Módulo 3: UI y DDBB

Desarrollo en lenguaje Python

Año de realización: 2021

PROFESORA
Carmen Bartolomé Valentín-Gamazo

www.eoi.es

BY NC SA

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Tema 12-3

Interfaz de usuario

Índice

CC BY NC SA

Caso práctico: editor de texto

Una guía de apoyo, en tres video tutoriales:

CC BY NC SA

Empezamos por crear el menú superior y también creamos un campo de texto central que será el cuerpo principal de la aplicación

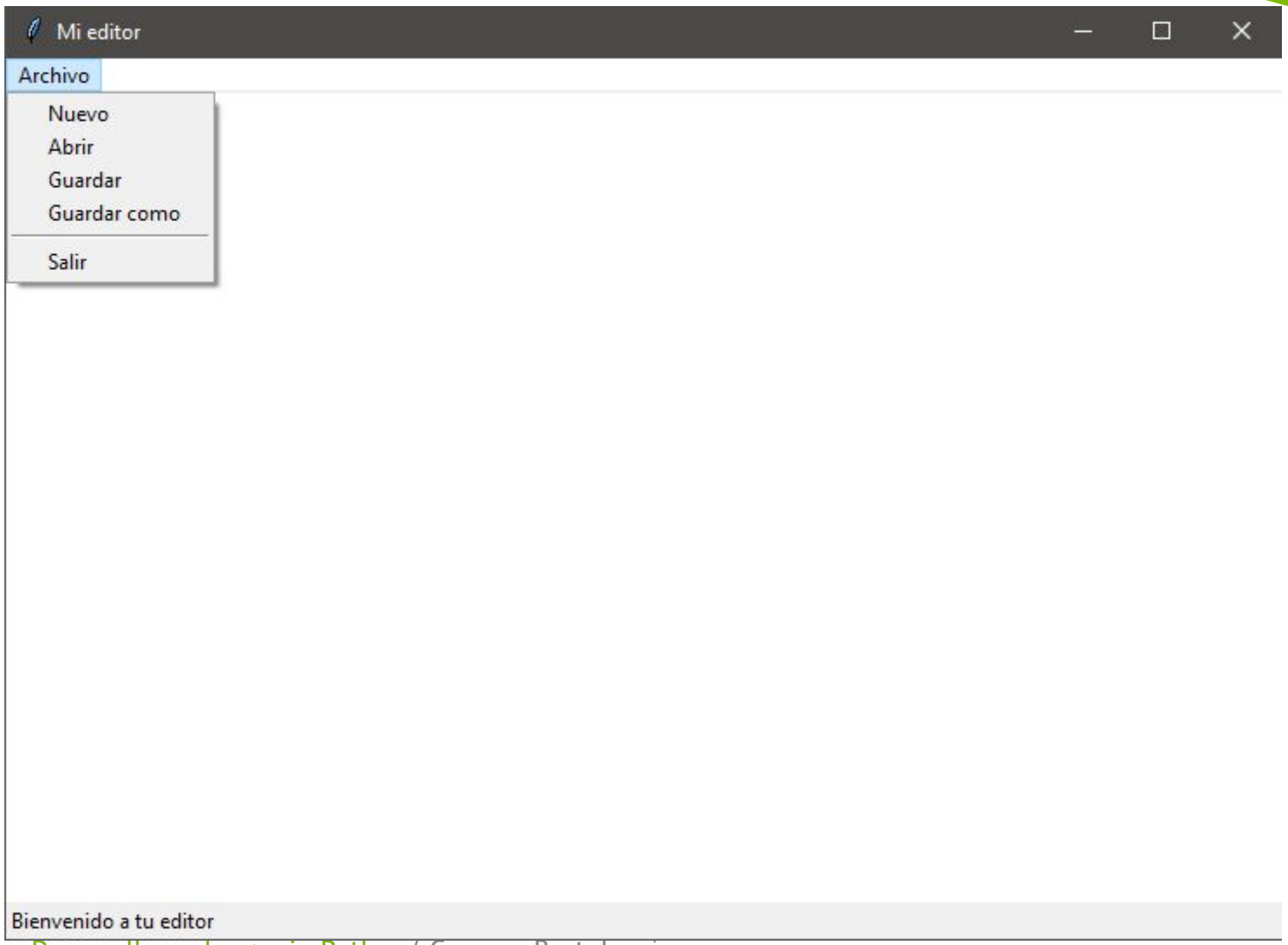
```python
from tkinter import *
root = Tk()
root.title("Mi editor")

# Menú superior
menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="Nuevo")
filemenu.add_command(label="Abrir")
filemenu.add_command(label="Guardar")
filemenu.add_command(label="Guardar como")
filemenu.add_separator()
filemenu.add_command(label="Salir", command=root.quit)
menubar.add_cascade(label="Archivo", menu=filemenu)

# Caja de texto central
texto = Text(root)
texto.pack(fill='both', expand=1)
texto.config(padx=6, pady=4, bd=0, font=("Consolas", 12))

# Menu y bucle de la aplicación
root.config(menu=menubar)
root.mainloop()
```


