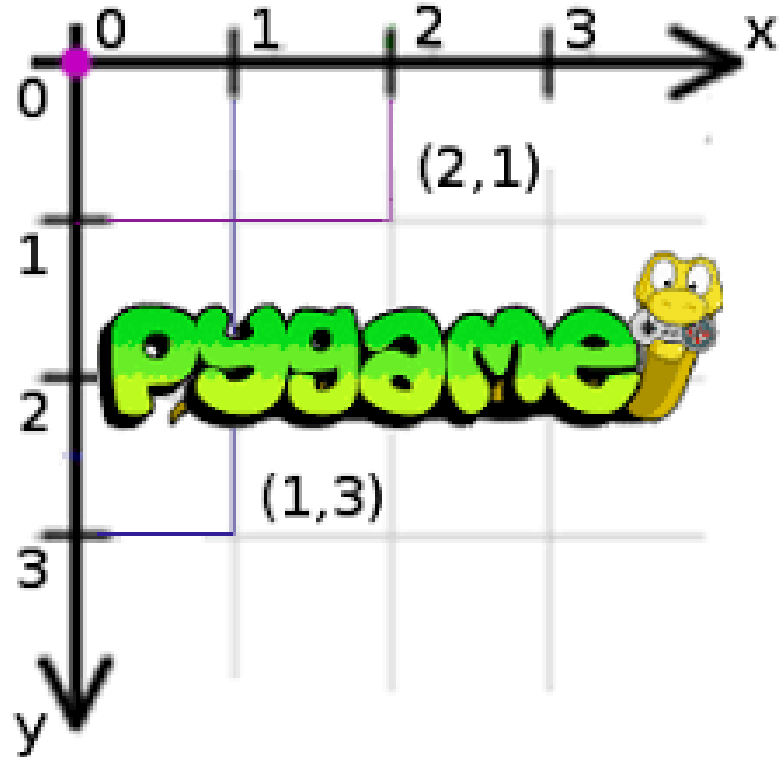


Libreria PyGame

- Instalar PyGame
- Juegos introductorios
 - race_Car
 - laberinto

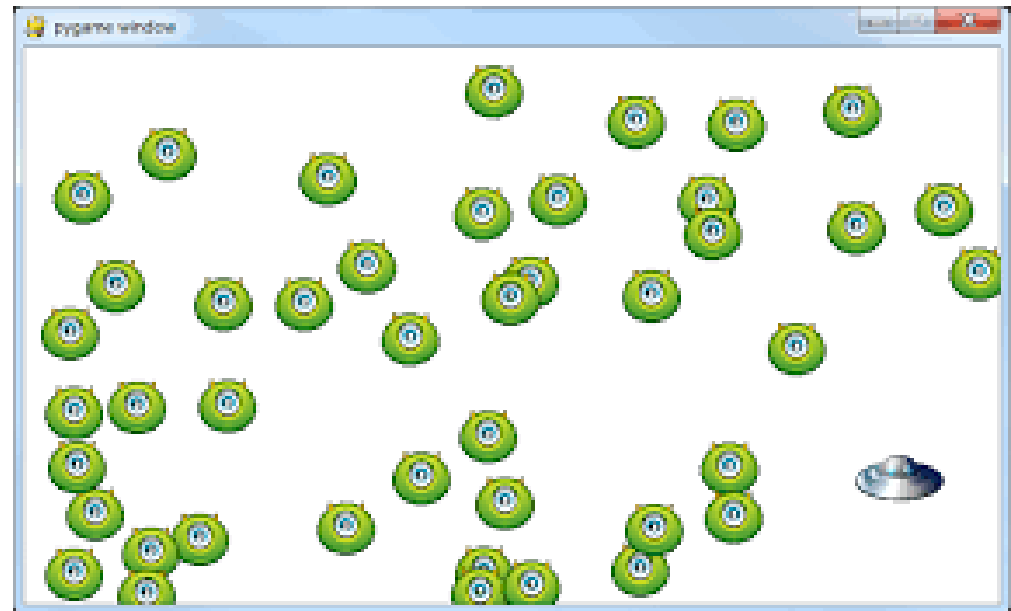
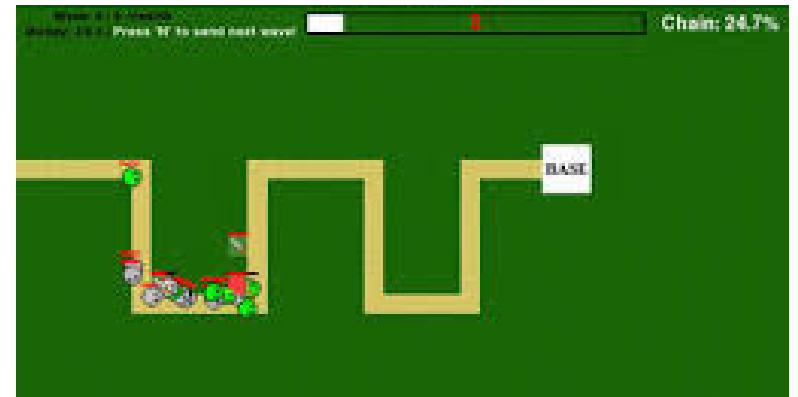


Rapidito : https://www.youtube.com/watch?v=2llq_J_R9qU

Pygame



- es un conjunto de módulos del lenguaje Python que permiten la creación de videojuegos en dos dimensiones de una manera sencilla.
- Está orientado al manejo de gráficos :
[sprites](#), [MOB's \(Movable Object Block\)](#) .
- Gracias al lenguaje, se puede prototipar y desarrollar rápidamente



Instalar PyGame

- 1.- Necesitamos tener instalada la biblioteca **pygame** coherente con nuestra versión de Python.
- En nuestro caso :
 - Abrimos Anaconda Prompt
 - Cambiamos al directorio c:\conda3
 - Y en la consola tecleamos
pip install pygame

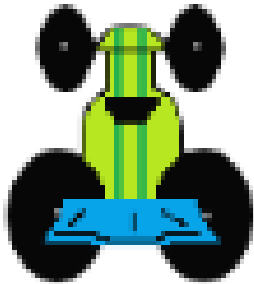
Tecleando `pip list` podemos ver los paquetes instalados.

Una vez que tengas PyGame, ¡estás listo para crear tu primer juego de PyGame!

