



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Proyecto Final

Snake

Alumno:

Eduardo Amilcar Flores Ramírez

Semestre 2021-2

Profesor:

M.I. Marco Antonio Martínez Quintana

Estructuras de Datos y Algoritmos 1



06/agosto/2021 al 13/agosto/2021

Resumen:

El proyecto, es una adaptación del juego Snake en Python.

El juego consta de un marcador que enseña el puntaje durante la partida y otro marcador que enseña la puntuación más alta comparando todas las partidas anteriormente jugadas. El juego es simple consta de una serpiente que al principio es una cabeza y que mientras mas comida recolecte, esta se hará más grande.

Al correr el programa se despliega una ventana titula Snake, en donde se puede ver un cuadrado de color verde, colocado justamente en el centro de la ventana, el cual representa la cabeza de la serpiente que está en reposo, como también hay un círculo café el cual representa la comida para dicha serpiente, y finalmente los dos marcadores uno que guarda el progreso durante la partida y otro que guarda el progreso al finalizar la partida.

Las maneras de terminar el juego son dos, la primera es colisionando con los limites del tablero, en este caso, con los limites de la ventana, y la segunda es chocando con tu propio cuerpo. Una vez terminado el juego, el programa no se detiene, sino que reinicia la partida y guarda tus puntos obtenidos solo si superaste el valor que esta guardado en el marcador con la puntuación mas alta.

El principio en el que se basa este Snake es el del plano cartesiano, ya que su tablero es un plano con dimensiones de seiscientos por seiscientos en donde se hace referencia a los planos "x" y "y" los cuales van desde menos trecientos hasta trecientos, la generación de la comida esta dada por un módulo "random" el cual no puede superar la cifra de trecientos, garantizando la aparición de la comida en todos los casos.

Finalmente, al terminar una partida, el programa no termina, sino que se reinicia y guarda un dato el cual es la puntuación mas alta. En caso de querer cerrar el juego la única manera es cerrando la ventana y forzando al programa a terminar.

Snake

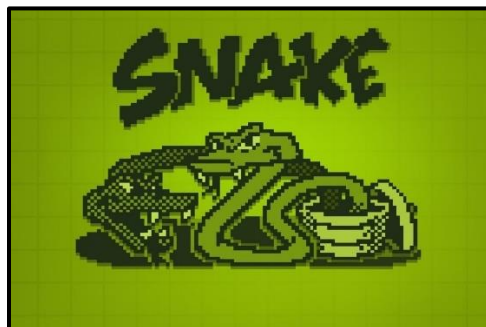
En este proyecto se desarrolla el juego Snake o la serpiente. Este juego clásico fue creado a mediados de los años 70, pero volvió a retomar mayor relevancia en 1998 debido a que era un juego contenido en los teléfonos móviles Nokia.

El origen del juego Snake fue el Blockade, un juego arcade desarrollado por Gremlin en 1976. Era un juego para dos jugadores que consistía en el manejo de una flecha mediante cuatro botones direccionales que va dejando una línea sólida a su paso. El objetivo era sobrevivir consiguiendo la estela más larga sin chocar con los bordes y las estelas del propio jugador o del jugador contrario. El juego terminaba cuando uno de los dos jugadores ganaba seis veces.

A diferencia del Blockade, el juego Snake es de un solo jugador que controla la dirección que toma una serpiente en constante movimiento y debe hacer que coma trozos de comida que van apareciendo en la pantalla que a su vez hace que crezca su longitud. A medida que va transcurriendo la partida y la serpiente va creciendo aumenta la complicación de la misma.

La primera versión conocida para el juego se tituló Worm y fue programada en 1978 por P. Trefonas para los ordenadores TRS-80 (Tandy Radio Shack 2-80), que eran un ordenador personal doméstico de la época que tenía un procesador con una velocidad de 1,77 MHz. Posteriormente hubo versiones también para Commodore y Apple II.

A lo largo del tiempo se han realizado múltiples versiones para ordenadores, como móviles, consolas u otras plataformas. Las últimas versiones que nos podemos encontrar hoy día son en tres dimensiones.



Descripción general:

Este proyecto es un juego muy simple pero también muy entretenido, el desarrollar este juego tiene muchas ventajas, entre las cuales destacan que es un clásico, este juego podría implementarse en los buscadores de Google como una opción para jugar bajo ciertas circunstancias.