CC BY NC SA

Añadimos una etiqueta abajo a la izquierda, para ir mostrando información sobre la tarea que se está llevando a cabo, que llamaremos “monitor”

```
# Monitor inferior
mensaje = StringVar()
mensaje.set("Bienvenido a tu Editor")
monitor = Label(root, textvar=mensaje, justify='left')
monitor.pack(side="left")
```


CC BY NC SA


Mi editor
Archivo
Nuevo
Abrir
Guardar
Guardar como
Salir
Bienvenido a tu editor

Para la parte de la lógica de programación, necesitaremos definir una función para cada una de las opciones principales del menú. De momento, incluimos el mensaje que se mostraría abajo

```
1   from tkinter import *
2
3   def nuevo():
4       mensaje.set('Nuevo fichero')
5
6   def abrir():
7       mensaje.set('Abrir fichero')
8
9   def guardar():
10      mensaje.set('Guardar fichero')
11
12  def guardar_como():
13      print("Guardar fichero como")
14
15  root = Tk()
16  root.title("Mi editor")
```


CC BY NC SA

Y actualizamos el menú incluyendo command con las llamadas a estas funciones

```
1   from tkinter import *
2
3   def nuevo():
4       mensaje.set('Nuevo fichero')
5
6   def abrir():
7       mensaje.set('Abrir fichero')
8
9   def guardar():
10      mensaje.set('Guardar fichero')
11
12  def guardar_como():
13      print("Guardar fichero como")
14
15  root = Tk()
16  root.title("Mi editor")
17
18  # Menú superior
19  menubar = Menu(root)
20  filemenu = Menu(menubar, tearoff=0)
21  filemenu.add_command(label="Nuevo", command=nuevo)
22  filemenu.add_command(label="Abrir", command=abrir)
23  filemenu.add_command(label="Guardar", command=guardar)
24  filemenu.add_command(label="Guardar como", command=guardar_como)
25  filemenu.add_separator()
26  filemenu.add_command(label="Salir", command=root.quit)
27  menubar.add_cascade(menu=filemenu, label="Archivo")
```

Escuela de organización industrial

i.es

BY NC SA

Creamos la variable global “ruta” y programamos que el campo de texto se borre cuando se ejecuta la función nuevo

```
from tkinter import *

ruta = "" # La utilizaremos para almacenar la ruta del fichero

def nuevo():
    mensaje.set('Nuevo fichero')
    texto.delete(1.0, END)  # En flotante, el primer carácter es un salto
```


CC BY NC SA

Importamos filedialog para la ventana de apertura de ficheros y open para poder cargarlo, y actualizamos la función abrir. Asignaremos a la variable ruta el resultado de pedir al usuario que seleccione el fichero gracias a AskOpenFileName

```
from tkinter import *
from tkinter import filedialog
from io import open

ruta = "" # La utilizaremos para almacenar la ruta del fichero

def nuevo():
    mensaje.set('Nuevo fichero')
    texto.delete(1.0, END)  # En flotante, el primer carácter es un salto

def abrir():
    global ruta
    mensaje.set("Abrir fichero")
    ruta = filedialog.askopenfilename(
        initialdir='.',
        filetype=(("Ficheros de texto", "*.txt"),),
        title="Abrir un fichero de texto")

    if ruta != "":
        fichero = open(ruta, 'r')
        contenido = fichero.read()
        texto.delete(1.0,'end')
        texto.insert('insert', contenido)
        fichero.close()
        root.title(ruta + " - Mi editor")
```

