



**Universidad Nacional Autónoma
de México**



Facultad de ingeniería

Snake y Breakout en Python

Sánchez García Rocío

Semestre 2021-2

Ing. Marco Antonio Martínez Quintana

ESTRUCTURA DE DATOS Y ALGORITMOS I

Fecha de elaboración: agosto 2021

Resumen

El presente documento tiene como propósito mostrar el desarrollo del proyecto que fue llevado a cabo durante el transcurso del semestre 2021-2 en la asignatura de Estructura de Datos y Algoritmos I. Puesto que el proyecto podía desarrollarse de manera libre en base a intereses e inquietudes propias, se tomó la decisión de replicar unos juegos sencillos, basándose en un tutorial, en lenguaje de programación Python con la finalidad de aplicar todos los conceptos aprendidos como lo son las estructuras de selección, estructuras de repetición, funciones, etc.

Antes de comenzar a programar fue necesario desarrollar un algoritmo donde se planteó la problemática, después se tuvieron en consideración las restricciones que giran en torno al proyecto, como la disponibilidad de tiempo, los datos de entrada y salida para posteriormente describir los pasos a seguir de manera cronológica para obtener el resultado deseado. Conforme se iba replicando el programa en entorno de desarrollo requerido se comentó el código fuente de manera continua para poder entender cuales eran los propósitos de cada línea de código.

Las capturas de pantalla tienen como finalidad evidenciar la funcionalidad del proyecto, la tabla de recursos informáticos muestra los recursos que fueron necesarios instalar para tener un entorno de desarrollo ideal, mientras que en la tabla de costos se evaluó propuso un costo basándose en la complejidad y el tiempo invertido en el mismo, ejercicio que resulto ser relativamente complicado por tratarse de una actividad enfocada principalmente en análisis y reconocimiento de comandos y funciones.

Antes de llevarlo a cabo fue necesario llevar una metodología para administrar las horas de trabajo que se le dedicarían a cada etapa. Una herramienta para la planificación fue el diagrama de Gantt donde se programaron las tareas y se estimó el tiempo en el que se culminaría el proyecto.

Finalmente se expresan las conclusiones a las que se llegó durante todo el proceso en base a lo que se esperaba en un principio y cómo fue que la perspectiva se fue modificando cada vez que se presentaba alguna dificultad.

Los videojuegos

Los videojuegos son muy populares entre las personas porque una de sus principales funciones consiste en entretener al usuario. Con el desarrollo de la tecnología hoy existe una cantidad considerable de videojuegos de diferentes géneros como: carreras, simulación, acción, estrategia, aventura, rol, etc. Y estos podemos encontrarlos en diferentes plataformas, desde dispositivos portátiles hasta consolas.

La creación de los videojuegos es una actividad compleja donde, las personas que llevan a cabo esta actividad deben diseñarlo y programarlo hasta lograr su versión final para posteriormente comercializarlos. El equipo de desarrollo involucra a profesionales de diferentes disciplinas para que puedan desempeñar tareas enfocadas en la informática, sonido, actuación y diseño. Lo que permite crear situaciones virtuales en las que el jugador podrá controlar a un jugador y seguir la trama o lógica del videojuego.

Los primeros videojuegos eran juegos muy simples y en un principio solo funcionaban equipos que poseían los institutos de investigación y algunas universidades. Un ejemplo de juegos clásicos es:

- Tetris
- Golden Axe
- Super Mario
- Pang
- Final Fight

La popularidad de los videojuegos gira en torno a varios factores, como la calidad de sus gráficos, el género, trama o disponibilidad en plataformas. Los videojuegos traen consigo algunos beneficios, pero también existen personas que argumentan que estos pueden generar efectos psicológicos negativos que comprometen la seguridad e integridad del individuo.

Desarrollo

La idea principal fue replicar dos tutoriales de YouTube y los cuales consistían en realizar juegos clásicos como “Snake” y “Breakout” en lenguaje de programación Python, esto con la finalidad de analizar y poder describir la función de cada una de las líneas de código descritas dentro del programa.

Algoritmo

Programa compilador de videojuegos

PROBLEMA: Realizar un programa donde se compilen dos juegos hechos a partir de tutoriales en YouTube.

RESTRICCIONES: Debido a la disponibilidad de tiempo se replicarán los juegos exactamente como en los tutoriales, pero se deberá de identificar el propósito principal de las líneas de código del programa mostrado.

DATOS DE ENTRADA: Librerías (pygame, random), listas, funciones, estructuras de repetición, estructuras de selección.

DATOS DE SALIDA: Un programa que nos permita elegir entre jugar un juego u otro, y si el usuario así lo desea podrá elegir la opción salir.

SOLUCIÓN:

1. Crear un menú
2. Mostrar un mensaje de bienvenida
3. Mostrar las opciones a elegir
4. Solicitar la opción al usuario
 - 4.1 En caso de elegir el juego de “snake”
 - 4.11 Importar las librerías de pygame y random
 - 4.12 Crear una clase que corresponda al cuerpo de la serpiente.
 - 4.121 Establecer la posición inicial de la serpiente
 - 4.122 Establecer la dirección en la que se comenzara a mover el cuerpo
 - 4.13 Crear una función para dibujar el cuerpo de la serpiente
 - 4.14 Crear una función para que el cuerpo se mueva.
 - 4.15 Crear una clase que corresponda a la comida de la serpiente