Todos conocemos al dinosaurio de Google que salta cactus y esquivo pterodáctilos, el cual nos aparece cada vez que no tenemos internet y nos encontramos en una de las páginas de dicha marca, pero ahora imaginemos que en lugar de jugar con este dinosaurio jugamos con una serpiente que come manzanas y se desplaza a través del tablero, las personas no lo rechazarían dado que es un clásico y muy reconocido juego.

Otra opción puede ser como juego de teléfono, este juego no necesita de una presentación ya que, con decir el nombre de Snake, sabemos que nos referimos a un clásico entre clásicos el cual siempre nos tendrá entretenidos y contentos, aunque sea por un momento.

Algoritmo:

1. Crear una ventana gráfica:

- Nombrar a la ventana.
- Definir las dimensiones de la ventana.

2. Definir a la serpiente dentro de la ventana:

- Crear un elemento gráfico.
- Mover un elemento gráfico.

3. Definir la comida de la serpiente:

- Crear un elemento grafico que se mueva de manera aleatoria.

4. Definir el cuerpo de la serpiente:

- Crear un elemento grafico que siga a otros elementos gráficos, de forma que se cree una cola.

5. Colisiones:

- Definir el final del juego a partir de sobrepasar los límites de la ventana.
- Definir el final del juego a partir de chocar con su mismo cuerpo.

6. Marcador de puntos:

- Definir un marcador que cuente el progreso durante el juego.
- Definir un marcador que registré el progreso al finalizar el juego, y lo evalúe contra el más puntaje más alto.

Código:

```
import turtle          #Modulo turtle
import time            #Modulo time
import random          #Modulo random

posponer = 0.1         #Hace el movimiento de nuestro programa
                      #sea más lento.

#Marcador
score = 0              #Inicializamos el marcador en cero.
high_score = 0         #E inicializamos el marcador más alto también en cero.

#Ventana
wn = turtle.Screen()
                      #Con el modulo turtle, abrimos una ventana y la asignamos a wn(window).
```

```

wn.title("Snake")
    #Le ponemos nombre a la ventana que abrimos anteriormente
wn.bgcolor("black")          #Agregamos un fondo negro a la ventana
wn.setup(width = 600, height = 600)
    #La damos dimensiones a la ventana de 600*600. Con forma de plano
    #cartesiano.
wn.tracer(0)
    #Le pedimos a la consola que mejores las animaciones dentro de
    #nuestra ventana.

#Cabeza Serpiente
cabeza = turtle.Turtle()
    #Creamos la variable "cabeza" como un objeto turtle.
cabeza.speed(0)
    #Solicitamos que el objeto aparezca al mismo tiempo que la ventana.
cabeza.shape("square")
    #Le damos forma de cuadrado a nuestro objeto cabeza.
cabeza.color("green")        #Le otorgamos el color verde al objeto cabeza.
cabeza.penup()
    #Limpiamos el rastro que el modulo turtle deja cuando mueve un objeto.
cabeza.goto(0,0)
    #Ubicamos al objeto cabeza en el centro de la ventana.
cabeza.direction = "stop"
    #Hacemos que la cabeza espere de hacia donde tiene que ir por
    #parte del teclado.

#Comida
comida = turtle.Turtle()
    #Creamos variable "comida" como un objeto turtle.
comida.speed(0)
    #Solicitamos que el objeto aparezca al mismo tiempo que la ventana.
comida.shape("circle")
    #Le damos forma circular a nuestro objeto comida.
comida.color("brown")        #Le otorgamos el color café al objeto comida.
comida.penup()
    #Limpiamos el rastro que el modulo turtle deja cuando mueve un objeto.
comida.goto(0,100)
    #Ubicamos al objeto comida en la posición (0,100) unos lugares
    #arriba de la cabeza.

#Cuerpo Serpiente
segmentos = []

'''

```

El cuerpo de la serpiente esta dado por una lista vacia, en donde a medida que la serpiente "coma su comida" estetendra mas valores registrados, por ende, mientras más grande la serpiente, más grande la lista.

