiones 80 x 80.

Esta función con los argumentos superficie y muro con un ciclo for dibuja las baldosas.

```
mapa = [
    "XXXXXXXXXXXXXXXX",
    "X              X",
    "X XXXXXX XXXXX X",
    "X X            X",
    "X XXXX XXXXXXX X",
    "X X    X       X",
    "X XX XXXXX XXX X",
    "X              X",
    "XXXXXXXXXXXXXXXX"
]
```

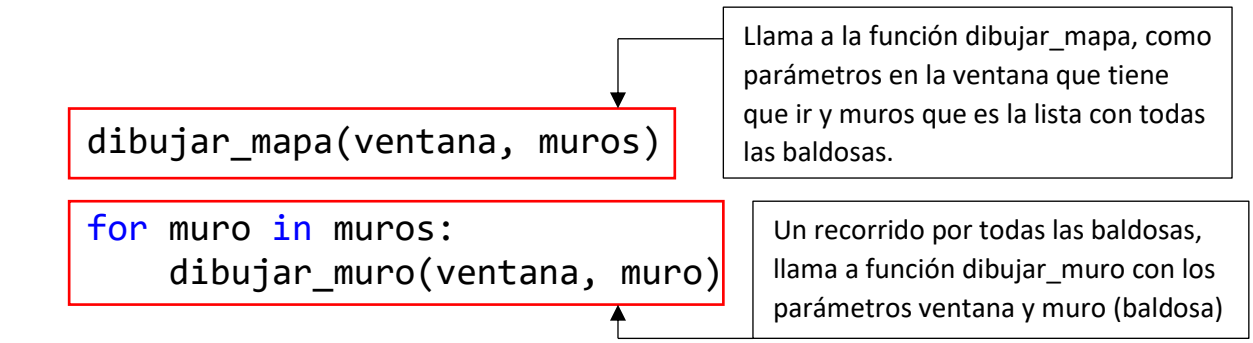
Creamos una lista llamada mapa donde dibujamos con X como queremos que sean los muros, en este caso con unas dimensiones de 16 x 9.

```
# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

# Datos
muros = construir_mapa(mapa)
personaje = pygame.Rect(80, 80, 50, 50)
personaje_vel_x = 0
personaje_vel_y = 0

roca = pygame.Rect(500, 200, 300, 100)
nave = pygame.Rect(600, 500, 50, 50)
nave_vel_x = 0
nave_vel_y = 0


# Bucle principal
jugando = True
while jugando:

    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                direccion ="derecha"
```

Llamamos a la función construir_mapa para que retorne una lista con las baldosas.

```
                personaje_vel_x = 10
            if event.key == pygame.K_LEFT:
                direccion = "izquierda"
                personaje_vel_x = -10
            if event.key == pygame.K_DOWN:
                direccion = "abajo"
                personaje_vel_y = 10
            if event.key == pygame.K_UP:
                direccion = "arriba"
                personaje_vel_y = -10


        if event.type == pygame.KEYUP:
            if event.key == pygame.K_RIGHT:
                personaje_vel_x = 0
            if event.key == pygame.K_LEFT:
                personaje_vel_x= 0
            if event.key == pygame.K_DOWN:
                personaje_vel_y = 0
            if event.key == pygame.K_UP:
                personaje_vel_y = 0

    # Lógica
    personaje.x += personaje_vel_x
    personaje.y += personaje_vel_y

    if personaje.x > ANCHO - personaje.width:
        personaje.x = ANCHO - personaje.width
    if personaje.x < 0:
        personaje.x = 0
    if personaje.y > ALTO - personaje.height:
        personaje.y = ALTO - personaje.height
    if personaje.y < 0:
        personaje.y = 0

    for muro in muros:
        if personaje.colliderect(muro):
                personaje.x -= personaje_vel_x
                personaje.y -= personaje_vel_y
    # Dibujos
    ventana.fill(NEGRO)
```

Este bucle controla que nuestro personaje no pueda traspasar el muro.




```
        dibujar_mapa(ventana, muros)
        for muro in muros:
            dibujar_muro(ventana, muro)
        dibujar_personaje(ventana, personaje)


        # Actualizar
        pygame.display.update()

# Salir
pygame.quit()
```

24.- Método collidepoint

Vamos a seguir viendo las posibilidades que nos ofrece los mapas para hacer juegos en pygame.

Vamos a escribir el siguiente código:

```
import pygame

# Incializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Colores
NEGRO = (0, 0, 0)
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)
CLARO = (200, 165, 120)
MARRON = (120, 60, 20)

# Mapas
mapa = [
    "XXXXXXXXXXXXXXXX",
    "XXXFXXXXXXXXFXX",
    "X               X",
    "X XXXXXX XXXXX X",
    "X XXXFXX  FXXX X",
    "X XXX XX XXXXF X",
    "X               X",
    "XXXXXXFXXXXXXXXX",
    "XXXXXXXXXXXXXXXX"
]

# Funciones
def dibujar_muro(superficie, rectangulo):
    pygame.draw.rect(superficie, MARRON, rectangulo)

def dibujar_manzana(superficie, rectangulo):
    pygame.draw.rect(superficie, VERDE, rectangulo)

def dibujar_personaje(superficie, rectangulo):
```

Este mapa tiene dos letras, la X para el muro y la F para las manzanas.

Creamos una función para dibujar también las manzanas.

```
    pygame.draw.rect(superficie, AZUL, rectangulo)

def construir_mapa(mapa):
    muros = []
    manzanas = []
    x = 0
    y = 0
    for fila in mapa:
        for baldosa in fila:
            if baldosa == "X":
                muros.append(pygame.Rect(x, y, 80, 80))
            if baldosa == "F":
                manzanas.append(pygame.Rect(x, y, 80,
80))
            x += 80
        x = 0
        y += 80
    return muros, manzanas

def dibujar_mapa(superficie, muros, manzanas):
    for muro in muros:
        dibujar_muro(superficie, muro)
    for manzana in manzanas:
        dibujar_manzana(superficie, manzana)

# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

# Datos
muros, manzanas = construir_mapa(mapa)

personaje = pygame.Rect(100, 400, 40, 40)
personaje_vel_x = 0
personaje_vel_y = 0

# Bucle principal
jugando = True
while jugando:
    reloj.tick(60)

    # Eventos
```

Tenemos dos listas vacías muros y manzanas, según baldose detecte kuna X o una F lo agregará a la correspondiente lista.

Retorna dos listas.

La función dibujar_mapa ahora le pasamos un argumento de más para llamar a dos funciones, dibujar_muro y dibujar_manzana

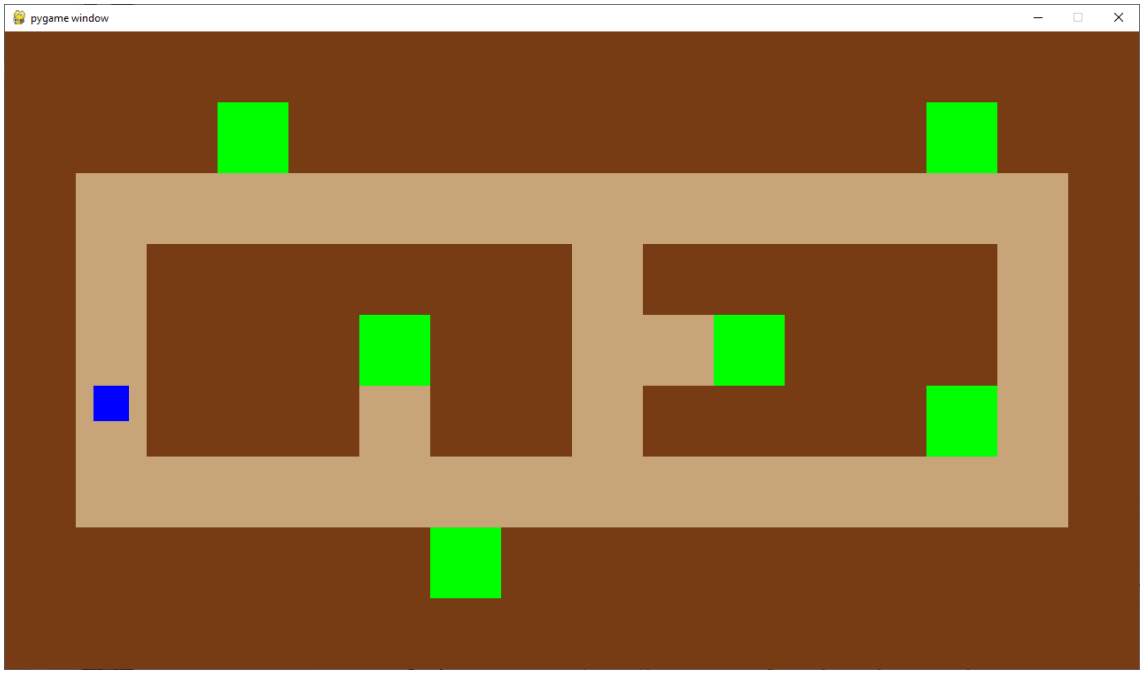
La función construir_mapa(mapa) retorna dos valores que almacenaremos en las listas muros y manzanas.

