Programando el videojuego ”Pong.[en] Python usando la libreria ≪turtle≫

Perez Erick, Cruz Flores Angel Daniel, Nava Villavicencio Abraham, y López Bello Rebeca
Profesor: Pedro Arturo Flores Silva

04 de diciembre de 2022

***Resumen*—Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo como Instagram, Netflix, Spotify, Panda3D, entre otros.**
La finalidad de este proyecto es evaluar la cantidad de conocicciento adquirido durante el ciclo escolar universitario de los estudiantes de computación mediante la creación del juego ”Pong.[en] python, en el cual se vio el funcionamiento básico de python; la sintaxis que usa, algunos de los problemas que se pueden resolver usando la programación y otras caracteristicasutiles del mismo.

I. Introducción

Lanzado en 1972 por la compañía ≪Atari≪, el ≪Pong≫ está considerado por muchos como el principal iniciador de la industria del videojuego. Su dinámica era bien sencilla: El jugador controlaba en el juego una paleta moviéndola verticalmente en la parte izquierda de la pantalla pudiendo competir tanto contra un oponente controlado por computadora, como contra jugador humano que controlaba una segunda paleta en la parte opuesta. Dichos jugadores podían usar las paletas para pegarle a la pelota hacia un lado u otro. El objetivo consistía en que uno de los jugadores obtuviera más puntos que el oponente al finalizar el juego. Puntos que se obtenían cuando el jugador adversario fallaba al devolver la pelota. Pues bien, en el artículo de hoy nos proponemos a programar nuestra propia versión de aquel juego, en Python, valiéndonos dela librería ≪turtle≪.

II. Desarrollo experimental

Como es habitual, iniciaremos la elaboración de nuestro código realizandolas importaciones pertinentes, es decir, la librería ≪turtle≫ para los gráficos.

```
#Importamos la libreria turtle que nos permite crear gráficos
import turtle
```

Figura 1. Importamos la libreria

Realizada las importaciones de los recursos a emplear, empezaremos creando y configurando a nuestro gusto una ventana nueva que se iniciará cada vez que se ejecute el código.

Como se ve, empezamos creando el área en el que se desarrollará el juego, usando el método ≪turtle.Screen()≫ cuyo título, color de fondo y dimensiones, estableceremos mediante los métodos ≪.title()≪, ≪bgcolor()≫ y ≪setup()≫ respectivamente. Trasello, crearemos lasvariables ≪marcadorA≫ y

El presente documento corresponde a la práctica de [] presentado en la UNAM durante el periodo 2022-2

```
#Creamos la ventana donde veremos el juego
wn = turtle.Screen()
#Configuramos la ventana
wn.title("Ping-Pong - Equipo 6")
wn.bgcolor("black")
wn.setup(width = 800, height = 600)
wn.tracer(0)

#Creamos el marcador
marcadorA = 0
marcadorB = 0
```

Figura 2. Creación y configuración de la ventana y las variables marcadorA y marcadorB.

≪marcadorB≫ que con valor inicial 0, llevarán la cuenta de los puntos obtenidos por jugadores, (los cuales serán controladas mediante el teclado) que pasaremos a crear a continuación con el método ≪turtle.Turtle()≪:

```
#Creamos los objetos del juego
#Jugador A
jugadorA = turtle.Turtle()
jugadorA.speed(0)
jugadorA.shape("square")
jugadorA.color("white")
jugadorA.penup()
jugadorA.goto(-350,0)
jugadorA.shapesize(stretch_wid=3.5,
stretch_len=0.7)

#Jugador B
jugadorB = turtle.Turtle()
jugadorB.speed(0)
jugadorB.shape("square")
jugadorB.color("white")
jugadorB.penup()
jugadorB.goto(350,0)
jugadorB.shapesize(stretch_wid=3.5,
stretch_len=0.7)
```