Actualizamos la función nuevo para que se reinicie “ruta” si creamos un nuevo fichero. Si no, no podríamos distinguir un fichero nuevo de uno que se ha abierto desde el disco duro

```
def nuevo():
    global ruta
    mensaje.set("Nuevo fichero")
    ruta = ""
    texto.delete(1.0, "end")
    root.title("Mi editor")
```


www.eoi.es

CC BY NC SA

Actualizamos también la función guardar, teniendo en cuenta la opción de que se trate de un fichero ya existente (primera condicion del if), o que se trate de un fichero nuevo (en la parte else, llamando a la función guardar_como(), que tenemos que definir del todo a continuación)

```
def guardar():
    mensaje.set("Guardar fichero")
    if ruta != "":
        contenido = texto.get(1.0,'end-1c')
        fichero = open(ruta, 'w+')
        fichero.write(contenido)
        fichero.close()
        mensaje.set("Fichero guardado correctamente")
    else:
        guardar_como()
```


CC BY NC SA

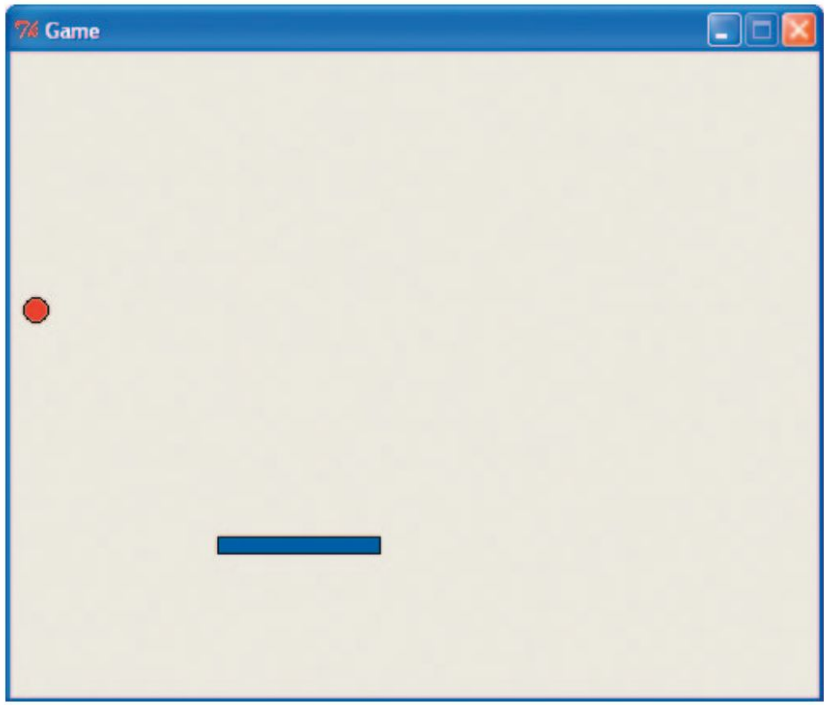
Por último, actualizamos la función guardar_como con el widget AskSaveAsFile

```
def guardar_como():
    global ruta
    mensaje.set("Guardar fichero como")
    fichero = filedialog.asksaveasfile(title="Guardar fichero", mode="w", defaultextension=".txt")
    if fichero is not None:
        ruta = fichero.name
        contenido = texto.get(1.0,'end-1c')
        fichero = open(ruta, 'w+')
        fichero.write(contenido)
        fichero.close()
        mensaje.set("Fichero guardado correctamente")
    else:
        mensaje.set("Guardado cancelado")
        ruta = ""
```