'''

#Contador

texto = turtle.Turtle()

#Creamos una variable "texto" como un objeto turtle para escribir en pantalla.

texto.speed(0)

#Solicitamos que el texto aparezca al mismo tiempo que la ventana.

texto.color("grey") #Le otorgamos el color gris a nuestro contador.

texto.penup()

#Limpiamos el rastro que el módulo turtle deja cuando actualiza al objeto.

texto.hideturtle() #Escondemos la plumilla que va a escribir el texto.

texto.goto(0,260)

#Elegimos que el texto aparezca en la parte superior, centrado.

texto.write("Score: 0 High Score: 0",

#La función write es como un print en donde se solita

align = "center", font = ("Courier", 22, "normal"))

#Que escriba los diferentes marcadores, con sus especificaciones.

#Funciones

'''

En esta parte vamos a definirle al interprete que es para nosotros arriba, abajo, derecha e izquierda, para posteriormente conectar nuestro programa a nuestro teclado.

'''

def arriba():

cabeza.direction = "up"

def abajo():

cabeza.direction = "down"

def derecha():

cabeza.direction = "right"

def izquierda():

cabeza.direction = "left"

'''

En la siguiente parte se relaciona con la línea 24 del apartado

"#Cabeza Serpiente" en donde vamos a generar el movimiento de la

"cabeza de la serpiente" a partir de sustituir el valor de "stop"

por el valor que recibimos de nuestro teclado, en donde dependiendo

de la dirección recibida esta se moverá hacia arriba, abajo,

derecha o izquierda.

```

'''
def mov():
    if cabeza.direction == "up":          #Entonces, si el valor es up.
        y = cabeza.ycor()
        #Se solicita el valor de "y" que se modifiko para almacenarlo
        en la variable y.
        cabeza.sety(y + 20)
        #Actualizamos el valor de "y" y lo aumentamos 20 para que se
        siga moviendo.

    if cabeza.direction == "down":        #Entoces si el valor es down.
        y = cabeza.ycor()
        #Se solicita el valor de "y" que se modifiko para almacenarlo
        en la variable y.
        cabeza.sety(y - 20)
        #Actualizamos el valor de "y" y lo disminuimos 20 para que se
        siga moviendo.

    if cabeza.direction == "right":       #Entonces si el valor es right.
        x = cabeza.xcor()
        #Se solicita el valor de "x" que se modificó para almacenarlo
        en la variable x.
        cabeza.setx(x + 20)
        #Actualizamos el valor de "x" y lo aumentamos 20 para que se
        siga moviendo.

    if cabeza.direction == "left":        #Entonces si el valor es left.
        x = cabeza.xcor()
        #Se solicita el valor de "x" que se modificó para almacenarlo
        en la variable x.
        cabeza.setx(x - 20)
        #Actualizamos el valor de "x" y lo disminuimos 20 para que se
        siga moviendo.

#Teclado
wn.listen()
    #Solicitamos que la pantalla tenga una reacción inmediata a
    nuestro teclado.
wn.onkeypress(arriba, "Up")
    #Al precionar la tecla "Up" se debe activar la función arriba.
wn.onkeypress(abajo, "Down")
    #Al precionar la tecla "Down" se debe activar la función abajo.
wn.onkeypress(derecha, "Right")
    #Al precionar la tecla "Right" se debe activar la función derecha.

```



```

wn.onkeypress(izquierda, "Left")
    #Al precionar la tecla "Left" se debe activar la función izquierda.

while True:
    wn.update()

    #Colisiones bordes
    if cabeza.xcor() > 280 or cabeza.xcor() < -280 or cabeza.ycor() > 280
        or cabeza.ycor() < -280:
'''
Dentro del if, hay una condición que impide que la serpiente salga
del "tablero" de 600*600, de tal, forma que si la serpiente
alcanza un valor mayor de 280 o 280 en cualquiera de los ejes, la cabeza
perderá su "cuerpo" y se posicionara en (0,0).
'''

        time.sleep(1)
        #Se le ordena detener al programa por un segundo.
        cabeza.goto(0,0)
        #Se le ordena a la cabeza reubicarse en la posición (0,0).
        cabeza.direction = "stop"    #Obliga a la cabeza a detenerse.

        #Limpiar segmentos
        for segmento in segmentos:
            segmento.goto(1000,1000)
        #Como no se puede eliminar el cuerpo de la serpiente, simplemente los
        #hace desaparecer de la ventana, enviándolos a posiciones lejanas.
        segmentos.clear()    #Limpia los valores de la lista segmentos.

        #Resetear Marcador
        score = 0    #Se le da el valor de cero a la variable score.
        texto.clear()    #Limpia el texto
        texto.write("Score: {}    High Score: {}".
            format(score, high_score),align = "center",
            font = ("Courier", 20, "normal"))

        #Imprime nuevo texto, pero conserva los valores de score y
        high_score, en el nuevo texto
        #con sus diferentes especificaciones.

        #Colisiones Comida
        if cabeza.distance(comida) < 20:
            #Define la distancia entre la cabeza y la comida.
            Comprobando si se tocan.