Figura 3. Creación de los jugadores A y B

Nuestros jugadores, van a consistir en dos barras rectangulares colocadas a ambos extremos del área de juego. Por ello para su forma usaremos como base la figura del cuadrado (con ≪shape(≪square≫)≪) cuyas dimensiones definiremos con el métodо≪shapesize()≪, su color con el método ≪color()≫ y su posición inicial, con ≪goto()≪.

```
#Linea divisora
linea = turtle.Turtle()
linea.color("white")
linea.goto(0, 400)
linea.goto(0,-400)
```

Figura 4. Creación de la Linea divisora

Como observamos en la figura 4, el siguiente elemento gráfico que introduciremos es la línea divisoria del área de juego. Para ello usaremos los mismos métodos que

empleamos para los jugadores, solo que aquí usaremos un color distinto, la posición base de las misma será el centro del área de juego (≪goto(0,0)≪) y su alto y ancho será bastante mayor y menor respectivamente.

El siguiente elemento gráfico que vamos a incluir es la pelota. La cual va a consistir en un círculo posicionado inicialmente en el centro del área de juego:

```
#Pelota
pelota = turtle.Turtle()
pelota.speed(0)
pelota.shape("circle")
pelota.color("white")
pelota.penup()
pelota.goto(0,0)
pelota.dx = 0.17
pelota.dy = 0.17
```

Figura 5. Creación de la pelota

Como se ve, volvemos a usar los métodos que utilizamos para los elementos anteriores, solo que en este caso añadimos tres variables. Por una parte, las variables ≪pelota.dx≫ y ≪pelota.dy≫ con las que definiremos el espacio en pixeles que tanto en el eje ≪x≫ como en el eje ≪y≪, va a avanzar nuestra pelota en su movimiento (determinando la fluidez del mismo), en estecaso fue de 0.17.

Finalmente, el último elemento gráfico es el contador de puntos. Introduciremos dicho texto mediante el método ≪.write()≫ al que pasaremos el texto en cuestión, la alineación y la fuente tipográfica:

```
#Pen
pen = turtle.Turtle()
pen.speed(0)
pen.color("white")
pen.penup()
pen.hideturtle()
pen.goto(0,260)
pen.write(
"Jugador A: 0          Jugador B: 0",
align="center", font=("Courier", 12,
"normal"))
```

Figura 6. Creación del marcador

Con esto ya tendríamos creados, con su configuración inicial, los elementos gráficos de nuestro juego. Los cuales, en pantalla se verán así al iniciarel programa:

Figura 7. Caption

Lo que toca ahora es definir las distintas funciones que gobernarán el comportamiento de los distintos elementos creados (Jugadores, pelota y el marcador) en función de los eventos que introduzcamos a través del teclado. Empezaremos esta explicación por el control de los jugadores:

```
#Creamos las funciones para el movimiento
de los objetos del juego
#Para el jugado A
def jugadorA_up():
    y = jugadorA.ycor()
    y += 20
    jugadorA.sety(y)

def jugadorA_down():
    y = jugadorA.ycor()
    y -= 20
    jugadorA.sety(y)

#Para el jugado B
def jugadorB_up():
    y = jugadorB.ycor()
    y += 20
    jugadorB.sety(y)

def jugadorB_down():
    y = jugadorB.ycor()
    y -= 20
    jugadorB.sety(y)
```

Figura 8. Control de los jugadores

Tenemos aquí cuatro funciones con las que controlaremos el movimiento hacia arriba y hacia abajo de cada una de las palas. Puesto que el movimiento de las mismas es exclusivamente vertical, vamos a usar la variable ≪y≫ que será igual a coordenada ≪y≫ en la que, en cada momento, se encuentre nuestra pala (dicho punto se obtendrá con ≪jugadorB.ycor()≪). De modo que cada vez que se ejecute alguna de estas funciones el valor de ≪y≫ se modificará en 20 unidades más (≪y += 20≪) cuandola función sea para mover la pala hacia abajo (funciones ≪jugadorA_down()≫ y ≪jugadorB_down()≪), y en 20 unidades menos (≪y -= 20≪) cuando el movimiento sea hacia arriba (≪jugadorA_up()≫ y ≪jugadorB_up()≪). Después de variar el valor de ≪y≫ se actualizará la posición del jugador con el método ≪sety()≪.