- 4.151 Establecer que la posición de la comida aparezca de manera aleatoria
- 4.152 Crear una función que corresponda al cuerpo de la serpiente
 - 4.1521 Determinar el color de la serpiente
- 4.16 Crear una función para dibujar de manera constante en la ventana.
 - 4.161 Llenar la ventana
 - 4.162 Dibujar la comida
 - 4.163 Usar un ciclo de repetición ya que la serpiente va a cambiar su tamaño durante el juego
 - 4.1631 Dibujar a la serpiente
- 4.17 Crear una función para que el cuerpo siga a la cabeza
- 4.18 Crear una función principal
 - 4.181 Crear una variable global
 - 4.182 Mostrar la ventana de juego
 - 4.183 Representar el cuerpo de la serpiente como una lista
 - 4.184 Crear una variable que indique el movimiento de la serpiente.
- 4.19 Usar la estructura de repetición while
 - 4.191 Usar la estructura de repetición for
 - 4.1911 Cambiar la dirección de los teclados
 - 4.192 Cada vez que el código corra la serpiente se debe mover
 - 4.193 Llamar a la función refrescar
 - 4.194 Refrescar la ventana completamente
 - 4.195 Establecer la velocidad que tarda en girar
 - 4.196 Cada vez que coma la serpiente debe crecer
 - 4.5561 Añadir una función a la serpiente
 - 4.197 El cuerpo debe de seguir a la serpiente
 - 4.198 Si la cabeza de la serpiente se sale de la pantalla se moverá la cabeza a la posición inicial de la pantalla
 - 4.199 Salida del juego
- 4.2 En caso de elegir el juego de "Breakout"
 - 4.21 Importar la librería de pygame
 - 4.211 Crear una función que defina la posición de la pelota
 - 4.212 Crear una función para dibujar a la pelota
 - 4.2121 Definir el color y el tamaño de la pelota
 - 4.213 Crear una función para mover a la pelota
 - 4.2131 Los movimientos se realizan de acuerdo a la posición
 - 4.22 Definir la plataforma en la que rebotara la pelota

- 4.221 Definir el tamaño de la plataforma
- 4.222 Definir la posición de la plataforma
- 4.223 Crear variables para que se mueva la plataforma
- 4.23 Crear una función para dibujar la plataforma en la ventana
- 4.24 Crear una función para definir el movimiento de la plataforma.
 - 4.241 poner limites para que la plataforma no se salga de la pantalla
- 4.25 Creación de una clase para que los bloques superiores
 - 4.251 creación de los bloques a destruir
- 4.26 Crear una función para dibujar el tablero en la pantalla
 - 4.261 Darle un color al tablero
- 4.27 crear una función para refrescar la ventana
 - 2.271 Llenar la ventana
 - 2.272 Dibujar la pelota en la ventana
 - 2.273 dibujar la plataforma en la ventana
 - 2.274 mostrar el tablero
 - 2.275 mostrar texto y centrarlo
- 4.28 crear una función que se encargara de evaluar los golpes en el tablero
- 4.29 Crear una función principal en la que se encuentren todas las funciones necesarias para que el programa se ejecute de manera correcta.
- 4.3 En caso de elegir la opción salir
 - 4.31 Mostrar un mensaje de despedida del programa
- 4.4 En caso de que el usuario ingrese una opción que no se encuentre en el menú
 - 4.41 Mostrar un mensaje que indique que la opción no es valida

Código fuente

```
import os
#Cración de un menú
op='1'
while(op!='3'):
    os.system("cls")
    # Mensaje de bienvenida
    print("\n\n\t\tBienvenid@")
    print("\n\t1) Jugar Snake\n \t2) Jugar Breakout\n \t3) Salir\n")
```

```

op=input("\n\tElige una opción: ")
if op == '1':
    import pygame
    #Se usa para la comida
    import random

    #Clase que corresponde al cuerpo de la serpiente
    class cuerpo:
        def __init__(self, ventana):
            #Posición inicial de la serpiente
            self.x = 0
            self.y = 0
            #Dirección que dirá hacia donde se mueve el cuerpo
            self.dir = 0
            self.ventana = ventana

        #Función para dibujar el cuerpo de la serpiente
        def dibujar(self):
            pygame.draw.rect(self.ventana, (255, 255, 255),
            (self.x, self.y, 10, 10))

        #Función para que el cuerpo se mueva
        def moverse(self):
            if self.dir == 0:
                self.x += 10
            elif self.dir == 1:
                self.x -= 10
            elif self.dir == 2:
                self.y += 10
            elif self.dir == 3:
                self.y -= 10

    #Clase que corresponde a la comida de la serpiente
    class manzanas:
        def __init__(self, ventana):
            #La posición de la comida es aleatoria
            self.x = random.randrange(40) * 10
            self.y = random.randrange(40) * 10
            self.ventana = ventana

        #Función que corresponde al cuerpo de la serpiente
        def dibujar(self):
            #Color de la serpiente (255, 0, 0)
            pygame.draw.rect(self.ventana, (255, 0, 0), (self.x,
            self.y, 10, 10))

        def nueva_manzana(self):
            self.x = random.randrange(40) * 10
            self.y = random.randrange(40) * 10