AN INTRODUCTION TO



Juegos Introductorios

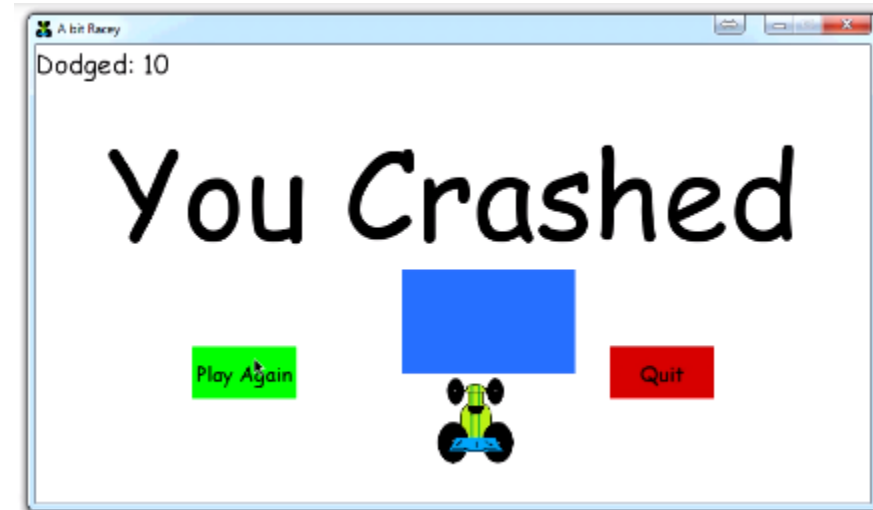


1. `race_car`
2. `laberinto`

1. RACE_CAR

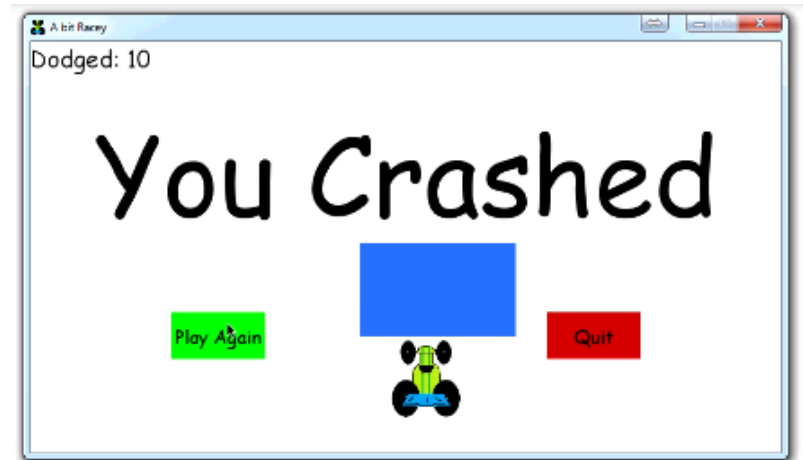
Este juego permite mover un coche a (<-) izquierda y derecha (->) esquivando cajas que caen.

El juego termina si el coche sale de la pantalla o es tocado por una caja.
O el usuario decide cerrar la ventana



1.1- Bucle de juego standard

```
import pygame  
pygame.init()
```



El tutor te proporciona un cuaderno con el juego.

Todavía no probaremos el juego, continuamos con la carcasa básica.

a) Lo primero que vamos a hacer, es darle medida a la ventana, y después le asignaremos un título.

```
import pygame
```

```
pygame.init()
```

```
gameDisplay = pygame.display.set_mode( (800,600) )
```

```
pygame.display.set_caption("Una carrerita")
```

b) Seguimos con la estructura básica :

- un bucle interior que captura eventos hasta que se cierre la ventana (que se interpreta evento QUIT)
- y un bucle exterior que controla la partida hasta que acabe.

```
clock = pygame.time.Clock()
acabado = False
```

```
while not acabado:
```

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            acabado = True
```

```
        print(event)
```

bucle
eventos

bucle
juego

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


```
pygame.quit()
quit()
```

c) Se inicializa un reloj que se ajusta a 60 ms de espera antes de volver a dibujar la pantalla.

```
clock = pygame.time.Clock()
acabado = False
#----- bucle Juego
while not acabado:
    #----- bucle evento
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            acabado = True

            print(event)
    #----- fin bucle evento
    pygame.display.update()
    clock.tick(60)
#----- fin bucle Juego

pygame.quit()
quit()
```



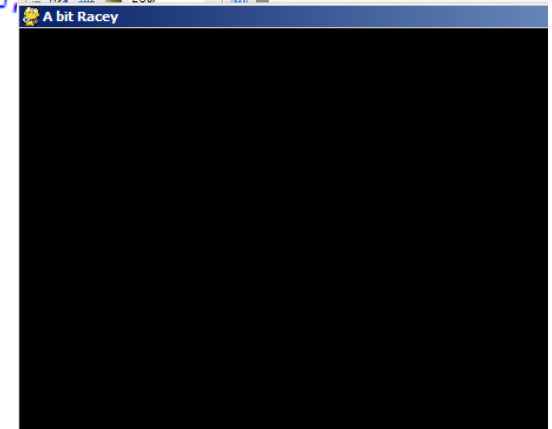
The diagram illustrates the execution flow of the provided Python code. It features two vertical orange bars. The shorter bar on the left is labeled 'bucle eventos' and corresponds to the inner 'for event in pygame.event.get():' loop. The longer bar on the right is labeled 'bucle juego' and corresponds to the outer 'while not acabado:' loop, which encloses the entire event loop and the game logic (display update and clock tick).

d) Ahora si que vamos a probar el programa-> **Run**.

Observa los mensajes que aparecen en la terminal cuando mueves el ratón sobre la pantalla de juego (que ahora está en negro)

```
===== RESTART: C:\Python3\Scripts\jocs31.py =====  
pygame 1.9.4  
Hello from the pygame community. https://www.pygame.org/contribute.html  
<Event(17-VideoExpose {})>  
<Event(16-VideoResize {'size': (800, 600), 'w': 800, 'h': 600})>  
<Event(1-ActiveEvent {'gain': 0, 'state': 1})>  
<Event(4-MouseMotion {'pos': (535, 275), 'rel': (536, 276), 'buttons': (0, 0, 0)  
})>  
<Event(1-ActiveEvent {'gain': 1, 'state': 1})>  
<Event(4-MouseMotion {'pos': (536, 275), 'rel': (1, 0), 'buttons': (0, 0, 0)})>  
<Event(4-MouseMotion {'pos': (536, 273), 'rel': (0, -2), 'buttons': (0, 0, 0)})>  
<Event(4-MouseMotion {'pos': (511, 194), 'rel': (-25, -79), 'buttons': (0, 0, 0)  
})>
```

Está devolviendo una estructura de tipo **Event** con los datos relevantes al evento.



e) Cargar i centrar la imagen: Vamos a añadir código, justo después de la instrucción `pygame.init`

`pygame.init()` → Insertar después de esta línea

```
display_width = 800
display_height = 600
```

```
gameDisplay =
pygame.display.set_mode((display_width,display_height))
```

```
black = (0,0,0)
white = (255,255,255)
red   = (255, 0, 0)
```

```
clock = pygame.time.Clock()
carImg = pygame.image.load('.\\img\\racecar.png')
```

```
x = (display_width * 0.45)    # parámetros de posición
y = (display_height * 0.8)
```

```
pygame.init()
```

```
display_width = 800  
display_height = 600
```



definir los limites con
variables

```
gameDisplay = pygame.display.set_mode((display_width,display_height))
```

```
black = (0,0,0)  
white = (255,255,255)  
red   = (255, 0, 0)
```



prepara colores

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

```
carImg = pygame.image.load('..\\img\\racecar.png')
```



carga la imagen
del coche

```
x = (display_width * 0.45)  
y = (display_height * 0.8)
```



Si no la tienes, guarda esta imagen como [racecar.png](#) (con medidas 99 x 97 px) en una carpeta que se llame [img](#) y quede dentro de la carpeta del programa.