```

```

    x = random.randint(280,280)
    #Si es cierto, crea una coordenada x aleatoria,
    y la asigna a x.
    y = random.randint(280,280)
    #Si es cierto, crea una coordenada y aleatoria, y la asigna a y.
    comida.goto(x,y)
    #Generando una nueva comida en las coordenadas aleatorias xy.

    nuevo_segmento = turtle.Turtle()
    #Creamos una variable "nuevo_segmento" como un objeto turtle.
    nuevo_segmento.speed(0)
    #Solicitamos que el texto aparezca al mismo tiempo que la ventana.
    nuevo_segmento.shape("square")
    #Le damos forma circular a nuestro objeto nuevo_segmento.
    nuevo_segmento.color("lightgreen")
    #Le otorgamos el color verde claro al objeto nuevo segmento.
    nuevo_segmento.penup()
    #Limpiamos el rastro que el módulo turtle deja cuando actualiza al objeto.
    segmentos.append(nuevo_segmento)
    #Guardamos a nuevo_segmento en la lista segmentos.

    #Aumentar Marcador
    score += 10
    #La variable score aumenta 10 cada vez que la zabeza y la comida
    se tocan.
    if score > high_score:
    #La estructura if dice que si score se vuelve mayor a high_score...
        high_score = score
    #Otrogar los valores de score a high_score.

    texto.clear() #Limpia el texto
    texto.write("Score: {} High Score: {}".
format(score, high_score), align = "center",
font = ("Courier", 20, "normal"))

    #Imprime nuevo texto, pero conserva los valores de score y
    high_score, en el nuevo texto
    #con sus diferentes especificaciones.

    #Mover Cuerpo Serpiente
    totalSeg = len(segmentos)
    #Asignamos a totalSeg el número de datos que hay en la lista
    segmentos.
    for index in range(totalSeg - 1, 0, -1):
    #Se itera entre cada uno de los segmentos.