```

```

#Función para dibujar constantemente en la ventana
def refrescar(ventana):
    #Llenado de color negro
    ventana.fill((0, 0, 0))
    #Dibujar la comida
    comida.dibujar()
    #Como la serpiente va a tener varias partes se usa el
ciclo for
    for i in range(len(serpiente)):
        #Dibujar a la serpiente
        serpiente[i].dibujar()

#Función que va a seguir a la cabeza
def seguir_cabeza():
    #A todas las posiciones
    for i in range(len(serpiente) - 1):
        '''
        La última posición de la serpiente va a ser igual a
la penúltima
        posición y así sucesivamente hasta que la segunda
posición sea igual
        a la cabeza, tanto en x como en y
        '''
        serpiente[len(serpiente) - i - 1].x =
serpiente[len(serpiente) - i - 2].x
        serpiente[len(serpiente) - i - 1].y =
serpiente[len(serpiente) - i - 2].y

#Función principal
def main():
    #Variable global
    global serpiente, comida
    #Para ventana de Juego
    ventana = pygame.display.set_mode((400, 400))
    ventana.fill((0, 0, 0))
    comida = manzanas(ventana)
    #Serpiente va a ser una lista
    serpiente = [cuerpo(ventana)]

    #Variable
    run = True
    #Estructura de repetición while
    while run:
        #Estructura de repetición for
        for event in pygame.event.get():
            #Si se sale del programa entonces la variable
run sera falsa
            if event.type == pygame.QUIT:

```



```

        run = False
        #Cambiar la dirección con los teclados
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                #Cambiar la dirección de la cabeza
                serpiente[0].dir = 0
            if event.key == pygame.K_LEFT:
                #Cambiar la dirección de la cabeza
                serpiente[0].dir = 1
            if event.key == pygame.K_DOWN:
                #Cambiar la dirección de la cabeza
                serpiente[0].dir = 2
            if event.key == pygame.K_UP:
                # Cambiar la dirección de la cabeza
                serpiente[0].dir = 3

#Cada vez que el código corra la serpiente se debe
mover
    serpiente[0].moverse()
    #Llamar a la función refrescar
    refrescar(ventana)
    #Refrescar la ventana completamente
    pygame.display.update()
    #Establece la velocidad que tarda en girar
    pygame.time.delay(96)
    #Cada vez que coma la serpiente debe crecer
    if serpiente[0].x == comida.x and serpiente[0].y ==
comida.y:
        comida.nueva_manzana()
        #A la serpiente le vamos a añadir una nueva
función
        serpiente.append(cuerpo(ventana))
        #Para que el cuerpo siga a la cabeza de la serpiente
        seguir_cabeza()
        #Si la cabeza de la serpiente se sale de la pantalla
se movera la cabeza a la posición 0
        if serpiente[0].x >= 400:
            serpiente[0].x = 0
        #Si la cabeza esta en una posición menor a cero
tiene que ir a la posición donde inicia la pantalla
        elif serpiente[0].x < 0:
            serpiente[0].x = 390
        if serpiente[0].y >= 400:
            serpiente[0].y = 0
        elif serpiente[0].y < 0:
            serpiente[0].y = 390

if __name__ == '__main__':
    main()

```

[illegible]

```

        #Función para definir el movimiento de la plataforma
    def mover(self):
        if self.izq: self.x -= 10
        if self.der: self.x += 10
        #Poner limites para que la plataforma no se salga de
la pantalla
        self.x = 0 if self.x < 0 else 600 - self.tamano if
self.x + self.tamano > 600 else self.x
        self.centro = self.x + self.tamano / 2

#Creación de una clase para los bloques superiores
class Bloques:
    def __init__(self, ventana):
        self.ventana = ventana
        #Creación de los bloques a destruir
        self.tablero = [[4, 4, 4, 4, 4, 4, 4, 4, 4, 4],
                        [3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
                        [2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
                        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]

#Función para dibujar el tablero en la pantalla
def dibujar(self):
    for i in range(4):
        for j in range(10):
            if self.tablero[i][j] != 0:
                #Darle un color al tablero
                if self.tablero[i][j] == 4:
                    color = (255, 255, 255)
                elif self.tablero[i][j] == 3:
                    color = (55, 255, 255)
                elif self.tablero[i][j] == 2:
                    color = (55, 55, 255)
                elif self.tablero[i][j] == 1:
                    color = (255, 55, 255)
                pygame.draw.rect(self.ventana, color, (j
* 60, i * 10 + 35, 59, 9))

#Función para refrescar la ventana
def refrescar(ventana):
    #Llenado de la ventana
    ventana.fill((0, 0, 0))
    #Dibujar la pelota en la ventana
    bola.dibujar()
    #Dibujar la plataforma en la ventana
    r1.dibujar()
    #Mostrar el tablero
    tablero.dibujar()
    #Dibujar el texto
    text = font.render(str(golpes), True, ((255, 255, 255)))

```

```

        text_rect = text.get_rect()
        #Centrar el texto
        text_rect.centerx = 300
        ventana.blit(text, text_rect)

#Función que se encargará de evaluar los golpes con el
tablero
def colisiones():
    global golpes
    if bola.y < 3 * 10 + 35 + 9:
        for i in range(4):
            for j in range(10):
                if tablero.tablero[i][j] != 0:
                    if ((j * 60 < bola.x < j * 60 + 59) or
(j * 60 < bola.x + 10 < j * 60 + 59)) and (
                    (i * 10 + 35 < bola.y < i * 10 +
35 + 9) or (
                    i * 10 + 35 < bola.y + 10 < i *
10 + 35 + 9)):
                        #Borrar un bloque cuando sea
golpeado
                        tablero.tablero[i][j] = 0
                        #La pelota rebotará al golpear un
bloque
                        bola.vy *= -1
                        #Los golpes solo se contarán cuando
la pelota rompa el tablero
                        golpes += 1

#Función principal
def main():
    #Generar una variable global
    global bola, golpes, font, r1, tablero
    #Creación de ventana
    ventana = pygame.display.set_mode((600, 400))
    #Rellenado de la ventana
    ventana.fill((0, 0, 0))
    bola = pelota(ventana, 50, 100)
    #Asignarle velocidad a la pelota
    bola.vx = 5
    bola.vy = 2
    golpes = 0
    #Aparición de texto en la pantalla
    pygame.font.init()
    font = pygame.font.SysFont("Arial", 30)
    #Variable que va a permitir hacer el ciclo while
    jugar = True
    #Variable para la plataforma
    r1 = Raqueta(ventana)