```


#----- Bucle Principal
while not acabado:
    #----- Bucle Evento
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            acabado = True


    #----- fin bucle Evento
    gameDisplay.fill(white)
    car(x,y)

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

#----- fin bucle Principal
pygame.quit()
quit()

```

 - Rellena fondo en blanco
 - Llama a una función car

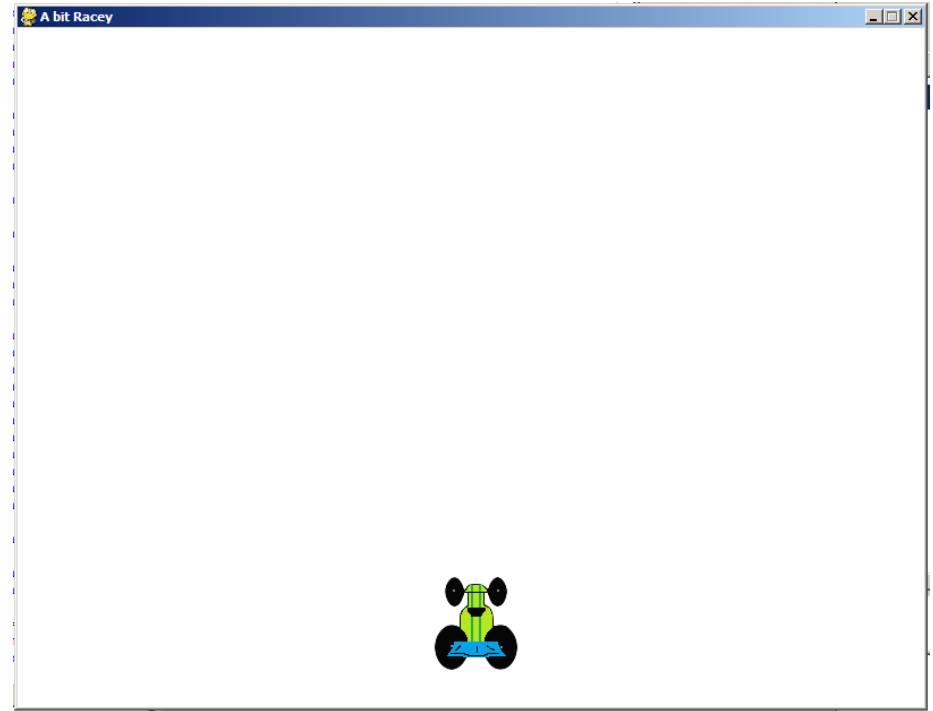
 - Refresca la pantalla de juego

```
import pygame
```

```
#----- Definición de funciones
```

```
def car(x,y):
    gameDisplay.blit(carImg, (x,y))
```

f) Probando ... Ya tenemos un fondo blanco y la figura situada !!!



```
#----- Definición de funciones
```

```
def car(x,y):  
    gameDisplay.blit(carImg, (x,y))
```


g) Vamos a controlar más eventos: las flechas

```
for event in pygame.event.get():
    #----- Tratamiento del evento
    if event.type == pygame.QUIT:
        acabado = True

    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT:
            x_change = -5
        elif event.key == pygame.K_RIGHT:
            x_change = 5

    if event.type == pygame.KEYUP:
        if event.key == pygame.K_LEFT or \
           event.key == pygame.K_RIGHT:
            x_change = 0
    #----- fin tratamiento del evento
```

1.2- Moviendo la imagen

En la zona de inicializaciones de variables definimos medidas y posiciones para algunos elementos ...

```
x = (display_width * 0.45)    # parámetros de posición  
y = (display_height * 0.8)
```

```
car_width = 73    # Ancho del coche  
x_change = 0      # parámetros para desplazamiento  
car_speed = 0     # y velocidad  
numCaidas = 0     # El contador de cajas será la puntuación
```

```
#----- Medidas de las cajas que caen del cielo  
caja_startx = random.randrange(0, display_width)  
caja_starty = -600  
caja_speed = 7  
caja_width = 100  
caja_height = 100
```

1.3- Añadiendo límites

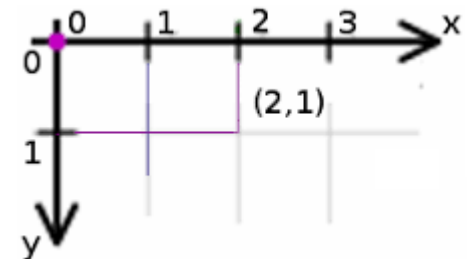
Al final del bucle de evento, añadimos la condición : si el coche se sale de la pantalla se termina el juego.

```
#----- fin bucle Evento

gameDisplay.fill(white)      # Blanquea la pantalla
x += x_change                # recalcula la coordenada x
car(x,y)                    # Dibuja el coche

if x > display_width - car_width or x < 0:
    acabado = True
```

Después de estos añadidos las flechas ya mueven el coche y al llegar al límite termina el juego.



Si has llegado hasta aquí felicidades,

Pero ahora te recomiendo que copies y pegues la
definición de funciones

y el bucle del juego

```

#----- fin bucle Evento
gameDisplay.fill(white)      # Blanquea la pantalla
x += x_change                # recalcula la coordenada x
car(x,y)                     # Dibuja el coche

                                # Dibuja las caja que cae y recalcula su altura
cajas(caja_startx, caja_starty, caja_width, caja_height, black)
caja_starty += caja_speed

if caja_starty > display_height:      # Si la caja se sale de la pantalla
    caja_starty = 0 - caja_height      # se vuelve a situar arriba en otra
    caja_startx = random.randrange(0,display_width)    # coordenada aleatoria
    numCaidas += 1                     # se suma 1 a las cajas caidas
    caja_speed += 1                   # se aumenta la velocidad
    caja_width += (numCaidas * 1.2)    #va aumentando la anchura de la caja

if y < (caja_starty + caja_height) :
    if (x > caja_startx and x < (caja_startx + caja_width)) or \
        ((x + car_width > caja_startx) and \
        (x + car_width) < (caja_startx + caja_width)) :
        acabado = True
        crash()

if x > display_width - car_width or x < 0:
    acabado = True
    crash()

cajas_caidas(numCaidas)
pygame.display.update()
clock.tick(60)

#----- fin bucle Principal

```

#----- Definición de funciones

```
def car(x,y):
    gameDisplay.blit(carImg, (x,y))

def text_objects(text, font):
    textSurface = font.render(text, True, black)
    return textSurface, textSurface.get_rect()

def message_display(text):
    # largeText = pygame.font.Font('.\\fonts\\freesansbold.ttf',115)
    largeText = pygame.font.SysFont("comicsansms", 72)
    TextSurf, TextRect = text_objects(text, largeText)
    TextRect.center = ((display_width/2),(display_height/2))
    gameDisplay.blit(TextSurf, TextRect)

    pygame.display.update()
    time.sleep(2)