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                personaje_vel_x = 2
            if event.key == pygame.K_LEFT:
                personaje_vel_x = -2
            if event.key == pygame.K_DOWN:
                personaje_vel_y = 2
            if event.key == pygame.K_UP:
                personaje_vel_y = -2

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_RIGHT:
                personaje_vel_x = 0
            if event.key == pygame.K_LEFT:
                personaje_vel_x = 0
            if event.key == pygame.K_DOWN:
                personaje_vel_y = 0
            if event.key == pygame.K_UP:
                personaje_vel_y = 0

    # Lógica
    personaje.x += personaje_vel_x
    personaje.y += personaje_vel_y

    for muro in muros:
        if personaje.colliderect(muro):
            personaje.x -= personaje_vel_x
            personaje.y -= personaje_vel_y

    for manzana in list(manzanas):
        if personaje.collidepoint(manzana.centerx,
manzana.centery):
            manzanas.remove(manzana)

    # Dibujos
    ventana.fill(CLARO)

    dibujar_mapa(ventana, muros, manzanas)
    dibujar_personaje(ventana, personaje)
```

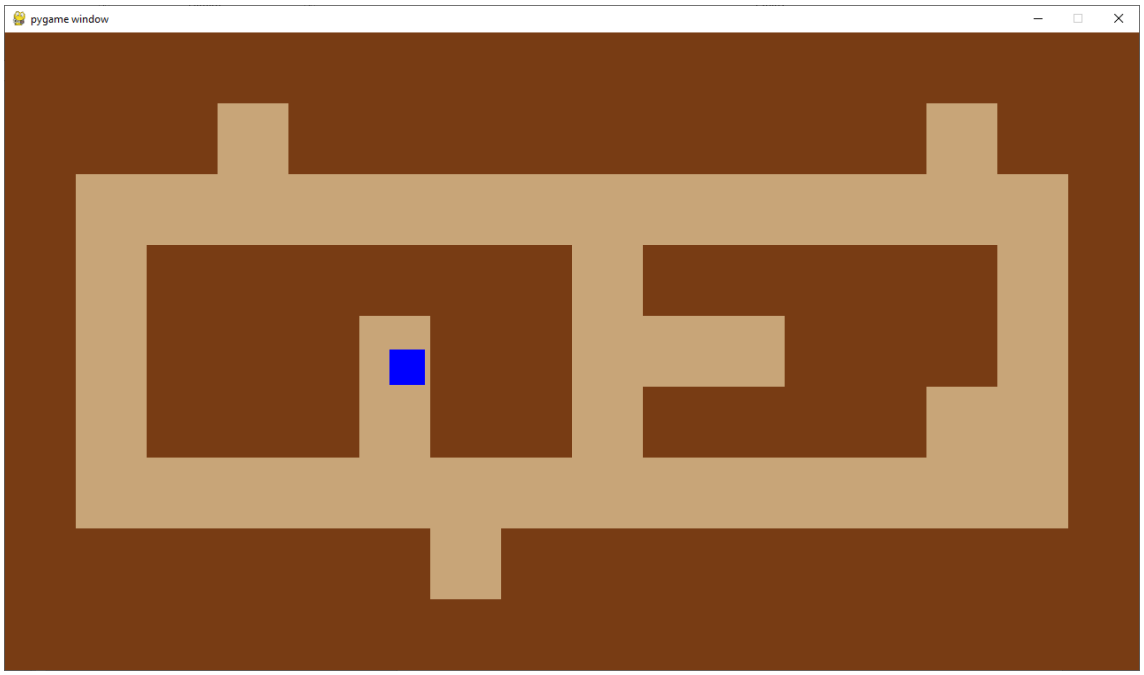
Para controla cuando el personaje detecta una manzana utilizaremos el método collidepoint de este modo le decimos que punto de la manzana tiene que tocar.

La función dibujar_mapa ahora le pasamos tres argumentos.

```
    # Actualizar
    pygame.display.update()

# Salir
pygame.quit()
```



Nos vamos a comer las manzanas.



25.- La clase Surface de pygame

En este capítulo vamos a ver otra clase predefinida que viene el la librería pygame que nos va a facilitar mucho la codificación de nuestros juegos.

Ya hemos visto la clase Rect que nos permite crear un objeto de tipo Rect y pasársela directamente a la función rect, para dibujar un rectángulo.

En este capítulo vamos a ver la clase Surface (superficie) que ya la hemos utilizado porque ya hemos creado objetos de tipo superficie mediante algunas funciones esenciales que hemos tenido de utilizar en nuestros juegos, por ejemplo la función .set_mode().

Esta función nos va a crear un objeto de tipo superficie.