CC BY NC SA

Caso práctico: mini juego bounce

Una de las aplicaciones de tkinter es crear aplicaciones de dibujo y juegos



BY NC SA

Empezamos por importar tkinter, random y time. Además de crear la ventana raíz, tenemos que crear un canvas, que es otro widget de tkinter, pero orientado a dibujo

```python
from tkinter import *
import random
import time

# Crea la ventana raíz
root = Tk()
root.title("Game")
root.resizable(0, 0)  # No se podrá redimensionar manualmente
root.wm_attributes("-topmost", 1)  # La ventana se posiciona al frente de las demás

# Crea el canvas del juego
canvas = Canvas(root, width=500, height=400, bd=0, highlightthickness=0)
canvas.pack()

# Añade el método mainloop para mantener la ejecución
root.mainloop()
```


www.eoi.es

Para la pelota, recurrimos a crear una clase:

```
# Crea la clase para la pelota
class Ball:
    def __init__(self,canvas,color):
        self.canvas = canvas
        self.id = canvas.create_oval(10,10,25,25,fill=color)
        self.canvas.move(self.id,245,100)

    def draw(self):
        pass
```


www.eoi.es

CC BY NC SA

Cambiamos el método mainloop por un bucle while que va a estar recargando la ventana raíz y redibujando la pelota en intervalos de centésimas de segundo, para conseguir efectos de animación

```
# Loop principal
while 1:

    ball.draw()
    root.update()
    time.sleep(0.01)
```

Por otro lado, completamos el método draw utilizando el identificador que habíamos definido previamente

```
def draw(self):
    self.canvas.move(self.id, 0, -1)
```


CC BY NC SA

Añadimos root.update() también para que se actualice el canvas.

```
# Crea el canvas del juego
canvas = Canvas(root, width=500, height=400, bd=0, highlightthickness=0)
canvas.pack()
root.update()
```


www.eoi.es

CC BY NC SA

Actualizamos el método __init__ y el draw, trabajando ahora con la altura del canvas y con las coordenadas de la pelota (pos es una lista de 4 números: x1, y1, x2, y2, que corresponden a la x y de la esquina superior izq. y las de la esquina inf. derecha respectivamente.

```
# Crea la clase para la pelota
class Ball:
    def __init__(self,canvas,color):
        self.canvas = canvas
        self.id = canvas.create_oval(10,10,25,25,fill=color)
        self.canvas.move(self.id,245,100)
        self.x = 0
        self.y = -1
        self.canvas_height = self.canvas.winfo_height()

    def draw(self):
        self.canvas.move(self.id, self.x, self.y)
        pos = self.canvas.coords(self.id)

        if pos[1] <= 0:
            self.y = 1
        if pos[3] >= self.canvas_height:
            sel.y = -1
```

En __init__ cambiamos self.x=0 y self.y=-1 para que no esté la pelota rebotando todo el rato hacia arriba y hacia abajo.
Creamos una lista de varios valores de los que se usará uno al azar para dar velocidad en x a la pelota, y aumentamos la velocidad de y pasando de -1 a -3.

Por otro lado, para que la pelota también rebote si toca los bordes laterales, añadimos self.canvas_width, que usaremos en draw()

```
def __init__(self,canvas,color):
    self.canvas = canvas
    self.id = canvas.create_oval(10,10,25,25,fill=color)
    self.canvas.move(self.id,245,100)
    starts = [-3,-2,-1,1,2,3]
    random.shuffle(starts)
    self.x = starts[0]
    self.y = -3
    self.canvas_height = self.canvas.winfo_height()
    self.canvas_width = self.canvas.winfo_width()
```


CC BY NC SA

Actualizamos la definición del método draw() con las nuevas instrucciones de rebotar en el borde para x

```
# Crea la clase para la pelota
class Ball:
    def __init__(self,canvas,color):
        self.canvas = canvas
        self.id = canvas.create_oval(10,10,25,25,fill=color)
        self.canvas.move(self.id,245,100)
        starts = [-3,-2,-1,1,2,3]
        random.shuffle(starts)
        self.x = starts[0]
        self.y = -3
        self.canvas_height = self.canvas.winfo_height()
        self.canvas_width = self.canvas.winfo_width()

    def draw(self):
        self.canvas.move(self.id, self.x, self.y)
        pos = self.canvas.coords(self.id)

        if pos[1] <= 0:
            self.y = 1
        if pos[3] >= self.canvas_height:
            self.y = -1
        if pos[0] <= 0:
            self.x = 3
        if pos[2] >= self.canvas_width:
            self.x = -3
```