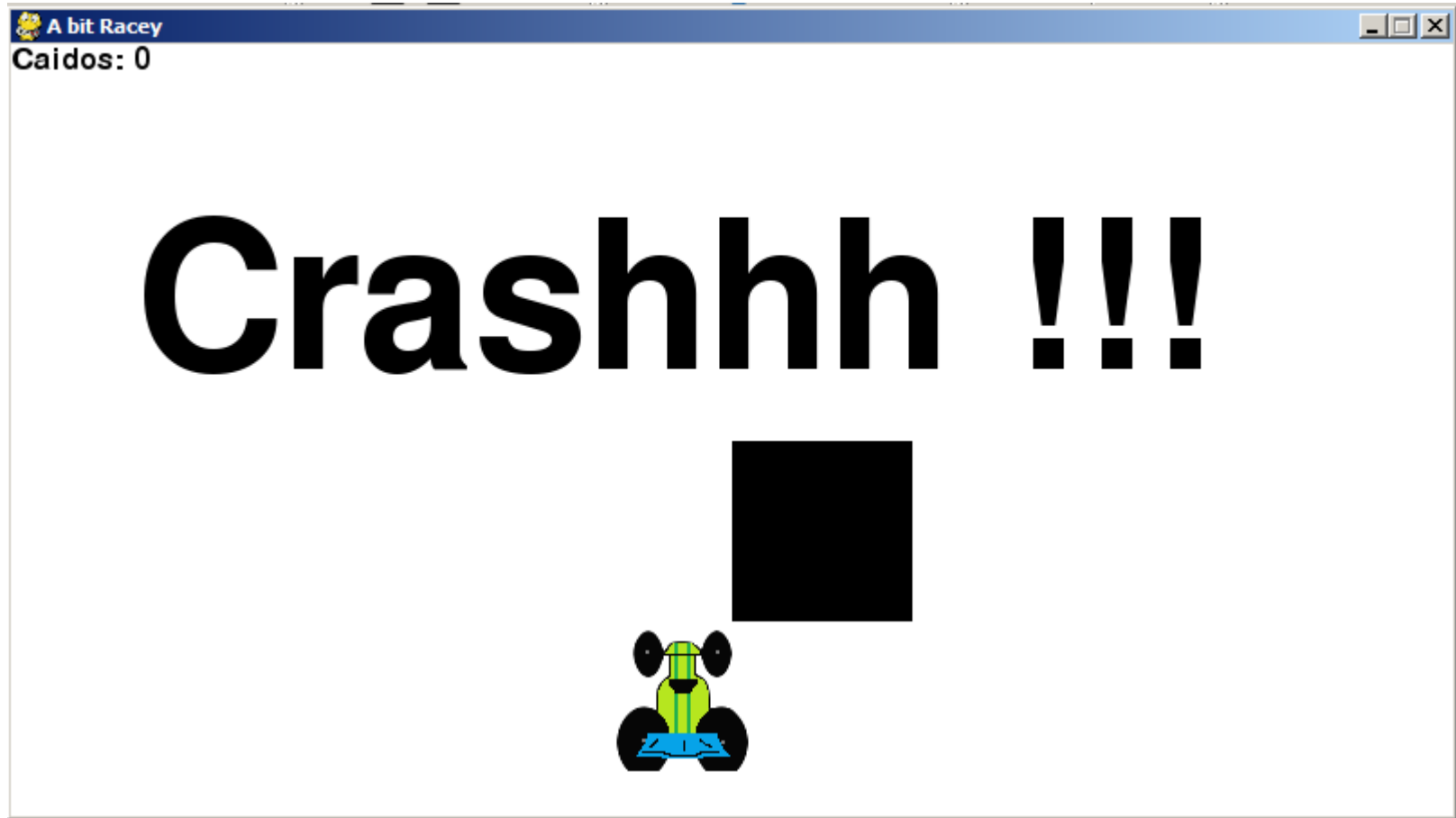
def crash():
    message_display('Crashhh !!!')

def cajas(cajax, cajay, cajaw, cajah, color):
    pygame.draw.rect(gameDisplay, color, [cajax, cajay, cajaw, cajah])

def cajas_caidas(count):
    font = pygame.font.SysFont(None, 25)
    text = font.render("Caidos: "+str(count), True, black)
    gameDisplay.blit(text, (0,0))
```

#----- Proceso Principal

Vamos a comentar las funciones añadidas



1.4-Funciones para mostrar mensajes

Añadir texto centrado y con formato

#----- Definición de funciones

```
def car(x,y):
    gameDisplay.blit(carImg, (x,y))

def text_objects(text, font):
    textSurface = font.render(text, True, black)
    return textSurface, textSurface.get_rect()

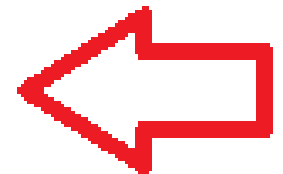
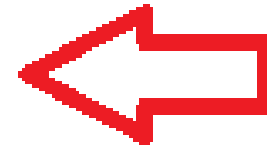
def message_display(text):
    largeText = pygame.font.Font('freesansbold.ttf',115)
    TextSurf, TextRect = text_objects(text, largeText)
    TextRect.center = ((display_width/2),(display_height/2))
    gameDisplay.blit(TextSurf, TextRect)

    pygame.display.update()

    time.sleep(2)

    game_loop()
```

```
import pygame
import time
```

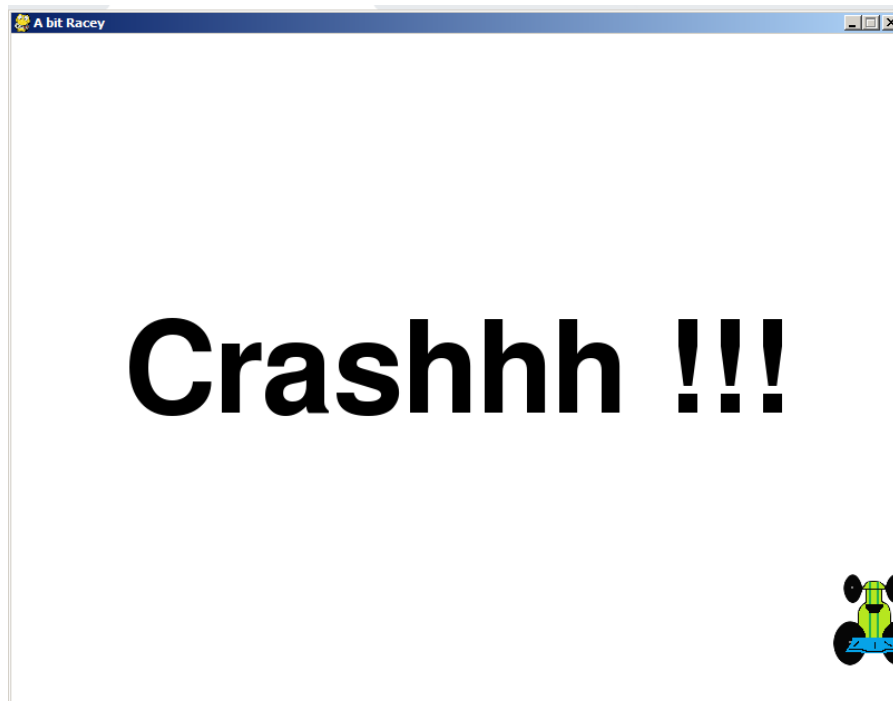



```
def crash():  
    message_display('Crashhh !!!')
```

```
if x > display_width - car_width or x < 0:  
    acabado = True  
    crash()
```

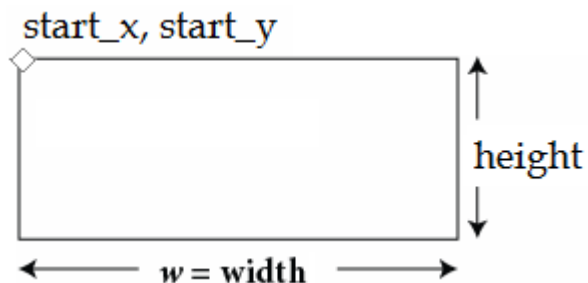


Se usa aquí



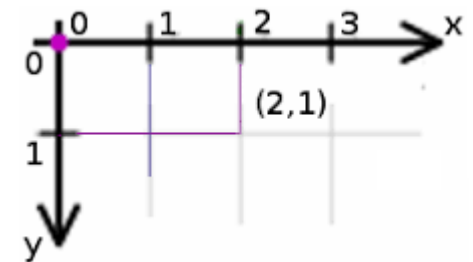
1.5- Añadiendo cajas que caen

```
def cajas(cajax, cajay, cajaw, cajah, color):  
    pygame.draw.rect(gameDisplay, color, [cajax, cajay, cajaw, cajah])
```



`import random`

```
#---- Cajas que caen del cielo  
caja_startx = random.randrange(0, display_width)  
caja_starty = -600  
caja_speed = 7  
caja_width = 100  
caja_height = 100
```