```
ventana = pygame.display.set_mode((ANCHO, ALTO))
```

Si hacemos un print(ventna) para que nos muestre que tipo de objeto es.

<Surface (1280x720x32 SW)>

Bueno pues no solo vamos utilizar las superficies a la hora de crear la ventana del juego, podemos crear superficies directamente mediante la clase Surface.

Por ejemplo:

manzana = pygame.Surface((50, 50)), vamos a ver el contenido de esta variable.

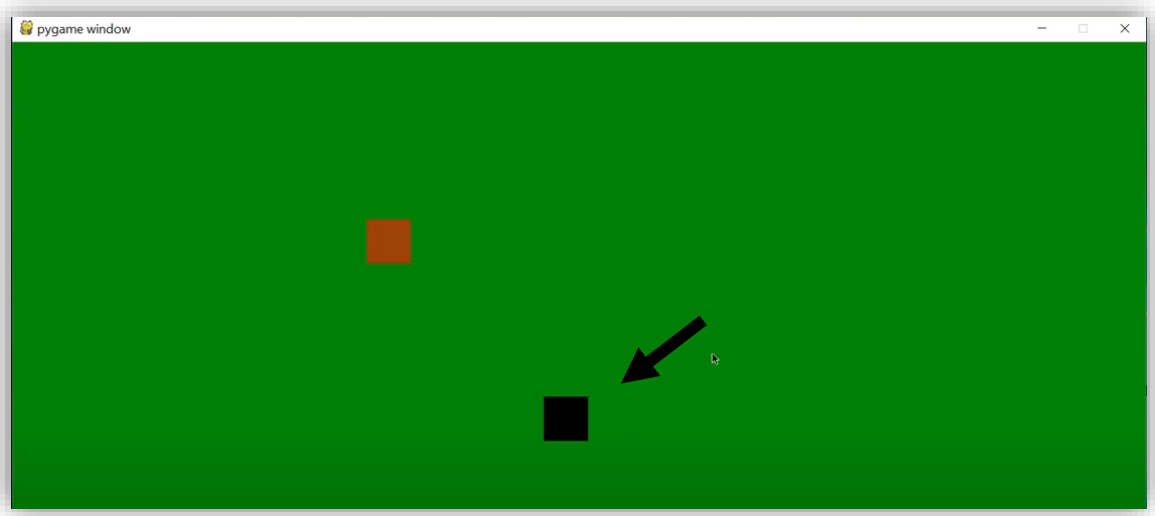
print(manzana)

<Surface (50x50x32 WS)>

La superficie menor la podemos pegar a una superficie mayor, la manzana a la ventana.

ventana.blit(manzana, (600, 400))

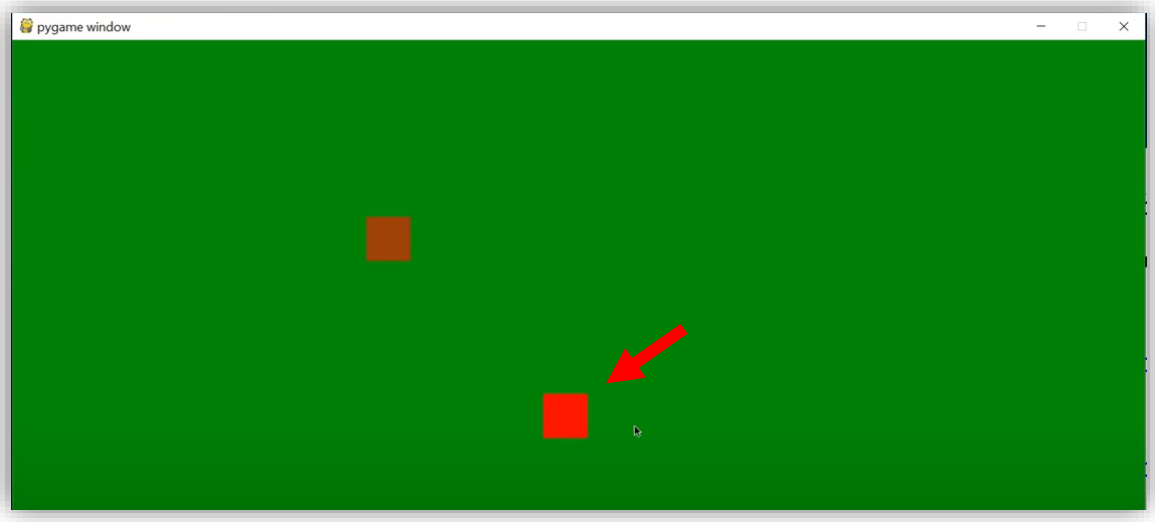
Nos pega una superficie, pero por defecto tiene el color negro.



Podemos hacer los siguiente:

manzana.fill(ROJO)

ventana.blit(manzana, (600, 400))



Sobre esta superficie podemos hacer dibujos, cargar imágenes, podríamos llevar a cabo diferentes opciones al igual que lo podemos hacer con la superficie ventana.

Para que nos puede servir este objeto de la clase superficies si ya podíamos dibujar un cuadrado perfectamente, pues esta clase va a tener muchos métodos que nos pueden facilitar la codificación.

Si nacemos un dir(pygame.Surface)

Veremos los métodos que tenemos a nuestra disposición para manejar Surface.

El que nos va a interesar en este capítulo van a ser el método get.rect, con este método no va a permitir obtener un objeto de tipo rectángulo que contenga las coordenadas y las medidas de esta superficie.

Ejemplo:

manzana_rect = manzana.get_tect()

Vamos a hace un print(manzana_rect)

<rect(0,0,50,50)>

Un objeto de tipo rectángulo que se sitúa por defecto en las coordenadas o,o y que sus dimensiones son 50 de ancho por 50 de alto.

Por lo tanto este método nos puede permitir muy fácilmente situar superficies en la ventana y poder manejar sus coordenadas de una forma sencilla.

Vamos a ver otro uso de esta clase que va a ser muy común, que va a ser a la hora de cargar imágenes en nuestros juegos.

En la carpeta del proyecto tengo una baldosa manzana.

baldosa_manzana.png

```
62  # Ventana
63  ventana = pygame.display.set_mode((ANCHO, ALTO))
64  imagen_manzana = pygame.image.load("baldosa_manzana.png").convert_alpha()
```

Si hacemos print(imagen_manzana)

<Surface (80x80x32 SW)>

Como las baldosas que hicimos en el capítulo del mapa.

```
67  # Datos
68  muros, manzanas = construir_mapa(mapa)
69  manzana_rect = imagen_manzana.get_rect()
70  manzana_rect.x = 600
71  manzana_rect.y = 400
```

Ya tenemos imagen_manzana y manzana_rect.

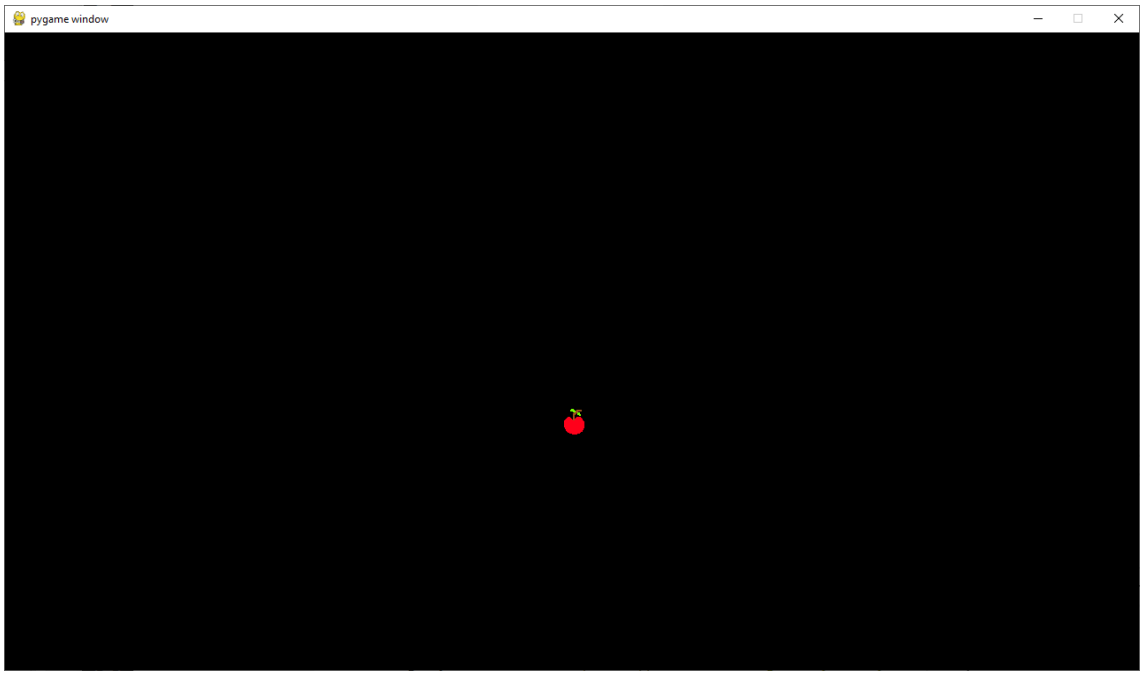
La ubicamos en las coordenadas x = 600 e y = 400.

```
127     # Actualizar
128     ventana.blit(imagen_manzana, manzana_rect)
129     pygame.display.update()
```

Un resumen de código:

```
import pygame

# Incializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))
imagen_manzana =
pygame.image.load("baldosa_manzana.png").convert_alpha()

# Datos
manzana_rect = imagen_manzana.get_rect()
manzana_rect.x = 600
manzana_rect.y = 400

reloj = pygame.time.Clock()