```

```

        tablero = Bloques(ventana)
        #Variable reloj
        clock = pygame.time.Clock()
        #Ciclo while
        while jugar:
            #Eventos que se generán
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    jugar = False
            #Movimiento de la plataforma de acuerdo a la
tecla oprimida
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_LEFT:
                        r1.izq = True
                    if event.key == pygame.K_RIGHT:
                        r1.der = True
                if event.type == pygame.KEYUP:
                    if event.key == pygame.K_LEFT:
                        r1.izq = False
                    if event.key == pygame.K_RIGHT:
                        r1.der = False
            #Para mover a la pelota
            bola.mover()
            r1.mover()
            #Llamar a la función colisiones
            colisiones()
            #Definir que la pelota se golpee contra los muros
            if bola.x >= 590:
                bola.vx *= -1
                bola.x = 590
            if bola.x <= 0:
                bola.vx *= -1
                bola.x = 0
            #Hacer que la pelota rebote en la plataforma
            if bola.y + 10 > r1.y:
                if (r1.x < bola.x < r1.x + r1.tamano) or (r1.x <
bola.x + 10 < r1.x + r1.tamano):
                    #Variable que va a definir el porcentaje de
las velocidades dependiendo del lugar de impacto de la pelota
                    porcentaje = (bola.x - r1.centro) /
(r1.tamano / 2)
                    bola.vx += porcentaje * 10
                    bola.vx = -10 if bola.vx < -10 else 10 if
bola.vx > 10 else bola.vx
                    bola.vy *= -1
                    bola.y = r1.y - 10
                elif bola.y > 400:
                    bola.y = 100
            if bola.y <= 0:

```

```

        bola.vy *= -1
        bola.y = 0
        #Llamar a la función refrescar
        refrescar(ventana)
        clock.tick(60)
        pygame.display.update()

    if __name__ == '__main__':
        main()
        pygame.quit()
    elif op == '3':
        #Mensaje en caso de elegir la opción salir
        print("\n\tElegiste salir, Gracias por usar mi programa")
        input("\n\tPresiona enter para salir...")
    else:
        #Mensaje en caso de elegir una opción que no venga en el
menú
        print("\n\t***OPCIÓN NO VALIDA***")
        input("\n\tPresiona enter para continuar...")

```

Análisis de las líneas de código

```

11     import pygame
12     #Se usa para la comida
13     import random
14
15     #Clase que corresponde al cuerpo de la serpiente
16     class cuerpo:
17     def __init__(self, ventana):
18         #Posición inicial de la serpiente
19         self.x = 0
20         self.y = 0
21         #Direccion que dirá hacia donde se mueve el cuerpo
22         self.dir = 0
23         self.ventana = ventana
24
25     #Función para dibujar el cuerpo de la serpiente

```

Librería necesaria para que la comida apareciera de manera aleatoria

```

40     #Clase que corresponde a la comida de la serpiente
41     class manzanas:
42     def __init__(self, ventana):
43         #La posición de la comida es aleatoria
44         self.x = random.randrange(40) * 10
45         self.y = random.randrange(40) * 10
46         self.ventana = ventana
47

```

Función encarga de generar la comida de manera aleatoria en la ventana

```

68     #Función que va a seguir a la cabeza
69     def seguir_cabeza():
70         #A todas las posiciones
71         for i in range(len(serpiente) - 1):
72             '''
73             La última posición de la serpiente va a ser igual a la penúltima
74             posición y así sucesivamente hasta que la segunda posición sea igual
75             a la cabeza, tanto en x como en y
76             '''
77             serpiente[len(serpiente) - i - 1].x = serpiente[len(serpiente) - i - 2].x
78             serpiente[len(serpiente) - i - 1].y = serpiente[len(serpiente) - i - 2].y

```

Esta función hizo que el cuerpo se moviera junto con la cabeza de la serpiente

```

171     #Definir la plataforma en la que rebotará
172     class Raqueta:
173         def __init__(self, ventana):
174             #Definir el tamaño de la plataforma
175             self.tamano = 80
176             #Definir la posición de la plataforma
177             self.x = 600 / 2 - self.tamano / 2
178             self.y = 380
179             self.centro = self.x + self.tamano / 2
180             self.ventana = ventana
181             #Variables para que se mueva la plataforma
182             self.izq = False
183             self.der = False
184
185     #Función para dibujar la plataforma en la ventana
186     def dibujar(self):
187         pygame.draw.rect(self.ventana, (255, 255, 255), (self.x, self.y, self.tamano, 10))
188
189     #Función para definir el movimiento de la plataforma
190     def mover(self):
191         if self.izq: self.x -= 10
192         if self.der: self.x += 10
193         #Poner límites para que la plataforma no se salga de la pantalla
194         self.x = 0 if self.x < 0 else 600 - self.tamano if self.x + self.tamano > 600 else self.x
195         self.centro = self.x + self.tamano / 2
196
197     #Creación de una clase para los bloques superiores
198     class Bloques:
199         def __init__(self, ventana):
200             self.ventana = ventana
201             #Creación de los bloques a destruir
202             self.tablero = [[4, 4, 4, 4, 4, 4, 4, 4, 4, 4],
203                             [3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
204                             [2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
205                             [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]
206
207     #Función para dibujar el tablero en la pantalla
208     def dibujar(self):
209         for i in range(4):
210             for j in range(10):
211                 if self.tablero[i][j] != 0:
212                     #Darle un color al tablero
213                     if self.tablero[i][j] == 4:
214                         color = (255, 255, 255)
215                     elif self.tablero[i][j] == 3:
216                         color = (55, 255, 255)
217                     elif self.tablero[i][j] == 2:

```

Uso de arreglos para la creación de los bloques superiores del tablero

```

266     #Asignarle velocidad a la pelota
267     bola.vx = 5
268     bola.vy = 2
269     golpes = 0
270     #Aparición de texto en la pantalla
271     pygame.font.init()

```

Con estas líneas de código, se podía cambiar la velocidad tanto en el eje x como en el eje y para el movimiento de la pelota

```

297     #Para mover a la pelota
298     bola.mover()
299     r1.mover()
300     #llamar a la función colisiones
301     colisiones()
302     #Definir que la pelota se golpee contra los muros
303     if bola.x >= 590:
304         bola.vx *= -1
305         bola.x = 590
306     if bola.x <= 0:
307         bola.vx *= -1
308         bola.x = 0

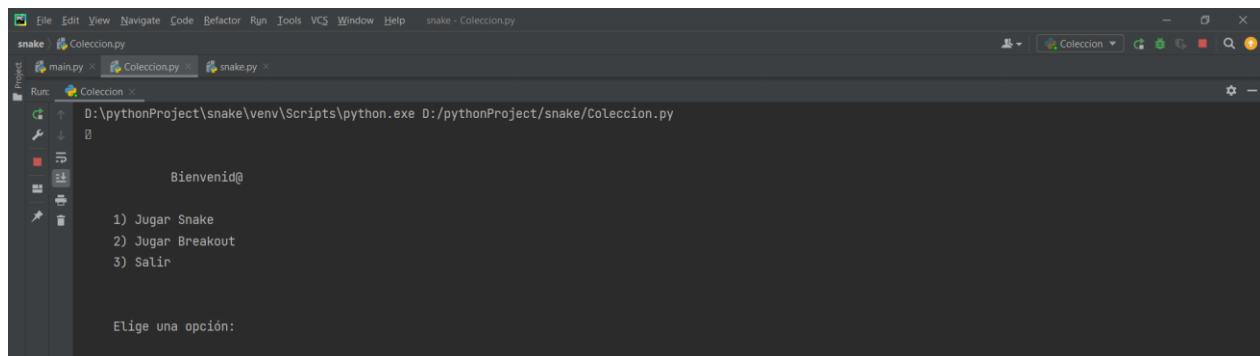
```

Sin estas estructuras la pelota simplemente desaparecía de la pantalla

Resultados

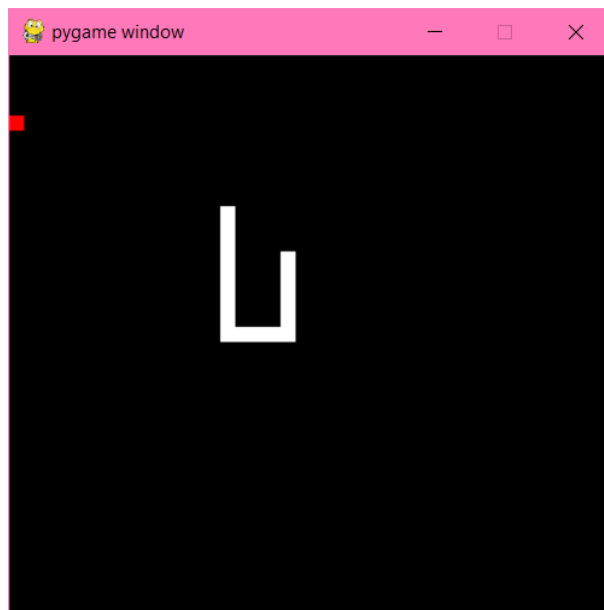
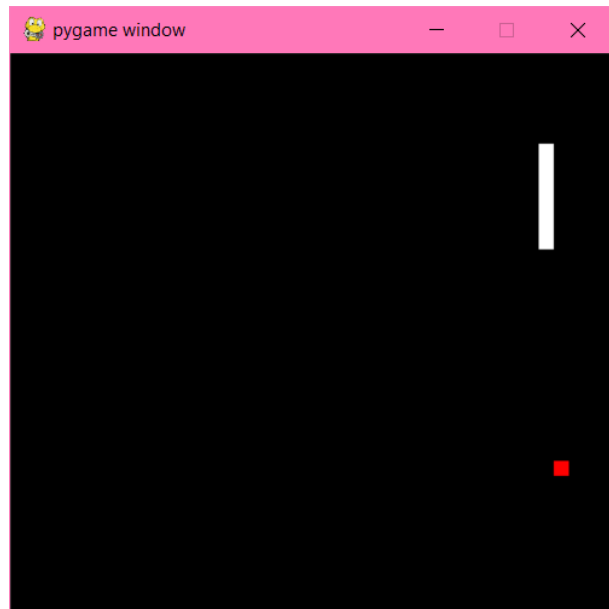
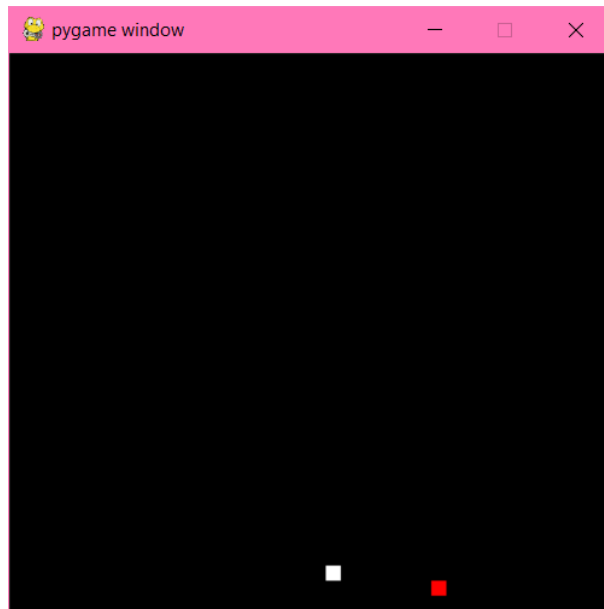
Capturas de pantalla del funcionamiento del proyecto