```
#----- Bucle Principal
```

```

gameDisplay.fill(white)      # Blanquea la pantalla
x += x_change                # recalcula la coordenada x
car(x,y)                     # Dibuja el coche

                                # Dibuja las caja que cae y recalcula su altura
cajas(caja_startx, caja_starty, caja_width, caja_height, black)
caja_starty += caja_speed

```



Ayuda por si te has atascado:

```
import random
```

```

                                #---- Cajas que caen del cielo
caja_startx = random.randrange(0, display_width)
caja_starty = -600
caja_speed = 7
caja_width = 100
caja_height = 100

```

```

                                # Dibuja las caja que cae y recalcula su altura
cajas(caja_startx, caja_starty, caja_width, caja_height, black)
caja_starty += caja_speed

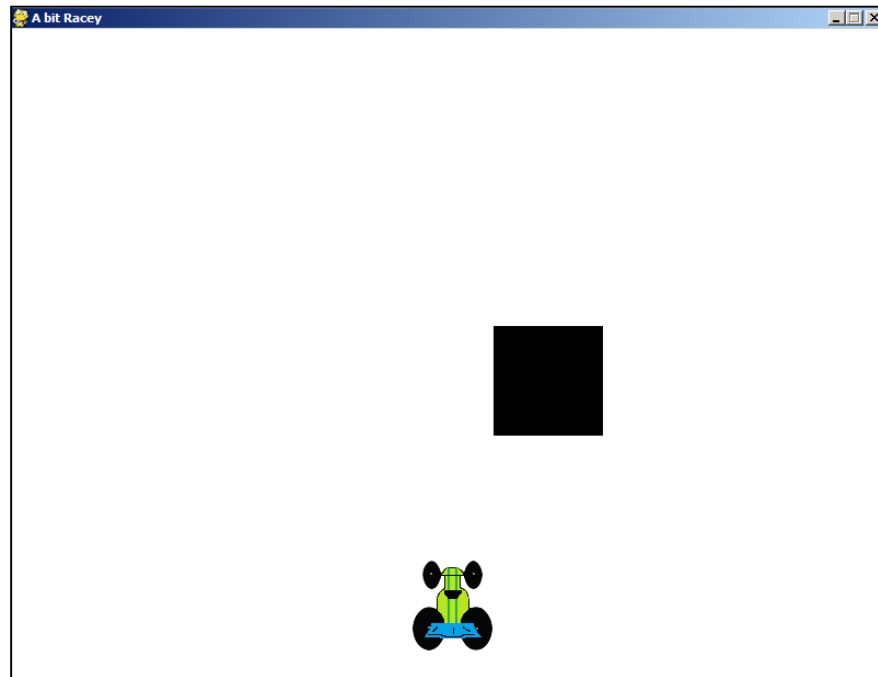
```

Ahora probamos el programa. Funciona bien ?

Que deberíamos cambiar, como lo hacemos ?

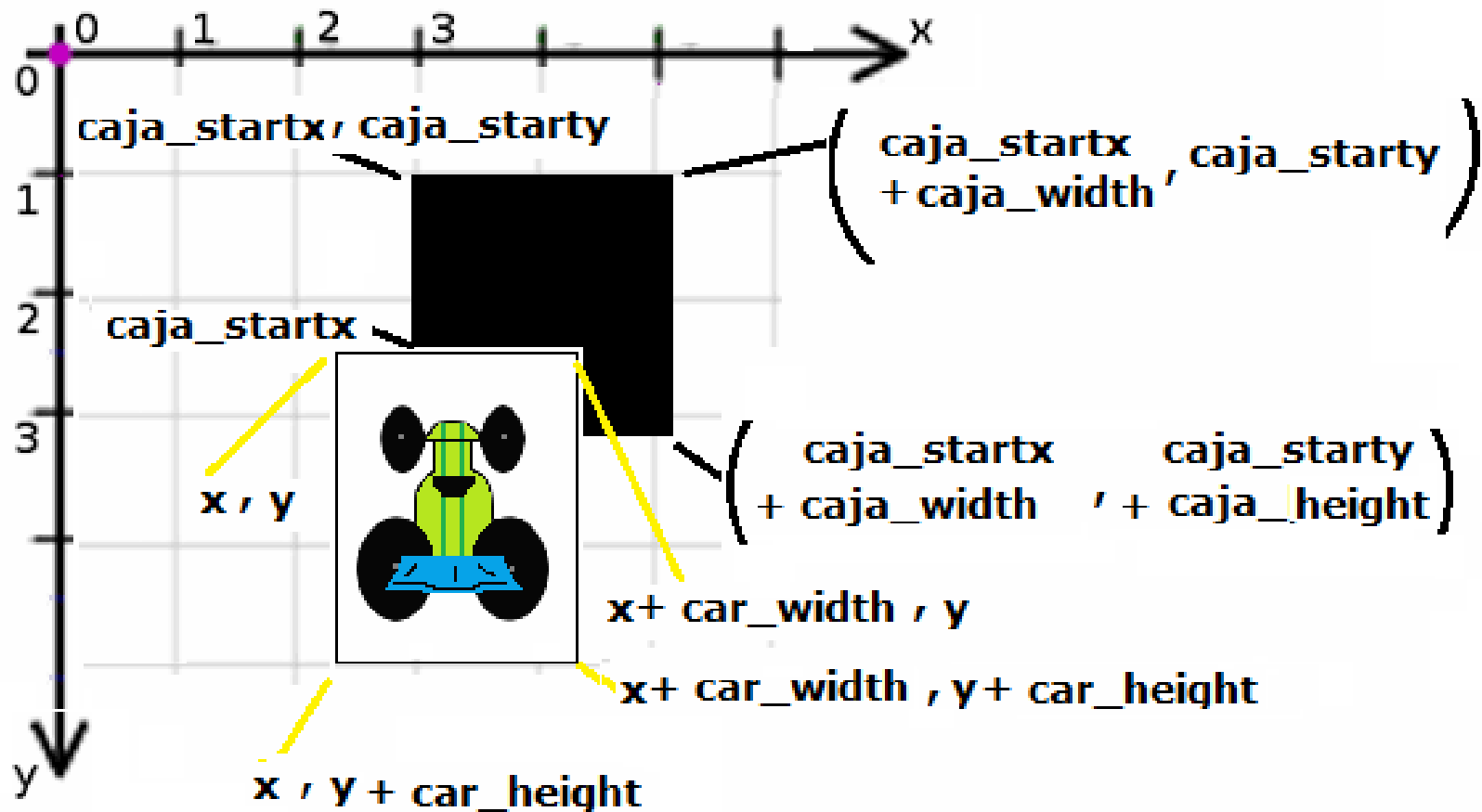
```
        # Dibuja las caja que cae y recalcula su altura
cajas(caja_startx, caja_starty, caja_width, caja_height, black)
caja_starty += caja_speed