CC BY NC SA

Ahora, a continuación de la definición de la clase de la pelota, creamos la clase para la paleta, que será un rectángulo de 100x10

```
class Paddle:
    def __init__(self,canvas,color):
        self.canvas = canvas
        self.id = canvas.create_rectangle(0,0,100,10,fill=color)
        self.canvas.move(self.id,200,300)

    def draw(self):
        pass
```


www.eoi.es

CC BY NC SA

Creamos el objeto paleta debajo de donde teníamos creada la pelota, y también introducimos en el bucle while la instrucción para que se muestre con el método draw()

```
# Crea un objeto pelota
ball = Ball(canvas,'red')
# Crea un objeto paleta
paddle = Paddle(canvas,'blue')

# Loop principal
while 1:

    ball.draw()
    paddle.draw()
    root.update()
    time.sleep(0.01)
```


CC BY NC SA

Añadimos la variable x al __init__() así como la variable necesaria para programar después que la paleta no se vaya a por uvas.

```
class Paddle:
    def __init__(self,canvas,color):
        self.canvas = canvas
        self.id = canvas.create_rectangle(0,0,100,10,fill=color)
        self.canvas.move(self.id,200,300)
        self.x = 0
        self.canvas_width = self.canvas.info_width()
```

También añadimos (debajo de draw()), los métodos turn_left() y turn_right() que nos permitirán mover la paleta

```
def turn_left(self,ev):
    self.x = -2

def turn_right(self,ev):
    self.x = 2
```


www.eoi.es

CC BY NC SA

Actualizamos el __init__ para incluir la reacción (a través de bind) a los eventos de pulsar la tecla flecha izq o derecha.

```
class Paddle:
    def __init__(self,canvas,color):
        self.canvas = canvas
        self.id = canvas.create_rectangle(0,0,100,10,fill=color)
        self.canvas.move(self.id,200,300)
        self.x = 0
        self.canvas_width = self.canvas.info_width()
        self.canvas.bind_all('<KeyPress-Left>',self.turn_left)
        self.canvas.bind_all('<KeyPress-Right>',self.turn_right)

    def draw(self):
        pass

    def turn_left(self,ev):
        self.x = -2

    def turn_right(self,ev):
        self.x = 2
```

FInalmente, completamos el método draw() de una manera muy similar a la del de la pelota, pero esta vez solo para x

```
def draw(self):
    self.canvas.move(self.id,self.x,0)
    pos = self.canvas.coords(self.id)
    if pos[0] <= 0:
        self.x = 0
    elif pos[2] >= self.canvas_width:
        self.x = 0
```


CC BY NC SA

Para poder programar el efecto de interacción entre pelota y paleta, tenemos que incluir en la clase de la pelota, la paleta como uno de los argumentos. Y actualizar los argumentos cuando creamos el objeto. ¡Ojo! también hay que cambiar el orden en que se crean los objetos, pues necesitamos que se cree primero “paddle” para poder introducirlo después en “ball”

```
class Ball:
    def __init__(self,canvas,paddle,color):
        self.canvas = canvas
        self.paddle = paddle
```

```
# Crea un objeto paleta
paddle = Paddle(canvas,'blue')
# Crea un objeto pelota
ball = Ball(canvas,paddle,'red')
```


www.eoi.es

CC BY NC SA

Creamos el método hit_paddle() para detectar si se está tocando la paleta con la pelota y la llamamos dentro de draw() para dar el movimiento de rebote si la toca

```
def hit_paddle(self,pos):
    paddle_pos = self.canvas.coords(self.paddle.id)
    if pos[2] >= paddle_pos[0] and pos[0] <= paddle_pos[2]:
        if pos[3] >= paddle_pos[1] and pos[3] <= paddle_pos[3]:
            return True
    return False

def draw(self):
    self.canvas.move(self.id, self.x, self.y)
    pos = self.canvas.coords(self.id)

    if pos[1] <= 0:
        self.y = 1
    if pos[3] >= self.canvas_height:
        self.y = -1
    if pos[0] <= 0:
        self.x = 3
    if pos[2] >= self.canvas_width:
        self.x = -3
    if self.hit_paddle(pos) == True:
        self.y = -3
```

Escuela de organización industrial
BY NC SA

Por último, para que el juego se acabe al tocar la pelota el borde inferior, añadimos la variable hit_bottom y la ponemos como False al crearse la pelota

```
class Ball:
    def __init__(self,canvas,paddle,color):
        self.canvas = canvas
        self.paddle = paddle
        self.id = canvas.create_oval(10,10,25,25,fill=color)
        self.canvas.move(self.id,245,100)
        starts = [-3,-2,-1,1,2,3]
        random.shuffle(starts)
        self.x = starts[0]
        self.y = -3
        self.canvas_height = self.canvas.winfo_height()
        self.canvas_width = self.canvas.winfo_width()
        self.hit_bottom = False
```


CC BY NC SA

Actualizamos el bucle while para que pelota y paleta se muevan siempre que hit_bottom siga siendo False.