# Bucle principal
jugando = True
while jugando:
    reloj.tick(60)

    # Evento
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False

    # Actualizar
    ventana.blit(imagen_manzana, manzana_rect)
    pygame.display.update()

# Salir
pygame.quit()
```

Este será el resultado:



Podemos agregar el siguiente código para que la manzana se mueva:

manzana_rect.x += 1

manzana_rect.y += 1

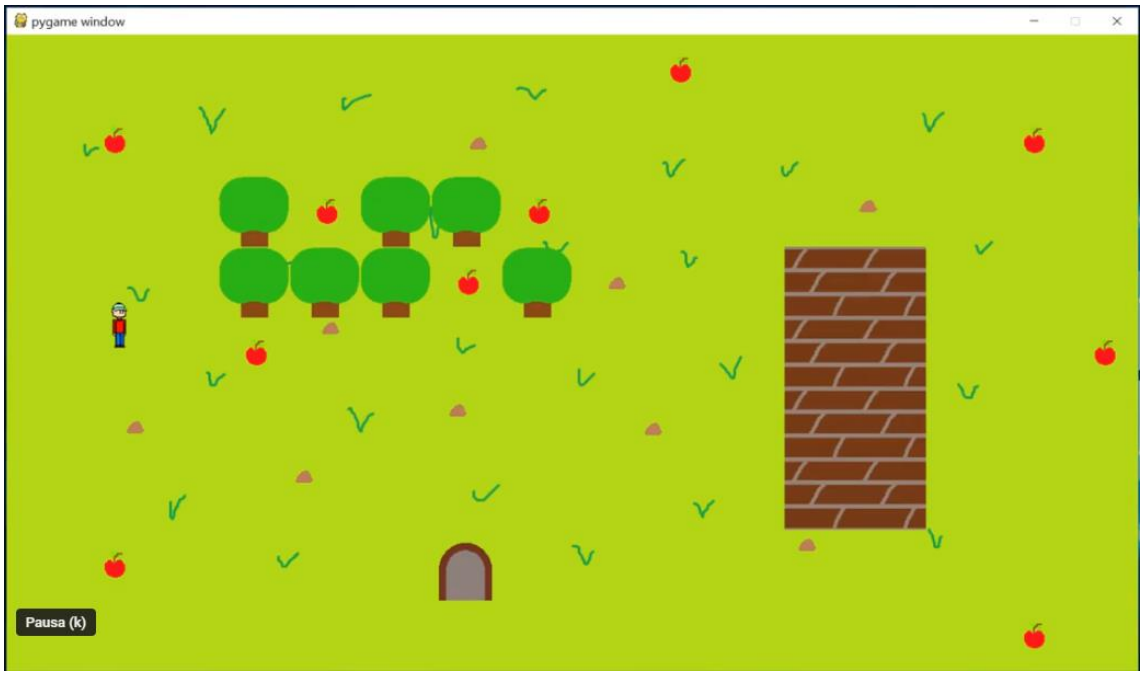
En las coordenadas:

manzana_rect.x = 100

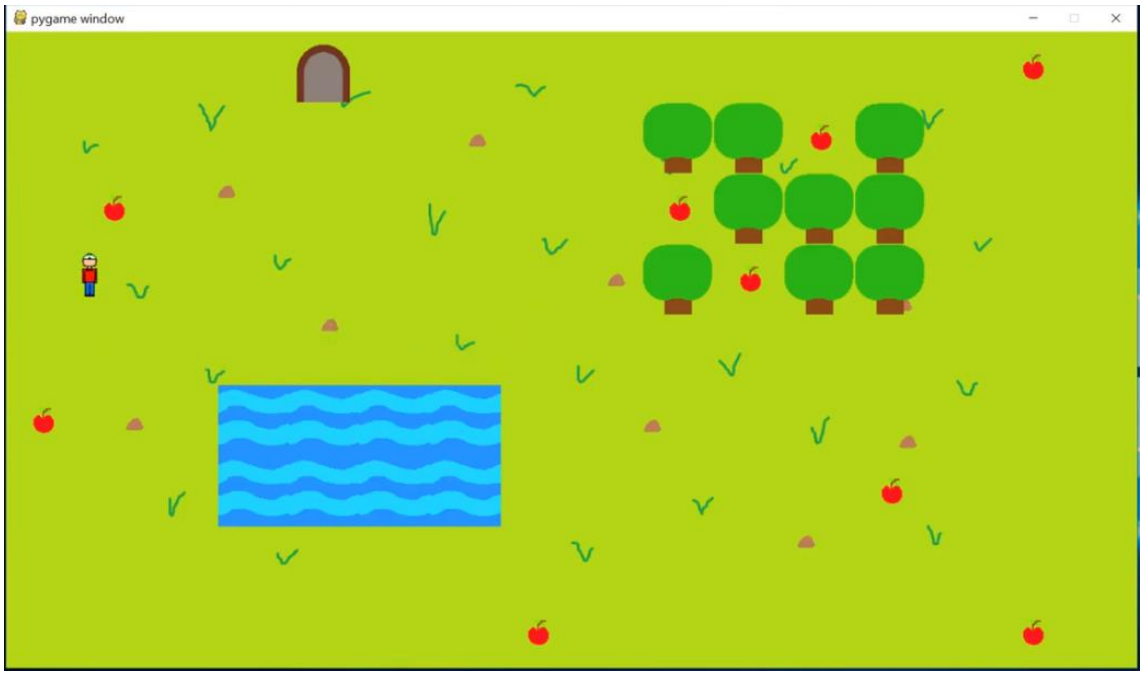
manzana_rect.y = 100

Cuando ejecutemos se moverá en diagonal.

26.- Habitaciones



Vamos a llevar a cabo un juego sencillo en el que triplemente hay un personaje que se puede mover por la ventana del juego en la que hemos puesto unos árboles, unas manzanas, un muro y una puerta para poder acceder a otra habitación.



Es un juego sencillo pero nos va a permitir recapitular un poco el uso de las superficies y de los rectángulos al mismo tiempo que vamos a poder ver algunas optimizaciones que pueden hacer el juego más fluido.

A continuación vamos a comentar el código de este proyecto.

```
import pygame
import copy

# Inicializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
VERDE = (0, 255, 0)
MARRON = (77, 38, 0)
AZUL = (0, 0, 255)
GRIS = (184, 184, 184)

# Mapa
mapa1 = [
    "          F       ",
    " F              F ",
    "   AFAAF          ",
    "   AAAFA   MM     ",
    "   F       MM   F",
    "           MM     ",
    "           MM     ",
    " F    P           ",
    "               F "
]

mapa2 = [
    "    P          F " ,
    "          AAFA    ",
    " F        FAAA    ",
    "          AFAA    ",
    "     F            ",
    "F  SSSS           ",
    "   SSSS     F     ",
    "       F       F "
]
```

Vamos a definir dos mapas, cada uno corresponde a una habitación.

mapa1 y mapa2.

```
# Funciones
def construir_mapa(superficie, mapa):
    limites = []
    frutas = []
    puertas = []
    x = 0
    y = 0
    for linea in mapa:
        for baldosa in linea:
            if baldosa == "M":
                limites.append([baldosa_muro,
pygame.Rect(x,y, 80, 80)])
            elif baldosa == "S":
                limites.append([baldosa_agua,
pygame.Rect(x,y, 80, 80)])
            elif baldosa == "A":
                limites.append([baldosa_arbol,
pygame.Rect(x,y, 80, 80)])
            elif baldosa == "F":
                frutas.append([baldosa_manzana,
pygame.Rect(x, y, 80, 80)])
            elif baldosa == "P":
                puertas.append([baldosa_puerta,
pygame.Rect(x, y, 80, 80)])
            x += 80
        x = 0
        y += 80
    return limites, frutas, puertas
```

A continuación tenemos la función construir_mapa, que nos va a retornar tres listas:

Limites, frutas y puertas.

En limites vamos guardar aquellas baldosas que representan limites a los movimientos del jugador.

baldosa_muro, baldosa_agua y baldosa_arbol.

M, S y A.

baldosa_manzana F

baldosa_puerta P

```
# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

imagen_fondo =
pygame.image.load("img/fondo_mapa.png").convert()

baldosa_muro =
pygame.image.load("img/baldosa_muro.png").convert()
baldosa_agua =
pygame.image.load("img/baldosa_agua.png").convert()
```

Cargamos la imágenes fondo

Cargamos la imágenes baldosas

Cargamos la imágenes baldosas

```
baldosa_arbol =
pygame.image.load("img/baldosa_arbol.png").convert_alph
a()
baldosa_puerta =
pygame.image.load("img/baldosa_puerta.png").convert_alp
ha()
baldosa_manzana =
pygame.image.load("img/baldosa_manzana.png").convert_al
pha()
```

Cargamos la imágenes del jugador

```
jugador0_par =
pygame.image.load("img/par_0.png").convert_alpha()
jugador1_der =
pygame.image.load("img/der_1.png").convert_alpha()
jugador2_der =
pygame.image.load("img/der_2.png").convert_alpha()
jugador3_der =
pygame.image.load("img/der_3.png").convert_alpha()
jugador4_der =
pygame.image.load("img/der_4.png").convert_alpha()
jugador1_izq =
pygame.image.load("img/izq_1.png").convert_alpha()
jugador2_izq =
pygame.image.load("img/izq_2.png").convert_alpha()
jugador3_izq =
pygame.image.load("img/izq_3.png").convert_alpha()
jugador4_izq =
pygame.image.load("img/izq_4.png").convert_alpha()
jugador1_arr =
pygame.image.load("img/arr_1.png").convert_alpha()
jugador2_arr =
pygame.image.load("img/arr_2.png").convert_alpha()
jugador3_arr =
pygame.image.load("img/arr_3.png").convert_alpha()
jugador4_arr =
pygame.image.load("img/arr_4.png").convert_alpha()
jugador1_abj =
pygame.image.load("img/baj_1.png").convert_alpha()
jugador2_abj =
pygame.image.load("img/baj_2.png").convert_alpha()
jugador3_abj =
pygame.image.load("img/baj_3.png").convert_alpha()
```

Llama a una función que retorna tres valores, por ese motivo son tuplas.

Cargamos la imágenes del jugador