Como ya hemos dicho, estas son las funciones que gobernarán el comportamiento de los elementos gráficos de nuestra área de juego. No obstante hay que recordar que la ejecución de algunas de ellas (concretamente lascuatro que mueven los jugadores hacia arriba y haciaabajo responde a determinados eventos introducidos mediante el teclado. Por ello usaremos un método (≪.listen()≪) que pondrá al programa en ≪escucha≫ de las teclas que pulsemos del teclado:

```
#Conectar las funciones al teclado
#Para el jugador A
wn.listen()
wn.onkeypress(jugadorA_up, "w")

wn.listen()
wn.onkeypress(jugadorA_down, "s")

#Para el jugador B
wn.listen()
wn.onkeypress(jugadorB_up, "i")

wn.listen()
wn.onkeypress(jugadorB_down, "k")
```

Figura 9. Registrar eventos del teclado

Una vez iniciada la escucha el teclado, usaremos el método ≪onkeypress()≫ pasándole el nombre de la función que se ha de ejecutar y la tecla o evento que la ejecutará. De modo que tal y como lo hemos configurado, las teclas ≪w≫ y ≪s≫ moverán al jugador A (la de la izquierda) hacia arriba y abajo respectivamente. Por otra parte las teclas ≪i≫ y ≪k≫ lo mismo respecto al jugador B (la de la derecha).

Y finalmente, ejecutaremos un ≪while≫ que contendrá el desarrollo mismo del juego en el que definiremos el comportamiento y movimiento de la pelota en función de su posición respecto a los jugadores (de modo que por ejemplo, si entra en contacto con los jugadores, inicie un movimiento de retroceso):

```
#Creamos nuestro bucle principal para actualizar la ventana
while True:
    wn.update()

    pelota.setx(pelota.xcor() + pelota.dx)
    pelota.sety(pelota.ycor() + pelota.dy)


#Creamos los bordes de la pelota, es decir, que rebote cua
ndo toque un borde
    if pelota.ycor() > 290:
        pelota.dy *= -1

    if pelota.ycor() < -290:
        pelota.dy *= -1

    if pelota.xcor() > 390:
        pelota.goto(0,0)
        pelota.dx *= -1
        marcadorA += 1
        pen.clear()
        pen.write("Jugador A: {}         Jugador B: {}",
format(marcadorA,marcadorB), align="center", font=(
"Courier", 12, "normal"))

    if pelota.xcor() < -390:
        pelota.goto(0,0)
        pelota.dx *= -1
        marcadorB += 1
        pen.clear()
        pen.write("Jugador A: {}         Jugador B: {}",
format(marcadorA,marcadorB), align="center", font=(
"Courier", 12, "normal"))
```

Figura 10. Función while con el desarrollo del mismo juego

Como se ve en la imagen, durante el desarrollo de la partida, la pelota se moverá por el área de juego mediante la función ≪setx()≫ (para el eje horizontal) y ≪sety()≫ que tendrán como argumento el resultado de sumar al valor actual de posición el valor de ≪pelota.dx≫ y ≪pelota.dy≫ (que recordaremos que era de 0.17). En el transcurso de dicho desplazamiento por pantalla pueden darse dos circunstancias distintas: Que la pelota entre en contacto con alguna de los jugadores (en cuyocaso simularemos el retroceso de la misma tanto en su dirección como en su sentido. Algo parecido sucederá si la pelota entra en contacto con alguno de los bordes superior e inferior del área de juego. La otra eventualidad es que la pelota rebase a alguna de los jugadpres (lo que supone un punto más para el contrario). En ese caso se incrementará e 1, el valor de la variable ≪marcadorA≫ o ≪marcadorB≫ y se actualizará el texto que muestra en pantalla dicho resultado.

Esro se hace gracias al comadno pen.write, en la cual definimos un texto en el cual incluimos la variable masrcadorA y marcadorB, la cual indicamos al principio su valor es igual a cero, sin embargo, también definimos que, cada vez que la pelota toca, ya sea el borde izquierdo o derecho, el contador aumente uno, respectivamente, dependiento del borde que ha tocado.