```

```

"""
Lo que sucede aquí es que el ultimo elemento de mi lista siga al
penúltimo y así sucesivamente. Empezando en la posición cero,
por eso se decrece uno.
"""

    x = segmentos[index - 1].xcor()
    #Obtenemos la coordenada x del segmento anterior.
    y = segmentos[index - 1].ycor()
    #Obtenemos la coordenada y del segmento anterior.
    segmentos[index].goto(x,y)
    #Hacemos que los segmento se muevan a las coordenadas xy.

if totalSeg > 0:
    x = cabeza.xcor()      #Obtenemos la coordenada x
    y = cabeza.ycor()      #Obtenemos la coordenada y
    segmentos[0].goto(x,y)
    #Se pide que el elemento se mueva donde está la cabeza.

mov()

#Colisiones Cuerpo
for segmento in segmentos:
    if segmento.distance(cabeza)<20:
        #La estructura if define que si, cabeza y segmentos se tocan...
        time.sleep(1)
        #Se le ordena detener al programa por un segundo.
        cabeza.goto(0,0)
        #Se le ordena a la cabeza reubicarse en la posición (0,0).
        cabeza.direction = "stop"      #Obliga a la cabeza a detenerse.

        #Limpiar segmentos
        for segmento in segmentos:
            #Como no se puede eliminar el cuerpo de la serpiente, simplemente
            segmento.goto(1000,1000)      #Los hace desaparecer de la
            #ventana, enviándolos a posiciones lejanas.
            segmentos.clear()
        #Limpia los valores de la lista segmentos.

        score = 0      #Se le da el valor de cero a la variable score.
        texto.clear()  #Limpia el texto
        texto.write("Score: {}      High Score: {}".
format(score, high_score),align = "center",
font = ("Courier", 20, "normal"))

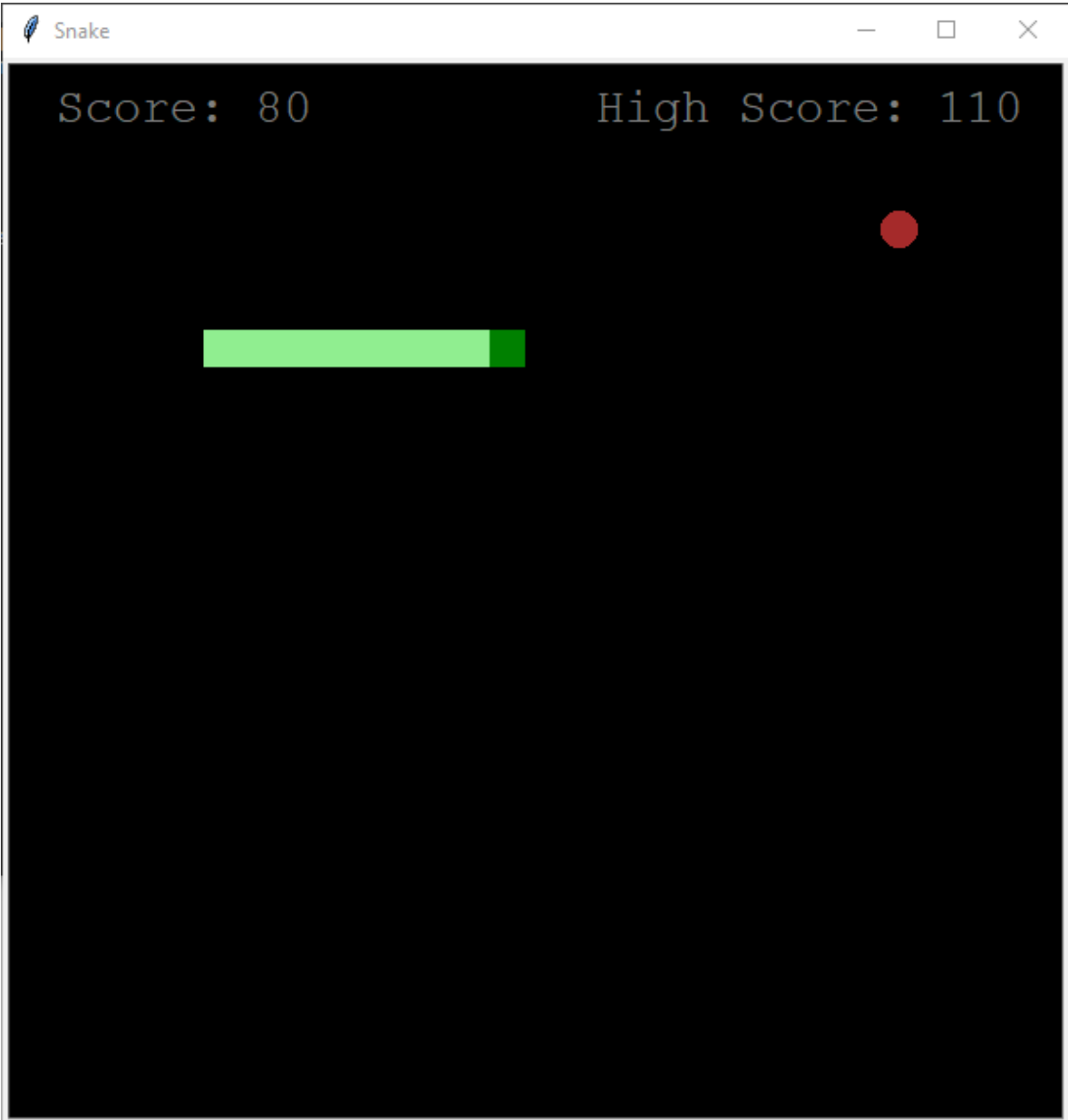
```

```
#Imprime nuevo texto, pero conserva los valores de score y  
high_score, en el nuevo texto  
#Con sus diferentes especificaciones.
```

```
time.sleep(posponer)  
#Hace el movimiento de nuestro programa sea más lento.
```

Funcionamiento:





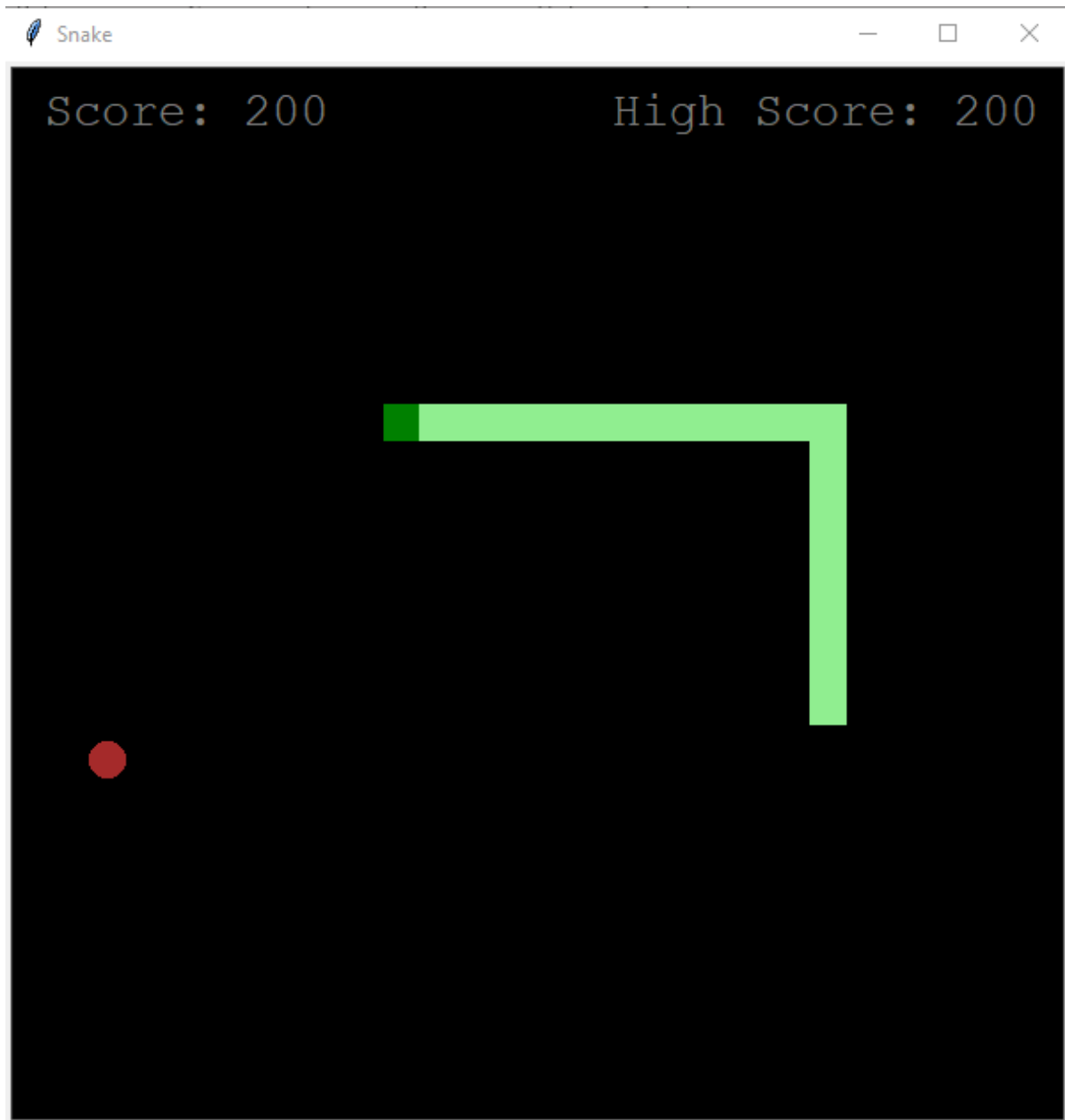


Tabla 1:

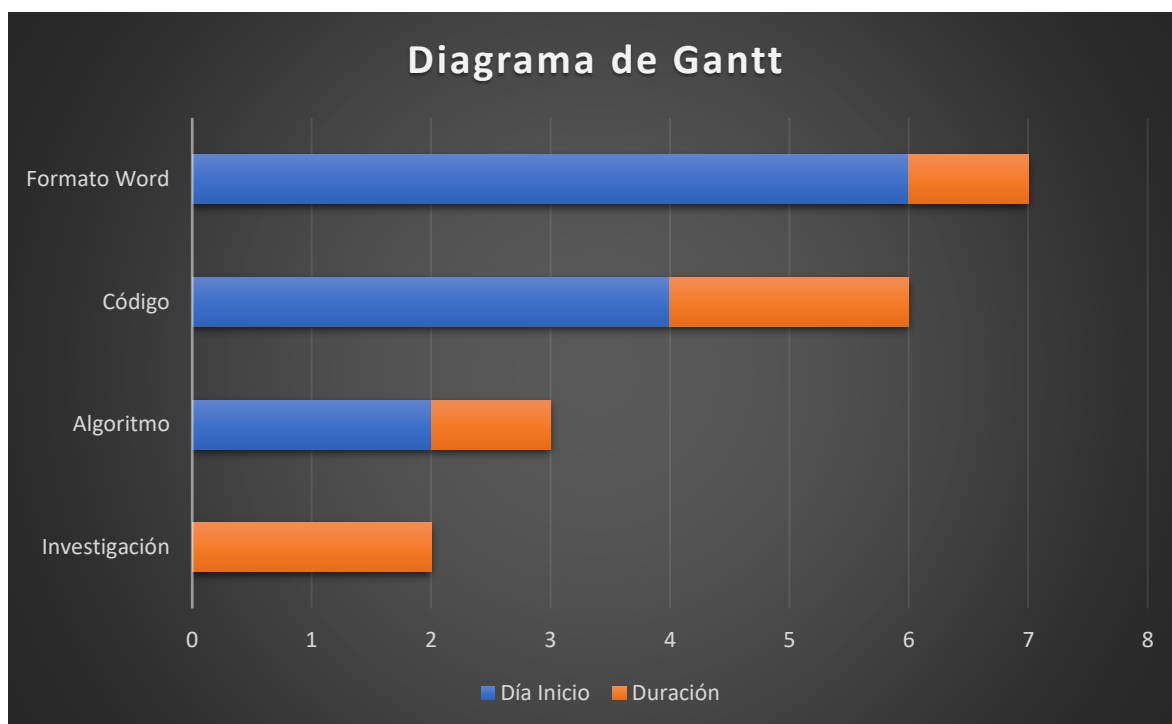
Software	Hardware
Visual Studio Code	Celular
Python 3.9	Laptop
Consola de Comandos	
You Tube	

Tabla 2:

Proyecto (Snake)		Costos
Software	N/A	0.0
Hardware	N/A	0.0
Tiempo	10.0	-
C/Hora	1,000.0	10,000.0
Total	-	10,000.0

Diagrama de Gantt:

Tarea	Día Inicio	Duración
Investigación	0	2
Algoritmo	2	1
Código	4	2
Formato Word	6	1



Conclusión:



Las estructuras de datos y algoritmos son demasiado importantes en nuestro día a día, ya que están presentes en casi todo lo que conocemos, estas estructuras son muy importantes para la formación de un ingeniero, y son necesarias en la formación de cada profesional cuando se trata de entender y desarrollar el funcionamiento de cualquier cosa que aplique a la ingeniería.

Aprender estructura de datos y algoritmos no solo sirve para manipular maquinas, sino que también nos sirve a nosotros como ingenieros para desarrollarnos en el ámbito organizacional, un ingeniero que no se sabe organizar nunca va a poder conocer el funcionamiento de una máquina o programa.

Al realizar este proyecto, entendí muchas cosas, pude mejorar mi entendimiento hacia las estructuras for, while e if, las cuales ayudaron a simplificar mucho el código, pude conocer un modo de crear un tablero en Python y poder trabajar sobre este mismo, aprendí a trabajar con elementos gráficos los cuales pude manipular para que dieran vida a un videojuego, y con la ayuda de las estructuras lógicas, poder darles sentido y recrear el juego de Snake.

Referencias:

Video de YouTube

- Nombre: Colección ( Python - Juego Snake )
- Canal: YouDevs
- Fecha de publicación: 30/agosto/2019 – 01/noviembre/2019
- Fecha de consulta: 06/agosto/2021
- Enlace del video:

<https://www.youtube.com/playlist?list=PLuaRu05D7vP5ij2oZIGHatrF9ZUetf2VA>