```
jugador4_abj =
pygame.image.load("img/baj_4.png").convert_alpha()

jugador_imagen = jugador0_par

# Datos
habitacion1 = construir_mapa(ventana, mapa1)
habitacion2 = construir_mapa(ventana, mapa2)

habitacion = habitacion1

jugador_rectangulo = jugador_imagen.get_rect()
jugador_rectangulo.x = 100
jugador_rectangulo.y = 300
jugador_vel_x = 0
jugador_vel_y = 0
frames_jugador = 0

# Bucle principal
jugando = True
while jugando:
    reloj.tick(60)

    # Eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False

    moviendose_derecha = False
    moviendose_izquierda = False
    moviendose_arriba = False
    moviendose_abajo = False

    jugador_vel_x = 0
    jugador_vel_y = 0

    pulsado = pygame.key.get_pressed()

    if pulsado[pygame.K_LEFT]:
        jugador_vel_x = -3
        moviendose_izquierda = True
    if pulsado[pygame.K_RIGHT]:
```

A la variable jugador_imagen le asignamos la del jugador parado.

Hemos creado dos habitaciones habitaccion1 y habitacion2. La variable habitación le asignamos habitacion1

El jugador_rectangulo lo vamos a obtener de jugador_imagen con la que llamamos al método get_rect().

La imagen jugador la hemos cargado y por lo tanto es un objeto de la clase superficie y vamos a obtener un objeto de la clase Rect.

Le asignamos las coordenadas x e y, donde queremos que aparezca el personaje al comienzo del juego.

Definimos a 0 velocidad y frames del jugador.

En el bucle principal, como evento somo vamos a definir el de cerrar ventana.

Vamos a utilizar la función get.pressed(), esta función lo que hace es ver el estado de cada una de las teclas del teclado y devolver un valor booleano.

Si pulsamos la tecla izquierda vamos cambiar el valor de la variable moviéndose_izquierda a True, además de controlar la velocidad de jugador.

Haremos lo mismo para el movimiento a la derecha, arriba y abajo.

Para que cuando no se toque ninguna flecha, al principio estas variables están todas en False.

```
        jugador_vel_x = 3
        moviendose_derecha = True
    if pulsado[pygame.K_UP]:
        jugador_vel_y = -3
        moviendose_arriba = True
    if pulsado[pygame.K_DOWN]:
        jugador_vel_y = 3
        moviendose_abajo = True

    # Lógica
    jugador_rectangulo.x += jugador_vel_x
    jugador_rectangulo.y += jugador_vel_y
```

Actualizamos las coordenadas del jugador.

```
    if jugador_rectangulo.x > ANCHO - 60:
        jugador_rectangulo.x = ANCHO - 60
    if jugador_rectangulo.x < 0:
        jugador_rectangulo.x = 0
    if jugador_rectangulo.y > ALTO - 60:
        jugador_rectangulo.y = ALTO - 60
    if jugador_rectangulo.y < 0:
        jugador_rectangulo.y = 0
```

Controlamos que no se sale de la ventana.

Comprobamos las colisiones habitación[0] son los límites.

```
    for limite in habitacion[0]:
        if jugador_rectangulo.colliderect(limite[1]):
            jugador_rectangulo.x -= jugador_vel_x
            jugador_rectangulo.y -= jugador_vel_y
```

```
    for fruta in copy.copy(habitacion[1]):
        if
jugador_rectangulo.collidepoint(fruta[1].centerx,
fruta[1].centery):
            habitacion[1].remove(fruta)
```

habitación[1] son las frutas.

```
    for puerta in habitacion[2]:
        if
jugador_rectangulo.collidepoint(puerta[1].centerx,
puerta[1].centery):
            if habitacion == habitacion1:
                habitacion = habitacion2
                jugador_rectangulo.x = 400
                jugador_rectangulo.y = 60
            else:
```

habitación[2] son las puertas.

```
habitacion = habitacion1
jugador_rectangulo.x = 560
jugador_rectangulo.y = 620
```

Pertenece al bucle anterior.

```
    # Dibujos

    ventana.blit(imagen_fondo, (0,0))
```

Mostramos todas las baldosas del mapa.

```
    for elemento in habitacion:
        for baldosa in elemento:
            ventana.blit(baldosa[0], baldosa[1])
```

```
    if moviendose_derecha:
        frames_jugador +=1
        if frames_jugador >= 21:
            frames_jugador = 1
        if frames_jugador <6:
            jugador_imagen = jugador1_der
        elif frames_jugador < 11:
            jugador_imagen = jugador2_der
        elif frames_jugador < 16:
            jugador_imagen = jugador3_der
        elif frames_jugador < 21:
            jugador_imagen = jugador4_der

        ventana.blit(jugador_imagen,
jugador_rectangulo)
```

moviéndose_derecha las imágenes del jugador serán:

```
    elif moviendose_izquierda:
        frames_jugador +=1
        if frames_jugador >= 21:
            frames_jugador = 1
        if frames_jugador <6:
            jugador_imagen = jugador1_izq
        elif frames_jugador < 11:
            jugador_imagen = jugador2_izq
        elif frames_jugador < 16:
            jugador_imagen = jugador3_izq
        elif frames_jugador < 21:
            jugador_imagen = jugador4_izq

        ventana.blit(jugador_imagen,
jugador_rectangulo)
```

moviéndose_izquierda las imágenes del jugador serán:

```
    elif moviendose_arriba:
        frames_jugador +=1
        if frames_jugador >= 21:
            frames_jugador = 1
        if frames_jugador <6:
            jugador_imagen = jugador1_arr
        elif frames_jugador < 11:
            jugador_imagen = jugador2_arr
        elif frames_jugador < 16:
            jugador_imagen = jugador3_arr
        elif frames_jugador < 21:
            jugador_imagen = jugador4_arr

        ventana.blit(jugador_imagen,
jugador_rectangulo)
```

moviéndose_arriba las imágenes del jugador serán:

```
    elif moviendose_abajo:
        frames_jugador +=1
        if frames_jugador >= 21:
            frames_jugador = 1
        if frames_jugador <6:
            jugador_imagen = jugador1_abj
        elif frames_jugador < 11:
            jugador_imagen = jugador2_abj
        elif frames_jugador < 16:
            jugador_imagen = jugador3_abj
        elif frames_jugador < 21:
            jugador_imagen = jugador4_abj