La figura 11 indica quem cuadno la pelota toca a cualquier jugaro, entonces vaya al lado contrario.

```
#Creamos las colisiones de la pelota
    if ((pelota.xcor() > 340 and pelota.xcor() < 350
)
        and (pelota.ycor() < jugadorB.ycor() + 50
        and pelota.ycor() > jugadorB.ycor() -50)):
        pelota.dx *= -1

    if ((pelota.xcor() < -340 and pelota.xcor() > -
350)
        and (pelota.ycor() < jugadorA.ycor() + 50
        and pelota.ycor() > jugadorA.ycor() -50)):
        pelota.dx *= -1
```

Figura 11. Condicional if que indica a la pelota como moverse cuando colisiona con los jugadores

Finalmente, el juego se ve así:

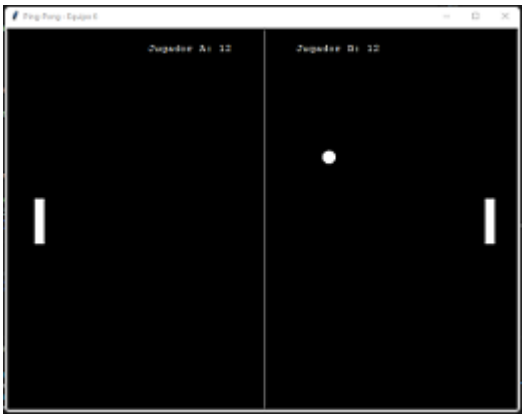

Figura 12. Juego Final

III. Resultados y análisis de resultados

El resultado fue el esperado, fuimos capaces de replicar el juego Pong con gran exactitud, a excepción del sonido que no pudimos adquirirlo (o por lo menos de forma legal); sin embargo el juego fuera de ese detalle o de jugar contra la maquina que tiene diferentes dificultades (que por cirto no estaba dentro de los objetivos ya que una de las finalidades era el poder divertirse y pasar el rato jugando ya sea solo o con amigos) es una replcia de muy buena calidad, cuenta con un marcador que mantiene el puntaje de cada jugadorademas de que se actualiza cada vez que se anota un nuevo punto que sucede cuando la pelota alcanza o supera el limite determinado de pixeles y la secuencia se reinicia mandando a la pelota de vuelta al centro para la siguiente ronda, las paletas tambien están condicionadas a un area predeterminada para evitar accidentes que puedan suceder y afectar la jugabilidad, de la misma manera es sencillo reprogramarlo para añadir dificultad o restarsela, ya sea reduciendo o aumentando el tamaño de las paletas, o cambiando la velocidad de la pelota, o incluso reduciendo las dimensiones del cuadro donde se desarrolla el juego lo cual está indicado en el mismo programa y basta con cambiar unos cuantos valores para poder personalizar lo más posible el juego a las preferencias del usuario, siendo el programa una gran alternativa cuando no te puedes conectar a

la red o simplemente no quieres instalar un juego o emulador que abarquen mucho espacio en la memoria de tu ordenador.

IV. Conclusiones

A lo largo de este proyecto pudimos observar que aunque la sintaxis del programa es sencilla, hay muchisimas cosas que aprender para poder llegar a ser un buen programador de python. A pesar de que tuvimos algunos tropiezos al desarrollar este proyecto pudimos solucionarlos y aprender de ellos trabajando en equipo, logrando concliur satisfactoriamente este tranajo, de el proceso nos llevamos más conocimiento a cerca de las bibliotecas que se pueden usar, el como funcionan y para que sirven, aunque reiteramos falta mucho por aprender de este lenguaje, pero ya estamos mas cerca que cuando empezamos.

Referencias

[1] Mundo Python.(2019). *Como programar pong en python para principianetes(y en 120 lineas de código)*. recuperado el 4 de diciembre de 2022 de: https://www.youtube.com/watch?v=o9ddr3J7FY8

[2] Sintes,M.(s/f). *Gráficos: el módulo turtle (1)*. Recuperado el 4 de diciembre de 2022 de https://www.mclibre.org/consultar/python/lecciones/python-turtle-1.html