if caja_starty > display_height:
    caja_starty = 0 - caja_height
    caja_startx = random.randrange(0,display_width)    # coordenada aleatoria
```

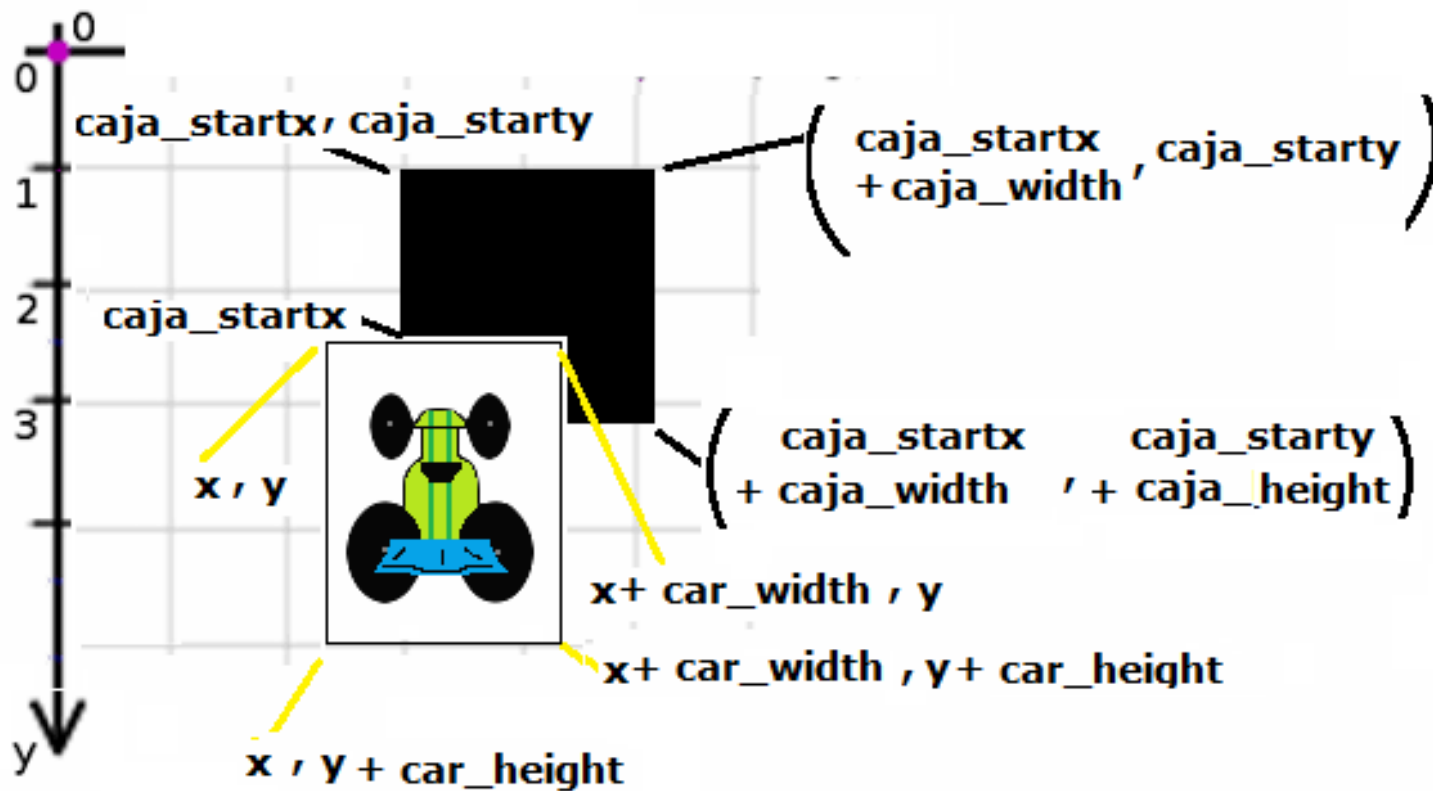


Probamos el programa. Funciona bien ?

1.6- Programando la colisión



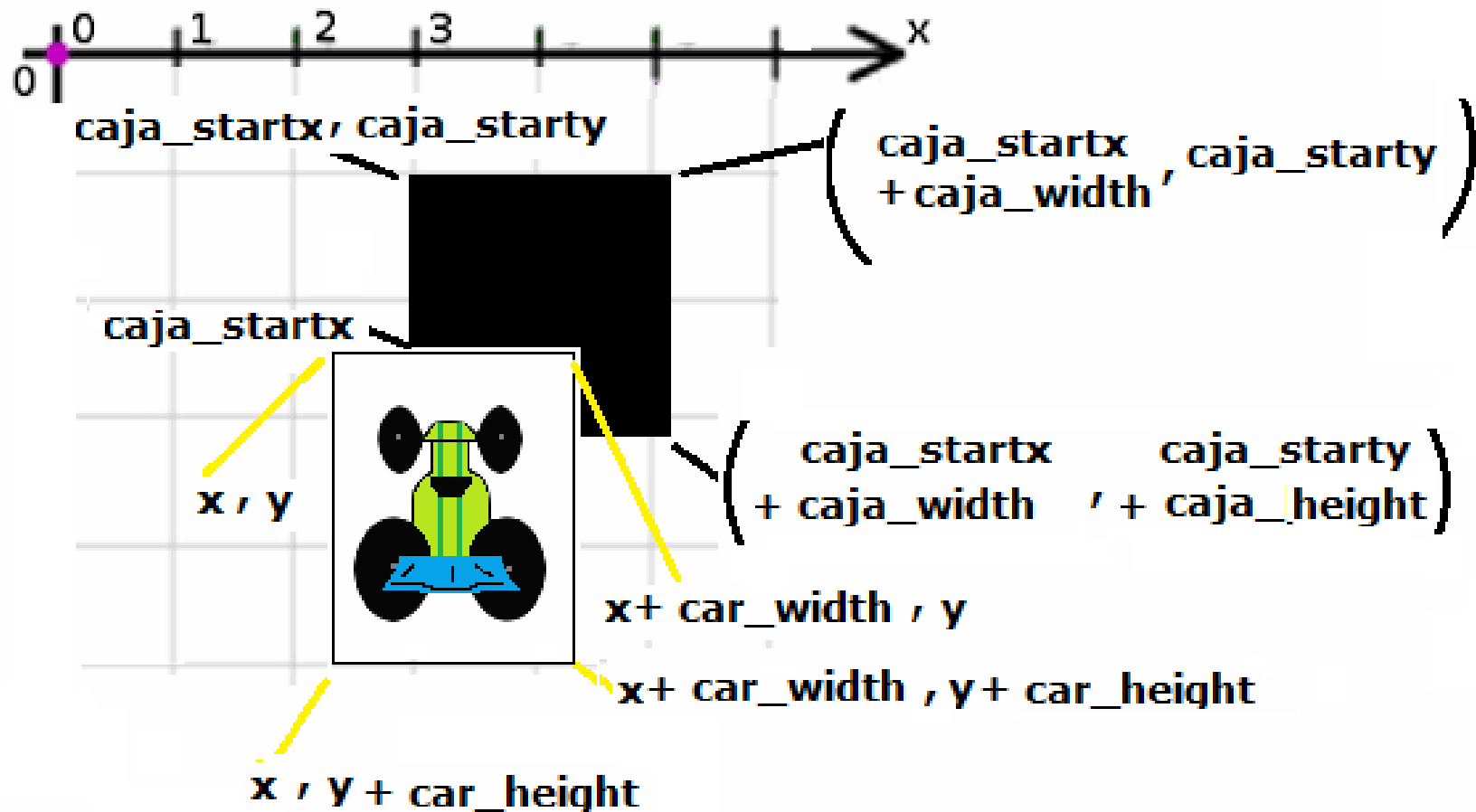
<https://pythonprogramming.net/pygame-crashing-objects/?completed=/drawing-objects-pygame-tutorial/>



La coordenada (y) del coche está más arriba que el final de la caja o dicho de otra manera ...

= El final de la caja ha sobrepasado el coche

```
if y < (caja_starty + caja_height) :  
    print('y se solapa')
```



```
if (x > caja_startx and x < (caja_startx + caja_width)) or
    ((x + car_width > caja_startx) and (x + car_width) < (caja_startx + caja_width)) :

    print('x se solapa')
    crash()
```

1.7- Mostrando puntuaciones

Añadiremos una variable para contar las cajas caídas y lo consideraremos como puntuación.