        ventana.blit(jugador_imagen,
jugador_rectangulo)
```

moviéndose_abajo las imágenes del jugador serán:

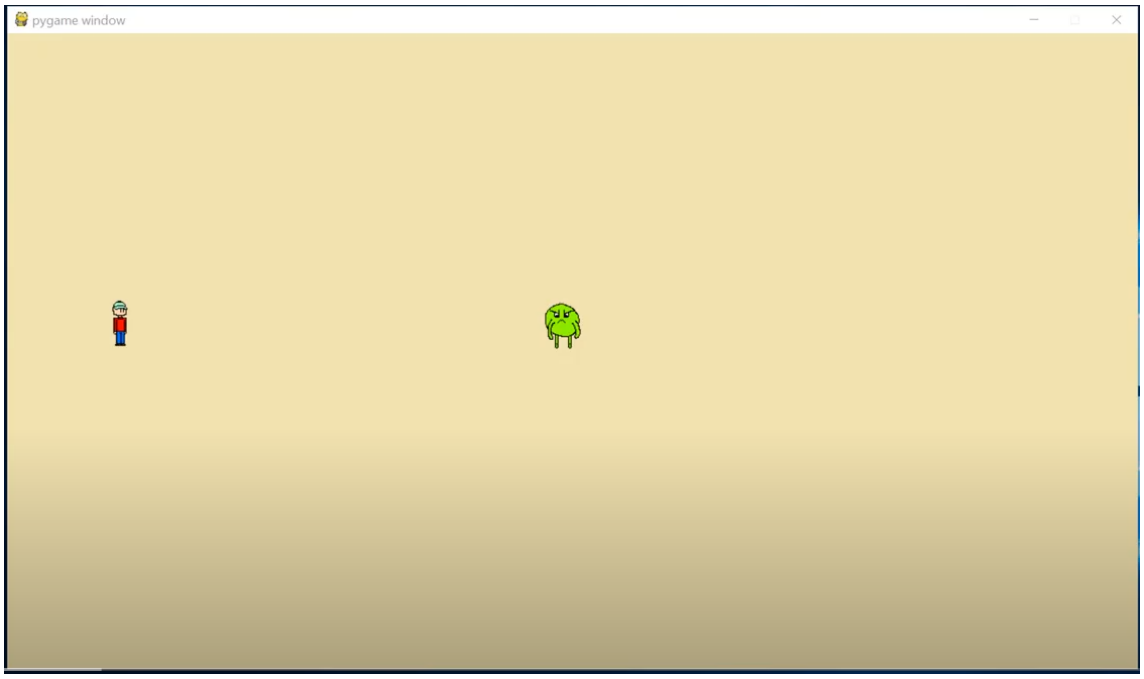
```
    else:
        jugador_imagen = jugador0_par
        ventana.blit(jugador_imagen,
jugador_rectangulo)
```

Si no mostramos la imagen del jugador parado.

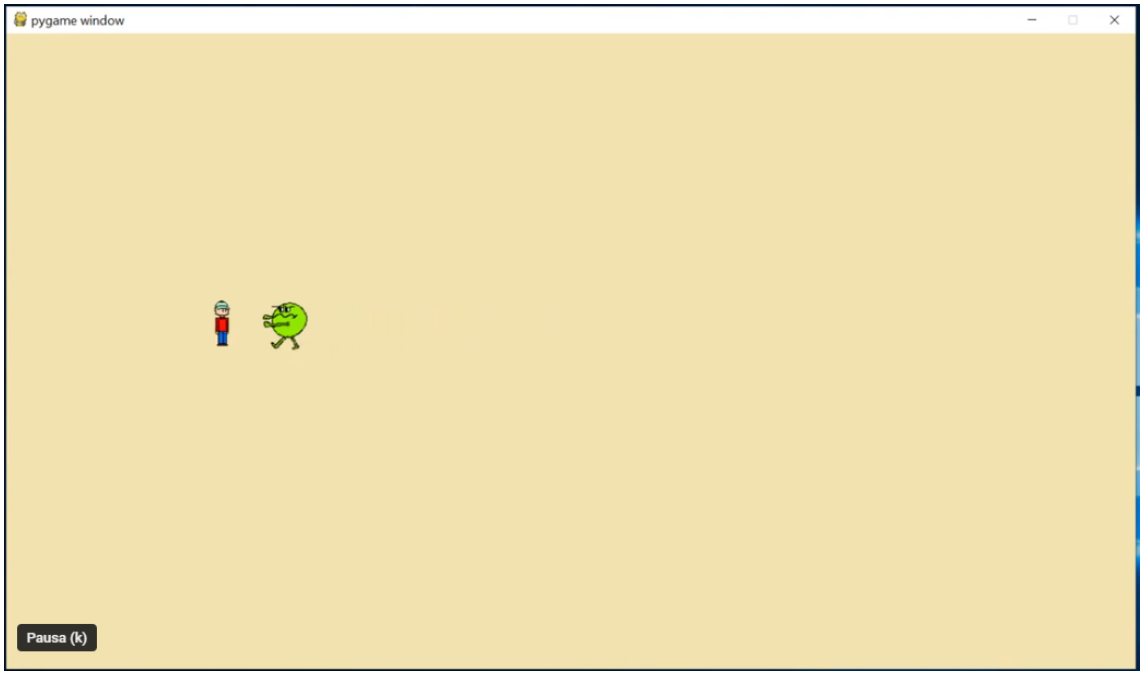
```
    # Actualizar
    pygame.display.update()

# Salir
pygame.quit()
```

Todas las imágenes necesarias para este proyecto:

27.- Persecuciones



Un juego de plataformas o de acción sin otro personajes además del jugador no tendrían mucha gracia, así que haremos otro personaje un monstruo, una versión simple que hemos estado viendo en videos anteriores.
Podríamos hacer que se moviese de un lado al otro o cubriendo el paso a un camino, pero vamos a ver otra posibilidad que podemos llevar a cabo.
Que si nos acercamos mucho a este monstruo, pues cuando entremos en su campo de visión el monstruo persiga al personaje.
Vamos a partir en este capitulo que el monstruo está parado, nos iremos acercando poco a poco cuando estamos cerca el monstruo nos empieza a seguir, cuando nos alejamos nos deja de seguir.

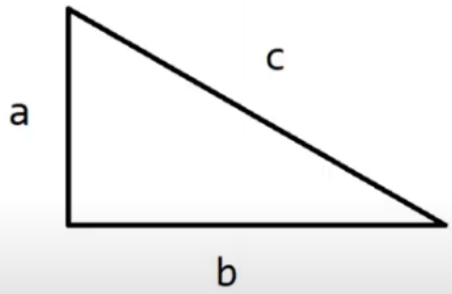


Como podemos implementar que el monstruo sea capaz de seguir al personaje, que una vez visto le persiga mientras se mantenga a una distancia indicada.
Para ello lo que vamos a utilizar la formula la distancia entre dos puntos A y B.

$$d = \sqrt{(Bx - Ax)^2 + (By - Ay)^2}$$

La distancia entre dos puntos A y B sería el valor absoluto de la coordenada x punto B menos la coordenada x del punto A elevado al cuadrado más la coordenada y menos la coordenada A elevado al cuadrado y del resultado hacemos la raíz cuadrada.

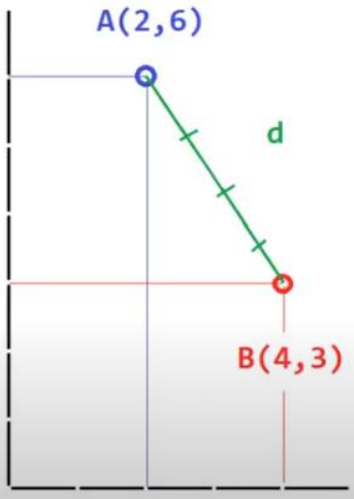
TEOREMA DE PITÁGORAS



$$a^2 + b^2 = c^2$$

$$c = \sqrt{a^2 + b^2}$$

Esta formula es una consecuencia del teorema de Pitágoras, el teorema de Pitágoras dice que la hipotenusa al cuadrado es igual a la suma de los catetos al cuadrado. Por lo tanto A al cuadrado más B al cuadrado es igual a C al cuadrado.
Despejamos el cuadrado de la C, la C será igual a la raíz cuadrada del cateto al cuadrado + cateto al cuadrado.




```
d = sqrt((Bx - Ax)**2 + (By - Ay)**2)

d = sqrt((4 - 2)**2   + (3 - 6)**2)

d = sqrt( 4 + (-3)**2)

d = sqrt( 13 )

d =  3,6
```

Si queremos calcular la distancia entre el punto A con las coordenadas (2,6) y el punto B con las coordenadas (4,3) el vector D será igual a la formula que se muestra en la figura anterior.

Vamos a seguir con el proyecto del capítulo anterior.

Agregamos las imágenes del monstruo:

```
87   monstruo1_der = pygame.image.load("img1/mon_der1.png").convert_alpha()
88   monstruo2_der = pygame.image.load("img1/mon_der2.png").convert_alpha()
89   monstruo1_izq = pygame.image.load("img1/mon_izq1.png").convert_alpha()
90   monstruo2_izq = pygame.image.load("img1/mon_izq2.png").convert_alpha()
91   monstruo1_arr = pygame.image.load("img1/mon_arr1.png").convert_alpha()
92   monstruo2_arr = pygame.image.load("img1/mon_arr2.png").convert_alpha()
93   monstruo1_baj = pygame.image.load("img1/mon_baj1.png").convert_alpha()
94   monstruo2_baj = pygame.image.load("img1/mon_baj2.png").convert_alpha()
95   monstruo0_par = pygame.image.load("img1/mon_par0.png").convert_alpha()
96
97   monstruo_imagen = monstruo0_par
```

Creamos una nueva función:

```
45   # Funciones
46   def avistamiento(a,b,distancia):
47       if (math.sqrt(((b.x - a.x)**2) + ((b.y - a.y)**2))) < distancia:
48           return True
49       else:
50           return False
```

Creamos el objeto monstruo_rectangulo de tipo rect, lo posicionamos con las coordenadas x e y. La variable frames_monstruo lo inicializamos a 0.