Pero actualizamos draw() en la clase Ball para que hit_bottom sea True al alcanzar la coordenada del fondo inferior

```
# Loop principal
while 1:
    if ball.hit_bottom == False
        ball.draw()
        paddle.draw()
    root.update()
    time.sleep(0.01)
```

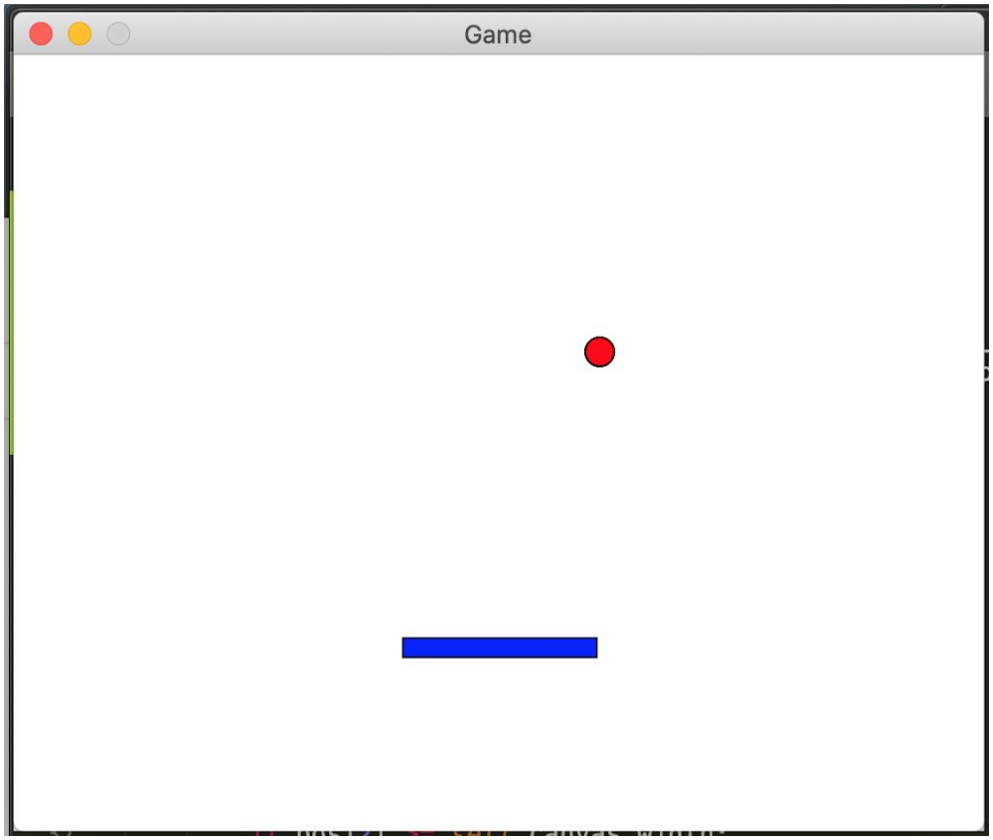
```
def draw(self):
    self.canvas.move(self.id, self.x, self.y)
    pos = self.canvas.coords(self.id)

    if pos[1] <= 0:
        self.y = 1
    if pos[3] >= self.canvas_height:
        self.y = -1
    if pos[0] <= 0:
        self.x = 3
    if pos[2] >= self.canvas_width:
        self.x = -3
    if pos[3] >= slef.canvas_height:
        self.hit_bottom = True
    if self.hit_paddle(pos) == True:
        self.y = -3
```


CC BY NC SA

Ya tenemos el juego listo. Pero hay varias opciones de mejora.

¿Se te ocurre cómo añadir un marcador de puntos?

¿Y una forma de reiniciar el juego?



CC BY NC SA

carmenbvg@gmail.com

CC BY NC SA