```
numCaidas = 0    # El contador de cajas servirá como puntuación
```

Y creamos una nueva función que muestra el contador en la parte superior de la pantalla

```
def cajas_caidas(count):  
    font = pygame.font.SysFont(None, 25)  
    text = font.render("Caidos: "+str(count), True, black)  
    gameDisplay.blit(text, (0,0))
```



Ahora programamos lo que vamos a realizar en cada vuelta del ciclo. Mostrar la puntuación y aumentar las cajas caídas, cuando desaparecen por abajo.

```
if caja_starty > display_height:           # Si la caja se sale de la pantalla
    caja_starty = 0 - caja_height           # se vuelve a situar arriba en otra
    caja_startx = random.randrange(0,display_width) # coordenada aleatoria
    numCaídas += 1                          # se suma 1 a las cajas caídas
    caja_speed += 1                         # se aumenta la velocidad
    caja_width += (numCaídas * 1.2)         # se va aumentando la anchura de la caja

if y < (caja_starty + caja_height) :
    if (x > caja_startx and x < (caja_startx + caja_width))
    or ((x + car_width > caja_startx) and (x + car_width) < (caja_startx + caja_width)) :
        acabado = True
        crash()

if x > display_width - car_width or x < 0:
    acabado = True
    crash()

cajas_caídas(numCaídas)
```



Felicidades !!!!

- Ya tenemos el juego funcionando.
- Claro que podemos seguir introduciendo mejoras...



Practica P01 – Ejercicio 1

Estudia el juego y analiza si la función `colliderect` puede simplificar el cálculo de la colisión

<https://www.pygame.org/docs/ref/rect.html#pygame.Rect.colliderect>

En la web de PyGame encontrarás la clase `Rect` y los métodos de que dispone.

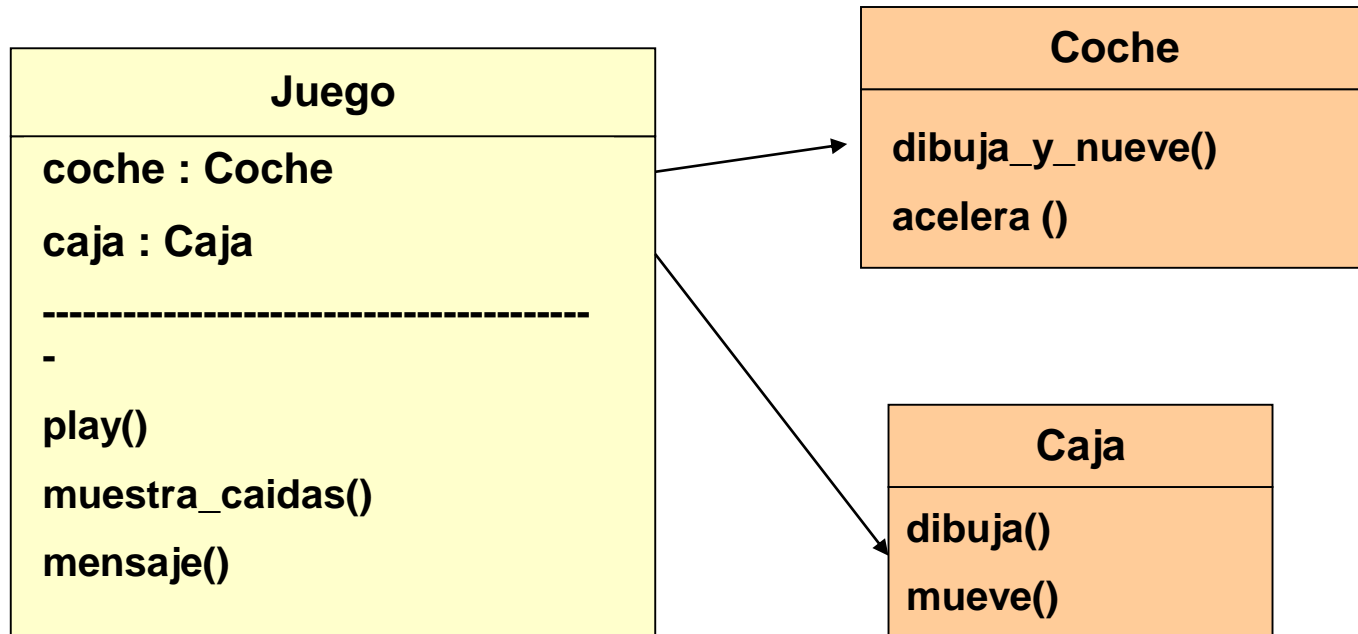
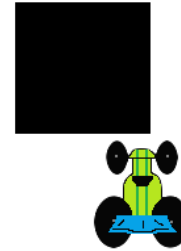
```
Rect(left, top, width, height) -> Rect
Rect((left, top), (width, height)) -> Rect
Rect(object) -> Rect
```

Practica P01 – Ejercicio 2 (Avanzado)

- Realiza los siguientes cambios en el juego `race_car`
 - Disminuye el punto de partida (y) de las cajas que caen.
 - Disminuye la velocidad de caída y el incremento de tamaño.
 - Asigna a las cajas que caen un color diferente cada vez que aparezcan desde arriba.
 - Cambia el sistema de puntuación
 - Cambia la imagen del coche (ojo al tipo de gráfico)!!!
 - Introduce otras mejoras que se te ocurran.

Race_Car : Jugando con objetos

TOCADO !!!



```
#----- DEFINICIONES GLOBALES
#

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

#----- DEFINICIONES DE CLASE
#

class Coche(object):
    pass

class Caja(object):
    pass

class Juego(object):
    def __init__(self):
        pass

    def play(self):
        pass

#----- INICIO DEL PROGRAMA
#
if __name__ == "__main__":
    Juego().play()
    quit()
```

```

class Juego(object):

    numCaidas = 0    # El contador de cajas será la puntuació
    ancho = 800
    alto = 600

    def __init__(self):
        pygame.init()
        self.ventana = pygame.display.set_mode((self.ancho, self.alto))
        pygame.display.set_caption('Pequeña carrera')
        self.ventana.fill(WHITE)    # Blanquea la pantalla

    def play(self):

        clock = pygame.time.Clock()    # Activa el reloj
        coche = Coche(self)    # Crea un objeto de tipo coche
        caja = Caja(self)    # Crea un objeto de tipo caja

        acabado = False
        #----- bucle del juego
        while acabado == False:
            #----- bucle evento
            for event in pygame.event.get():

                if event.type == pygame.QUIT:
                    acabado = True

                if event.type == pygame.KEYDOWN:    # Pulsan una tecla
                    if event.key == pygame.K_LEFT:
                        coche.acelera(-5)
                    elif event.key == pygame.K_RIGHT:
                        coche.acelera(5)

                if event.type == pygame.KEYUP:    # Sueltan una tecla
                    if event.key == pygame.K_LEFT or \
                       event.key == pygame.K_RIGHT:
                        coche.acelera(0)