```
132  monstruo_rectangulo = monstruo_imagen.get_rect()
133  monstruo_rectangulo.x = 600
134  monstruo_rectangulo.y = 300
135  frames_monstruo = 0
136
137  # Bucle principal
```

En la lógica del monstruo.

```
170      # Lógica monstruo
171      monstruo_derecha = False
172      monstruo_izquierda = False
173      monstruo_abajo = False
174      monstruo_arriba = False
175      monstruo_parado = False
```

Para que el monstruo se aproxime al jugador.

```
177     if avistamiento(monstruo_rectangulo, jugador_rectangulo, 200):
178         if monstruo_rectangulo.x > jugador_rectangulo.x:
179             monstruo_rectangulo.x -=1
180             monstruo_izquierda = True
181         elif monstruo_rectangulo.x < jugador_rectangulo.x:
182             monstruo_rectangulo.x +=1
183             monstruo_derecha = True
184         elif monstruo_rectangulo.y > jugador_rectangulo.y:
185             monstruo_rectangulo.y -=1
186             monstruo_arriba = True
187         elif monstruo_rectangulo.y < jugador_rectangulo.y:
188             monstruo_rectangulo.y +=1
189             monstruo_abajo = True
190     else:
191         monstruo_parado = True
```

Para que el monstruo no salga de la ventana.

```
193     if monstruo_rectangulo.x > ANCHO - 60:
194         monstruo_rectangulo.x = ANCHO - 60
195     if monstruo_rectangulo.x < 0:
196         monstruo_rectangulo.x = 0
197     if monstruo_rectangulo.y > ALTO - 60:
198         monstruo_rectangulo.y = ALTO - 60
199     if monstruo_rectangulo.y < 0:
200         monstruo_rectangulo.y = 0
```

La colisión con el monstruo.

```
202     # Colisiones
203     if monstruo_rectangulo.collidepoint(jugador_rectangulo.centerx,
204                                         jugador_rectangulo.centery):
205         pygame.time.delay(1000)
206         break
```

Ahora los movimientos del monstruo.

```
248     # Movimientos mostruo
249     if monstruo_derecha:
250         frames_monstruo += 1
251         if frames_monstruo >= 21:
252             frames_monstruo = 1
253         if frames_monstruo < 11:
254             monstruo_imagen = monstruo1_der
255         elif frames_monstruo < 21:
256             monstruo_imagen = monstruo2_der
257 
258         ventana.blit(monstruo_imagen, monstruo_rectangulo)
```

```
260     elif monstruo_izquierda:
261         frames_monstruo += 1
262         if frames_monstruo >= 21:
263             frames_monstruo = 1
264         if frames_monstruo < 11:
265             monstruo_imagen = monstruo1_izq
266         elif frames_monstruo < 21:
267             monstruo_imagen = monstruo2_izq
268 
269         ventana.blit(monstruo_imagen, monstruo_rectangulo)

271     elif monstruo_abajo:
272         frames_monstruo += 1
273         if frames_monstruo >= 21:
274             frames_monstruo = 1
275         if frames_monstruo < 11:
276             monstruo_imagen = monstruo1_baj
277         elif frames_monstruo < 21:
278             monstruo_imagen = monstruo2_baj
279 
280         ventana.blit(monstruo_imagen, monstruo_rectangulo)

282     elif monstruo_arriba:
283         frames_monstruo += 1
284         if frames_monstruo >= 21:
285             frames_monstruo = 1
286         if frames_monstruo < 11:
287             monstruo_imagen = monstruo1_arr
288         elif frames_monstruo < 21:
289             monstruo_imagen = monstruo2_arr
290 
291         ventana.blit(monstruo_imagen, monstruo_rectangulo)
```

Te adjunto el código completo:

```
import pygame
import copy
import math

# Inicializar
pygame.init()

# Medidas
ANCHO = 1280
ALTO = 720

# Colores
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
VERDE = (0, 255, 0)
MARRON = (77, 38, 0)
AZUL = (0, 0, 255)
GRIS = (184, 184, 184)

# Mapa
mapa1 = [
    "          F       ",
    " F              F ",
    "   AFAAF          ",
    "   AAAFA   MM     ",
    "   F       MM   F",
    "           MM     ",
    "           MM     ",
    " F    P           ",
    "               F "
]

mapa2 = [
    "    P          F " ,
    "          AAFA    ",
    " F        FAAA    ",
    "          AFAA    ",
    "     F            ",
    "F  SSSS           ",
    "   SSSS     F     ",
    "       F      F "
]

# Funciones
def avistamiento(a,b,distancia):
    if (math.sqrt(((b.x - a.x)**2) + ((b.y - a.y)**2))) < distancia:
        return True
```

```
    else:
        return False

def construir_mapa(superficie, mapa):
    limites = []
    frutas = []
    puertas = []
    x = 0
    y = 0
    for linea in mapa:
        for baldosa in linea:
            if baldosa == "M":
                limites.append([baldosa_muro, pygame.Rect(x,y, 80, 80)])
            elif baldosa == "S":
                limites.append([baldosa_agua, pygame.Rect(x,y, 80, 80)])
            elif baldosa == "A":
                limites.append([baldosa_arbol, pygame.Rect(x,y, 80, 80)])
            elif baldosa == "F":
                frutas.append([baldosa_manzana, pygame.Rect(x, y, 80,
80)])
            elif baldosa == "P":
                puertas.append([baldosa_puerta, pygame.Rect(x, y, 80,
80)])
            x += 80
        x = 0
        y += 80
    return limites, frutas, puertas

# Ventana
ventana = pygame.display.set_mode((ANCHO, ALTO))
reloj = pygame.time.Clock()

imagen_fondo = pygame.image.load("img/fondo_mapa.png").convert()

baldosa_muro = pygame.image.load("img/baldosa_muro.png").convert()
baldosa_agua = pygame.image.load("img/baldosa_agua.png").convert()
baldosa_arbol =
pygame.image.load("img/baldosa_arbol.png").convert_alpha()
baldosa_puerta =
pygame.image.load("img/baldosa_puerta.png").convert_alpha()
baldosa_manzana =
pygame.image.load("img/baldosa_manzana.png").convert_alpha()

monstruo1_der = pygame.image.load("img1/mon_der1.png").convert_alpha()
monstruo2_der = pygame.image.load("img1/mon_der2.png").convert_alpha()
monstruo1_izq = pygame.image.load("img1/mon_izq1.png").convert_alpha()
monstruo2_izq = pygame.image.load("img1/mon_izq2.png").convert_alpha()
monstruo1_arr = pygame.image.load("img1/mon_arr1.png").convert_alpha()
monstruo2_arr = pygame.image.load("img1/mon_arr2.png").convert_alpha()
```

```
monstruo1_baj = pygame.image.load("img1/mon_baj1.png").convert_alpha()
monstruo2_baj = pygame.image.load("img1/mon_baj2.png").convert_alpha()
monstruo0_par = pygame.image.load("img1/mon_par0.png").convert_alpha()

monstruo_imagen = monstruo0_par

jugador0_par = pygame.image.load("img/par_0.png").convert_alpha()
jugador1_der = pygame.image.load("img/der_1.png").convert_alpha()
jugador2_der = pygame.image.load("img/der_2.png").convert_alpha()
jugador3_der = pygame.image.load("img/der_3.png").convert_alpha()
jugador4_der = pygame.image.load("img/der_4.png").convert_alpha()
jugador1_izq = pygame.image.load("img/izq_1.png").convert_alpha()
jugador2_izq = pygame.image.load("img/izq_2.png").convert_alpha()
jugador3_izq = pygame.image.load("img/izq_3.png").convert_alpha()
jugador4_izq = pygame.image.load("img/izq_4.png").convert_alpha()
jugador1_arr = pygame.image.load("img/arr_1.png").convert_alpha()
jugador2_arr = pygame.image.load("img/arr_2.png").convert_alpha()
jugador3_arr = pygame.image.load("img/arr_3.png").convert_alpha()
jugador4_arr = pygame.image.load("img/arr_4.png").convert_alpha()
jugador1_abj = pygame.image.load("img/baj_1.png").convert_alpha()
jugador2_abj = pygame.image.load("img/baj_2.png").convert_alpha()
jugador3_abj = pygame.image.load("img/baj_3.png").convert_alpha()
jugador4_abj = pygame.image.load("img/baj_4.png").convert_alpha()

jugador_imagen = jugador0_par

# Datos
habitacion1 = construir_mapa(ventana, mapa1)
habitacion2 = construir_mapa(ventana, mapa2)

habitacion = habitacion1

jugador_rectangulo = jugador_imagen.get_rect()
jugador_rectangulo.x = 100
jugador_rectangulo.y = 300
jugador_vel_x = 0
jugador_vel_y = 0
frames_jugador = 0

monstruo_rectangulo = monstruo_imagen.get_rect()
monstruo_rectangulo.x = 600
monstruo_rectangulo.y = 300
frames_monstruo = 0