- *Menú que aparece una vez que se ejecuta el programa*

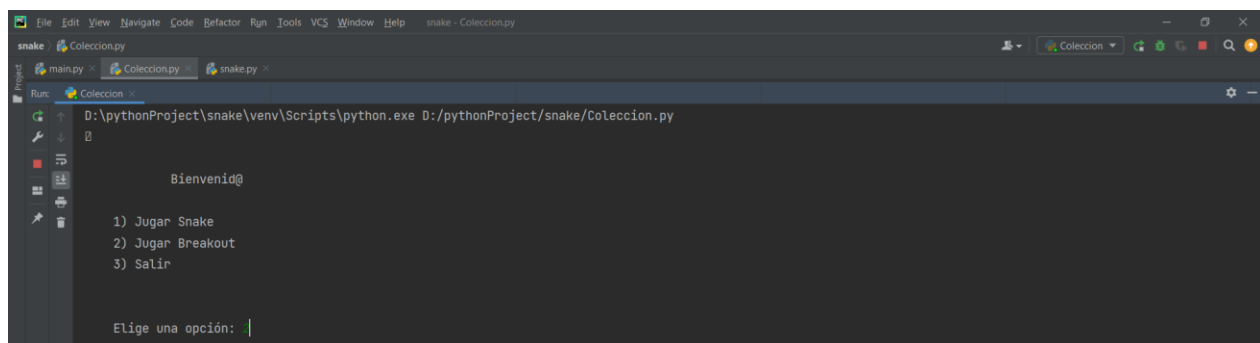


- *Ventana emergente al elegir la opción 1 del menú*

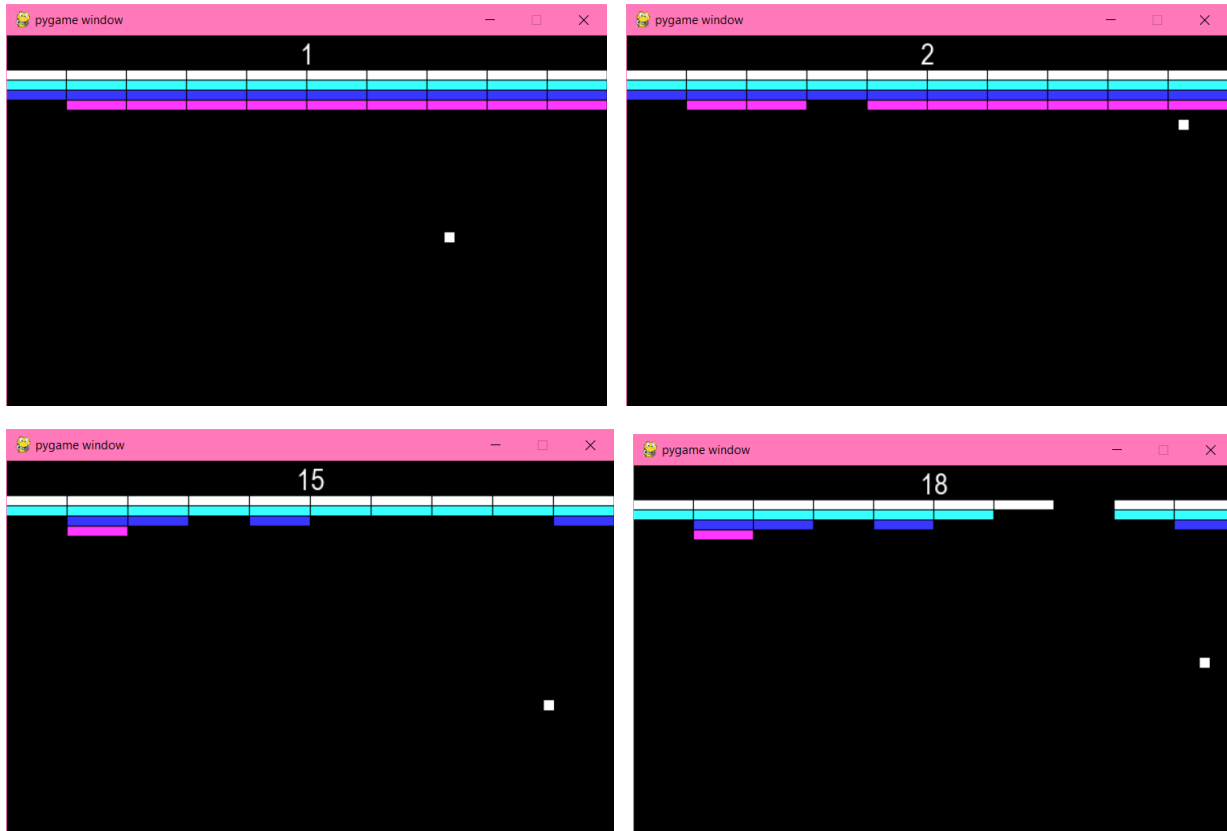
*Se puede el cambio de tamaño de la serpiente conforme se van comiendo los puntos rojos.