            #----- fin bucle evento
            #

```

```
class Coche(object):

    ancho = 73      # Ancho del coche
    x_suma = 0      # parámetros para desplazamiento
    velocidad = 0   # y velocidad

    def __init__(self, juego):
        self.x = (juego.ancho * 0.45)    # parámetros de posición
        self.y = (juego.alto * 0.8)
        self.img = pygame.image.load('.\\img\\racecar.png')

    def dibuja_y_mueve (self, juego):
        juego.ventana.fill(WHITE)        # Blanquea la pantalla
        self.x += self.x_suma             # recalcula la coordenada x
        juego.ventana.blit(self.img, (self.x,self.y)) # Dibuja el coche

    def acelera (self, valor):
        self.x_suma = valor
```



```

class Caja(object):
    ancho = 100
    alto = 100
    speed = 1

    def __init__(self, juego):
        self.x = random.randrange(0, juego.ancho)
        self.y = -600
        self.dibuja (juego)

    def dibuja (self, juego):
        pygame.draw.rect(juego.ventana, BLACK, [self.x, self.y, self.ancho, self.alto])

    def mueve (self, juego):
        #----- baja la caja
        self.y += self.speed

        # Si la caja sale por abajo sin choque reaparece por arriba
        if self.y > juego.alto:
            self.y = 0 - self.alto
            self.x = random.randrange(0,juego.ancho)
            juego.numCaidas += 1
            self.speed += 1
            self.ancho += (juego.numCaidas * 1.2)

            # Si la caja se sale de la pantalla
            # se vuelve a situar arriba en otra
            # coordenada aleatoria
            # se suma 1 a las cajas caidas
            # se aumenta la velocidad
            # va aumentando la anchura de la caja

```

... acabando la clase juego

```
#----- fin bucle evento
#

coche.dibuja_y_mueve(self)

#-----
# Dibuja la caja que cae y recalcula su altura

caja.dibuja (self)
caja.mueve (self)

#----- Calcula colisión coche-caja
if coche.y < (caja.y + caja.alto) :
    if (coche.x > caja.x and coche.x < (caja.x + caja.ancho)) or \
        ((coche.x + coche.ancho > caja.x) and \
         (coche.x + coche.ancho) < (caja.x + caja.ancho)) :
        acabado = True
        self.mensaje('TOCADO !!!')

#----- Calcula si coche sale por los margenes pantalla
if coche.x > (self.ancho - coche.ancho) or coche.x < 0:
    acabado = True
    self.mensaje('FUERA DE PISTA !!!')

self.muestra_caidas(self.numCaidas)
pygame.display.update()
clock.tick(60)

#----- fin bucle del juego
pygame.quit()
time.sleep(50)
```

```
def muestra_caidas(self, count):
    font = pygame.font.SysFont(None, 25)
    text = font.render("Caidos: "+str(count), True, BLACK)
    self.ventana.blit(text, (0,0))

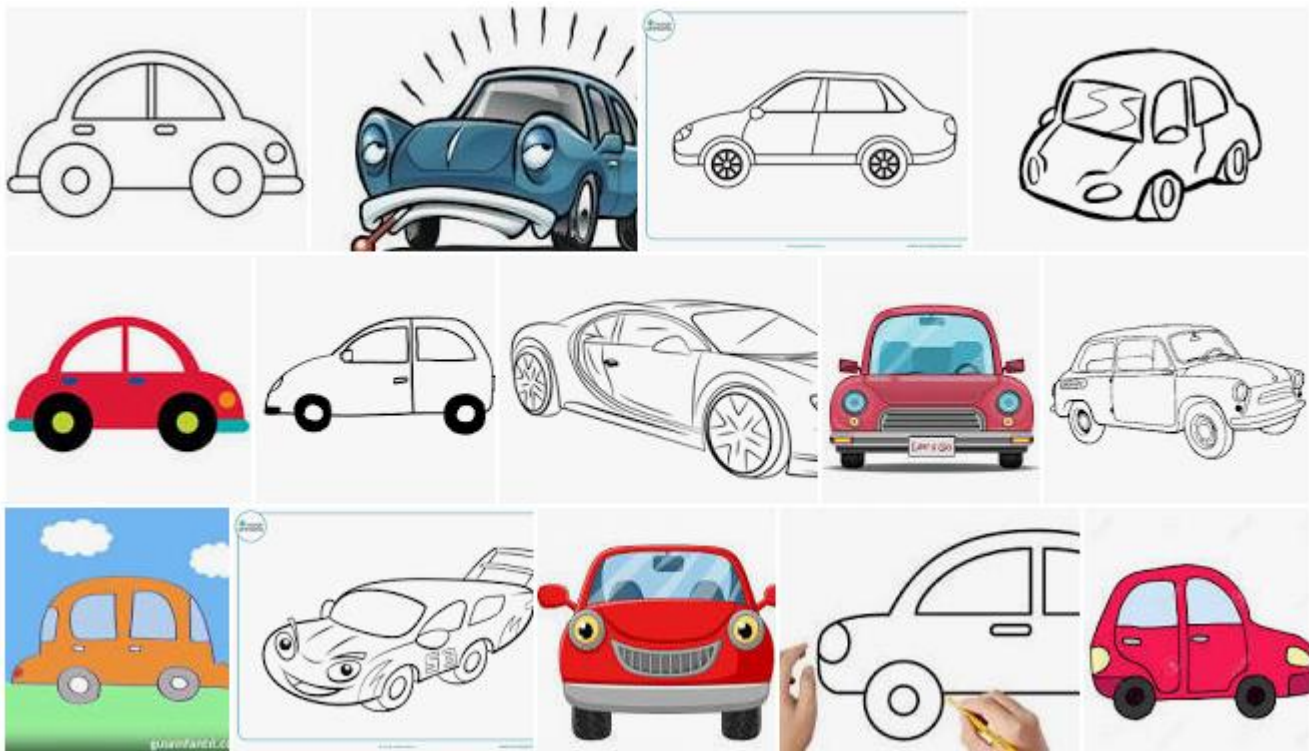
def text_objects(self, text, font, color):
    textSurface = font.render(text, True, color)
    return textSurface, textSurface.get_rect()

def mensaje(self, text):
    largeText = pygame.font.Font('..\\fonts\\freesansbold.ttf', 80)
    TextSurf, TextRect = self.text_objects(text, largeText, BLACK)
    TextRect.center = ((self.ancha/2), (self.alto/2))
    self.ventana.blit(TextSurf, TextRect)
    pygame.display.update()
    time.sleep(2)

#----- fin clase : Juego
```

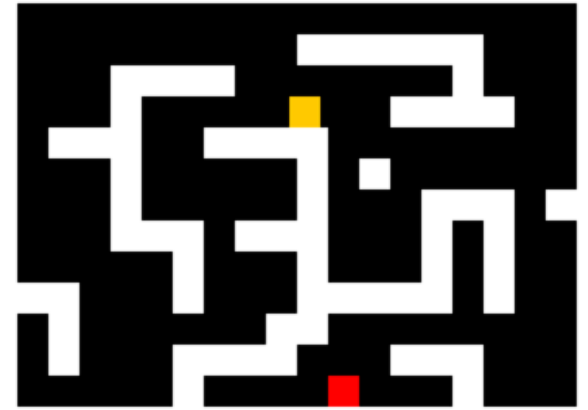
Práctica P05- Race-Car

- Crea una versión propia de [race_oop.py](#)
- Adapta el juego para que caigan dos cajas de distinto color y con distinta posición inicial.



2. Collision Response

- En esta url hay un juego de demo para estudiar las colisiones laterales.
- Para ello dibuja un laberinto a partir de un esquema de texto.
- Es interesante echarle un ojo al programa.



<https://www.pygame.org/project-Rect+Collision+Response-1061-.html>

Rectifica la linea 111:

```
En lugar de raise SystemExit, "You win!")  
Haz  print ("You win!")  
      raise SystemExit
```