# Bucle principal
jugando = True
while jugando:
    reloj.tick(60)
```

```
# Eventos
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        jugando = False

moviendose_derecha = False
moviendose_izquierda = False
moviendose_arriba = False
moviendose_abajo = False

jugador_vel_x = 0
jugador_vel_y = 0

pulsado = pygame.key.get_pressed()

if pulsado[pygame.K_LEFT]:
    jugador_vel_x = -3
    moviendose_izquierda = True
if pulsado[pygame.K_RIGHT]:
    jugador_vel_x = 3
    moviendose_derecha = True
if pulsado[pygame.K_UP]:
    jugador_vel_y = -3
    moviendose_arriba = True
if pulsado[pygame.K_DOWN]:
    jugador_vel_y = 3
    moviendose_abajo = True

# Lógica monstruo
monstruo_derecha = False
monstruo_izquierda = False
monstruo_abajo = False
monstruo_arriba = False
monstruo_parado = False

if avistamiento(monstruo_rectangulo, jugador_rectangulo, 200):
    if monstruo_rectangulo.x > jugador_rectangulo.x:
        monstruo_rectangulo.x -=1
        monstruo_izquierda = True
    elif monstruo_rectangulo.x < jugador_rectangulo.x:
        monstruo_rectangulo.x +=1
        monstruo_derecha = True
    elif monstruo_rectangulo.y > jugador_rectangulo.y:
        monstruo_rectangulo.y -=1
        monstruo_arriba = True
    elif monstruo_rectangulo.y < jugador_rectangulo.y:
        monstruo_rectangulo.y +=1
        monstruo_abajo = True
else:
```

```
            monstruo_parado = True

        if monstruo_rectangulo.x > ANCHO - 60:
            monstruo_rectangulo.x = ANCHO - 60
        if monstruo_rectangulo.x < 0:
            monstruo_rectangulo.x = 0
        if monstruo_rectangulo.y > ALTO - 60:
            monstruo_rectangulo.y = ALTO - 60
        if monstruo_rectangulo.y < 0:
            monstruo_rectangulo.y = 0

        # Colisiones
        if monstruo_rectangulo.collidepoint(jugador_rectangulo.centerx,
                                            jugador_rectangulo.centery):
            pygame.time.delay(1000)
            break

        # Lógica
        jugador_rectangulo.x += jugador_vel_x
        jugador_rectangulo.y += jugador_vel_y

        if jugador_rectangulo.x > ANCHO - 60:
            jugador_rectangulo.x = ANCHO - 60
        if jugador_rectangulo.x < 0:
            jugador_rectangulo.x = 0
        if jugador_rectangulo.y > ALTO - 60:
            jugador_rectangulo.y = ALTO - 60
        if jugador_rectangulo.y < 0:
            jugador_rectangulo.y = 0

        for limite in habitacion[0]:
            if jugador_rectangulo.colliderect(limite[1]):
                jugador_rectangulo.x -= jugador_vel_x
                jugador_rectangulo.y -= jugador_vel_y

        for fruta in copy.copy(habitacion[1]):
            if jugador_rectangulo.collidepoint(fruta[1].centerx,
fruta[1].centery):
                habitacion[1].remove(fruta)

        for puerta in habitacion[2]:
            if jugador_rectangulo.collidepoint(puerta[1].centerx,
puerta[1].centery):
                if habitacion == habitacion1:
                    habitacion = habitacion2
                    jugador_rectangulo.x = 400
                    jugador_rectangulo.y = 60
                else:
                    habitacion = habitacion1
```

```
            jugador_rectangulo.x = 560
            jugador_rectangulo.y = 620
# Dibujos

ventana.blit(imagen_fondo, (0,0))

for elemento in habitacion:
    for baldosa in elemento:
        ventana.blit(baldosa[0], baldosa[1])

# Movimientos mostruo
if monstruo_derecha:
    frames_monstruo += 1
    if frames_monstruo >= 21:
        frames_monstruo = 1
    if frames_monstruo < 11:
        monstruo_imagen = monstruo1_der
    elif frames_monstruo < 21:
        monstruo_imagen = monstruo2_der

    ventana.blit(monstruo_imagen, monstruo_rectangulo)

elif monstruo_izquierda:
    frames_monstruo += 1
    if frames_monstruo >= 21:
        frames_monstruo = 1
    if frames_monstruo < 11:
        monstruo_imagen = monstruo1_izq
    elif frames_monstruo < 21:
        monstruo_imagen = monstruo2_izq

    ventana.blit(monstruo_imagen, monstruo_rectangulo)

elif monstruo_abajo:
    frames_monstruo += 1
    if frames_monstruo >= 21:
        frames_monstruo = 1
    if frames_monstruo < 11:
        monstruo_imagen = monstruo1_baj
    elif frames_monstruo < 21:
        monstruo_imagen = monstruo2_baj

    ventana.blit(monstruo_imagen, monstruo_rectangulo)

elif monstruo_arriba:
    frames_monstruo += 1
    if frames_monstruo >= 21:
        frames_monstruo = 1
    if frames_monstruo < 11:
```

```
            monstruo_imagen = monstruo1_arr
        elif frames_monstruo < 21:
            monstruo_imagen = monstruo2_arr

        ventana.blit(monstruo_imagen, monstruo_rectangulo)

    # movimientos del jugador
    if moviendose_derecha:
        frames_jugador +=1
        if frames_jugador >= 21:
            frames_jugador = 1
        if frames_jugador <6:
            jugador_imagen = jugador1_der
        elif frames_jugador < 11:
            jugador_imagen = jugador2_der
        elif frames_jugador < 16:
            jugador_imagen = jugador3_der
        elif frames_jugador < 21:
            jugador_imagen = jugador4_der

        ventana.blit(jugador_imagen, jugador_rectangulo)

    elif moviendose_izquierda:
        frames_jugador +=1
        if frames_jugador >= 21:
            frames_jugador = 1
        if frames_jugador <6:
            jugador_imagen = jugador1_izq
        elif frames_jugador < 11:
            jugador_imagen = jugador2_izq
        elif frames_jugador < 16:
            jugador_imagen = jugador3_izq
        elif frames_jugador < 21:
            jugador_imagen = jugador4_izq

        ventana.blit(jugador_imagen, jugador_rectangulo)

    elif moviendose_arriba:
        frames_jugador +=1
        if frames_jugador >= 21:
            frames_jugador = 1
        if frames_jugador <6:
            jugador_imagen = jugador1_arr
        elif frames_jugador < 11:
            jugador_imagen = jugador2_arr
        elif frames_jugador < 16:
            jugador_imagen = jugador3_arr
        elif frames_jugador < 21:
            jugador_imagen = jugador4_arr
```

```
            ventana.blit(jugador_imagen, jugador_rectangulo)


        elif moviendose_abajo:
            frames_jugador +=1
            if frames_jugador >= 21:
                frames_jugador = 1
            if frames_jugador <6:
                jugador_imagen = jugador1_abj
            elif frames_jugador < 11:
                jugador_imagen = jugador2_abj
            elif frames_jugador < 16:
                jugador_imagen = jugador3_abj
            elif frames_jugador < 21:
                jugador_imagen = jugador4_abj

            ventana.blit(jugador_imagen, jugador_rectangulo)

        else:
            jugador_imagen = jugador0_par
            ventana.blit(jugador_imagen, jugador_rectangulo)

        # Actualizar
        pygame.display.update()

    # Salir
    pygame.quit()
```