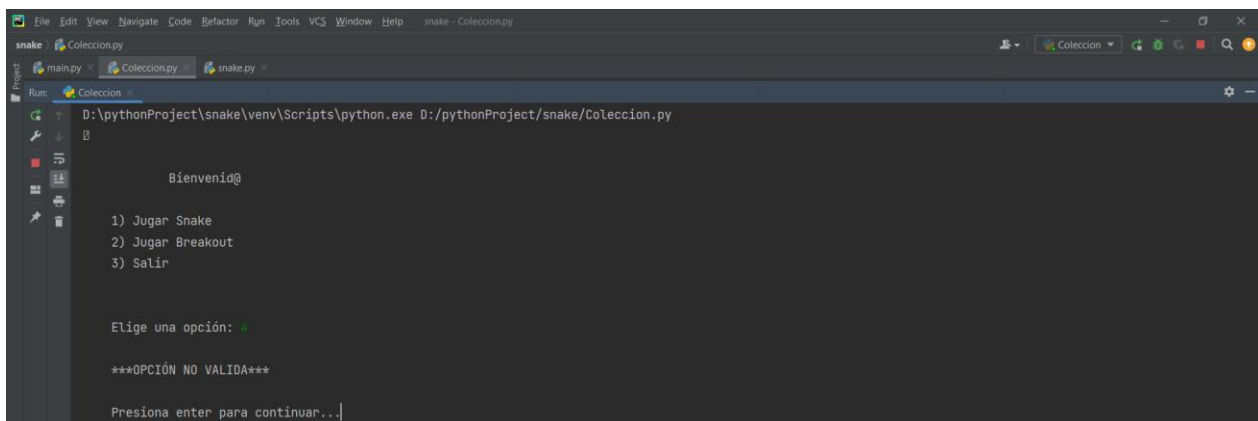
- *Al cerrar la ventana emergente aparece nuevamente al menú, en este caso se elige la opción 2.*



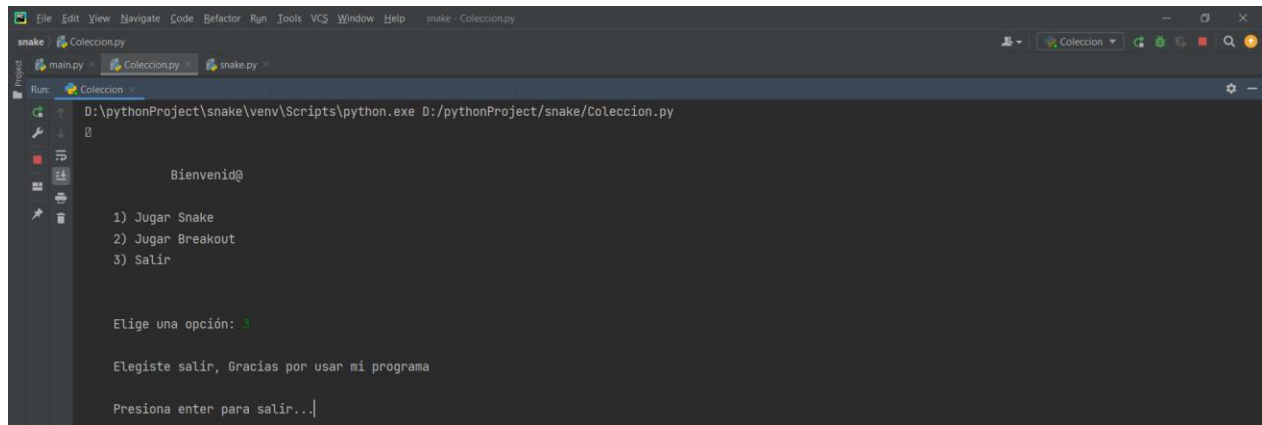
- *Emerge una nueva ventana donde se puede apreciar un tablero, una plataforma y una pelota, el objetivo es destruir los bloques superiores para sumar puntos.*



- *Mensaje que se muestra cuando se intenta ingresar una opción que no viene en el menú*



- *Mensaje mostrado cuando el usuario desea salir del programa*



```
snake - Coleccion.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project: snake
  main.py Coleccion.py snake.py
Run: Coleccion
D:\pythonProject\snake\venv\Scripts\python.exe D:/pythonProject/snake/Coleccion.py
Bienvenid@
1) Jugar Snake
2) Jugar Breakout
3) Salir
Elige una opción: 3
Elegiste salir, Gracias por usar mi programa
Presiona enter para salir...
```

Tabla de recursos informáticos necesarios para llevar a cabo el proyecto

Recursos informáticos
❖ Computadora
❖ Asesoría para configurar el equipo de computo
❖ Curso de lenguaje de programación Python
❖ Python 3
❖ IDE PyCharm

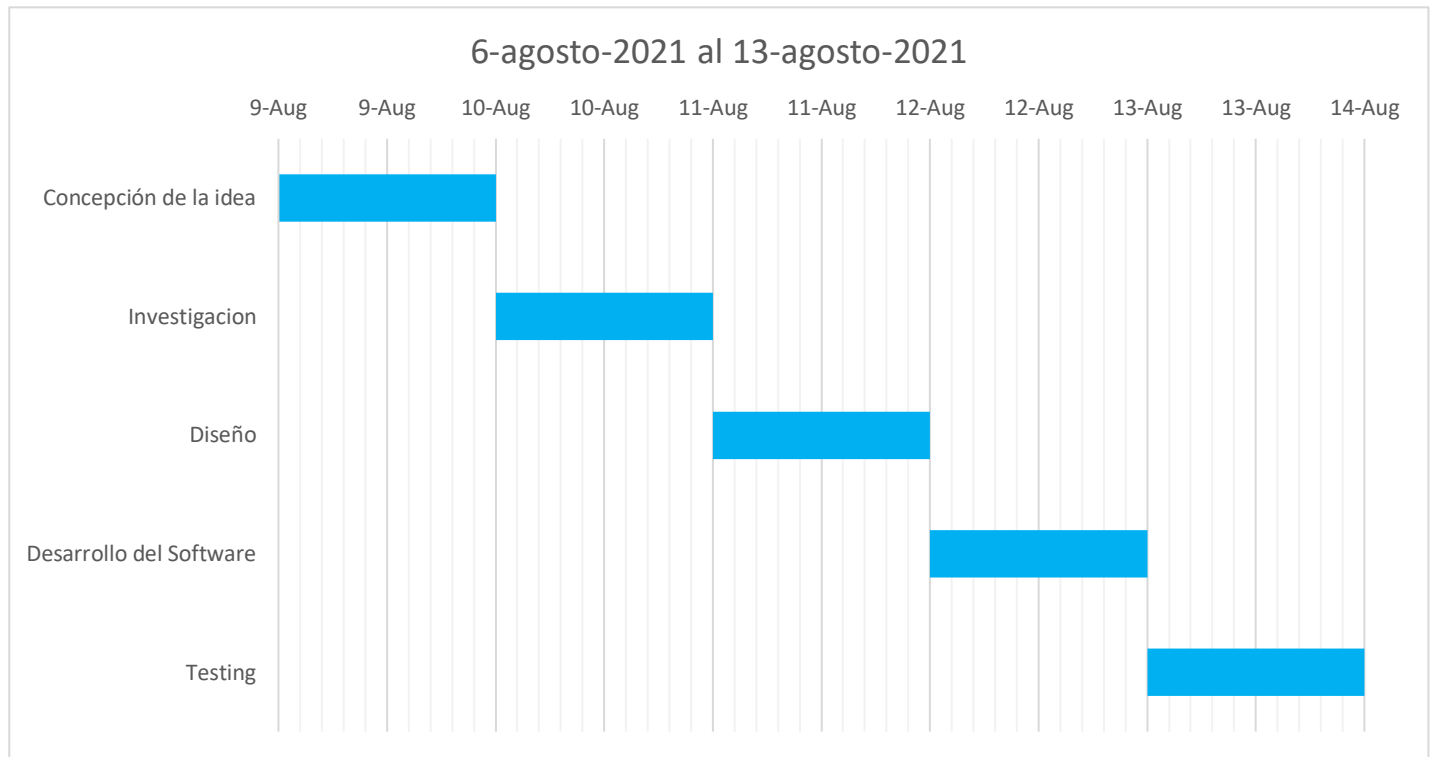
Tabla de costos propuestos para el desarrollo del proyecto

A partir de las siguientes preguntas:

¿Cuánto estarías dispuesto a pagar para que alguien desarrolle tu proyecto? Y ¿Cuánto cobrarías si tú lo hicieras?

Costo propuesto a pagar	Costo propuesto a cobrar
\$ 1 000.00 - \$ 1 500.00	\$ 1 000.00 - \$ 1 500.00

Diagrama de Gantt (6-agosto-2021 al 13-agosto-2021)



Canal de YouTube

<https://www.youtube.com/channel/UCtU2KFKvu6W5ovU6gzPloZw>

Repositorio de GitHub del Proyecto Final

<https://github.com/Rocio-Sanchez-Garcia/Estructura-de-Datos-y-algoritmos-/tree/main/Proyecto%20Final>

Conclusiones

Como se sabe, los algoritmos nos indican los pasos a seguir para poder resolver un problema. Si estos no se llevan a cabo al pie de la letra podrían provocar un desperfecto o resultados no deseados ya que estos se encuentran ordenados de manera cronológica, en este sentido los algoritmos son de vital importancia al programar ya que así se tienen en cuenta las características que debe poseer el programa.

Las estructuras de datos nos permiten manejar grandes cantidades de información, de tal modo que esto nos permite ahorrarnos algunas líneas de código, haciendo así que el código pueda ser interpretado por personas que sean ajenas a la realización del mismo

lo cual representa una ventaja al querer desempeñar un trabajo que gire en torno a la industria tecnológica, puesto que aquí el uso de grandes bases de datos es un trabajo que requiere o está en busca de personas capaces de abstraer ideas para implementar e innovar programas.

A pesar de las limitaciones a las que nos hemos enfrentado durante este semestre considero que el llevar a cabo este proyecto me ayudo bastante a entender algunos conceptos. Con la realización de este proyecto obtuve más conocimientos que en el del semestre pasado y aunque este trabajo en si no fue un proyecto propio el hecho de ir siguiendo paso a paso cada uno de los tutoriales de YouTube me permitió dominar las funciones debido que, anterior a esto, me era imposible usar funciones en los códigos de algunas de las actividades que desempeñé con anterioridad.

Referencias

Juego de Snake en Python | Tutorial. CarrasTech. 28 mar. 2020. Consultado el 9 de agosto de 2021. <https://www.youtube.com/watch?v=dFQjK0dI7CE>

Juego Breakout en Python | Tutorial en Python. CarrasTech. 19 abr. 2020. Consultado el 10 de agosto de 2021. <https://www.youtube.com/watch?v=HNVBL6CRnMw>