Documentación del grupo

Creación del grupo:

→Cuando al inicio nuestro profesor nos explico el nuevo proyecto que sería en grupo, los cuales los requisitos sería:

- Mínimo 3 integrantes
- Utilización del GitHub
- Un ejecutable
- Compaginar distintos archivos de código
- Un programa de python
- Manual para el usuario
- Documentación del trabajo

Teniendo en cuenta todo los requisitos mencionados anteriormente, nos juntamos y se creó el mejor grupo para este proyecto.
Todos tenemos nuestras virtudes y defectos a la hora de utilizar python, sin embargo donde un integrante flaqueaba otro integrante lo ayudaba para que pudiese comprender el código, así pudimos mejorar nuestras habilidades con python.

Lluvia de ideas:

→Una vez creado el grupo decidimos proceder a hacer una lluvia de ideas. Las cuales eran muy exigentes para nuestro nivel de programación, por ejemplo tuvimos una idea que era recrear el famoso videojuego de ordenadores League Of Legends (LOL), pero no sabíamos crearlo por lo cual decidimos abandonar la idea principal.
Cómo abandonamos la idea, procedimos a continuar con la lluvia lo cual esa lluvia nos llevó a una excelente idea, un programa que identifica el rostro facial de un usuario a través de la cámara principal del dispositivo y automáticamente encontrase una similitud de el rostro del usuario junto a una base de datos, con imágenes robóticas de otros usuarios, con lo cual en resultado final otorga una imagen con similitudes faciales, en el cual mostraría la imagen por pantalla para que el usuario supiese cómo sería su rostro con versiones robóticas.
Sin embargo decidimos no proseguir con la idea porque creímos que se le otorga más relevancia a la base de datos que al mismo programa de python y como estamos en programación queríamos un reto a nivel de código python no de SQL.

Como no teníamos nada claro por el momento y teníamos que ponernos manos en la masa, teníamos que decidir dos ideales finales que nos surgieron en la ya nombrada anteriormente lluvia de ideas. Fueron las mejores ideas que tuvimos, eran un reto de verdad, pero solo podíamos elegir una. La primera idea era crear un videojuego el cual se definiría como novela visual, la idea de esta novela visual es una historia donde las acciones del usuario proporciona diferentes continuidades de la novela con diferentes eventos y a la vez tener experiencias completamente diferentes y por lo cual daría al usuario una experiencia de juego distinta según sus avances y iniciar nuevas partidas pudiendo disfrutar de finales variados y dar la sensación de que sea un juego completamente distinto, estaría hecho de tal manera que el usuario estuviese con ganas de disfrutar de cada una de las ramas que proporciona la novela visual.
Teniendo en cuenta esto creíamos que era una idea dentro de nuestras posibilidades, con un nivel de muy complexo de python, pero cuando todo era felicidad nos dimos cuenta de la mayor adversidad en nuestro programa ideal; el apartado visual y los guiones; esto consumiría todo el tiempo que teníamos por consecuencia no nos daría el suficiente tiempo para que el proyecto esté en su mejor versión. Como nuestro nivel de dibujo es nulo, tendríamos que depender de personas ajenas al grupo del proyecto o incluso ajenas al centro STUCOM. Y el guión podría ser algo básico siendo sinceros no sería una obra magna pero queríamos cautivar a los jugadores por lo tanto y a nuestro pesar tuvimos que descartar la novela visual.

Idea final:

→Como habréis supuesto la idea final fue recrear el famoso videojuego de los años 80’s, Snake, tuvimos la sensación que seria una opcion donde llevaremos nuestros conocimientos al máximo, con un apartado gráfico complicado pero que con esfuerzo lo podríamos sacar adelante; y lo hicimos; como es un juego que todo aficionado al entretenimiento digital conoce, sería ideal para el proyecto.

Organización:

→Ahora que ya teníamos la idea tocaba empezar a organizarnos y para eso el profesor nos dio la opción de utilizar el github, podiamos subir codigo, imágenes, archivos de audio y lo más importante del github es que se puede unir al Visual Studio, nos ayudó mucho para subir cambios y descargar los cambios de los integrantes del grupo, si no fuese por el Github tendríamos cambios en los archivos y si alguien cambiaba algo se pondría encima del archivo anterior y solo relanterezia el trabajo.

→Estábamos por la mitad del proyecto y el profesor nos volvió a comentar que en el proyecto hay que entregar un ejecutable hecho con python, era un código corto y sencillo.

→Anteriormente de hacer el proyecto ya hicimos alguna práctica que se tenía que hacer en distintos documentos y luego “unirlos”, hemos hecho lo mismo, todo tiene su propio documento donde tiene todo y luego si necesitamos alguna clase utilizamos un ’import’.

Documentación de código

→En la documentación de nuestro código, explicaremos cada archivo de python con sus comandos. Al principio tuvimos muchas dificultades, sobretodo a la hora de hablar con el grupo, no nos podíamos ponernos de acuerdo y en alguna ocasión teníamos el trabajo repetido pero de manera distinta porque hicimos lo mismo
Empezamos a funcionar mejor, cuando nos estuvieron comentando el profesorado conforme al principio de cada clase deberíamos realizar una pequeña reunión de nuestro grupo para comentar el trabajo el cual nos ayudará para organizar el trabajo en el aula y así poder progresar de manera optimizada y eficaz, con lo cual nos ayudará para entender que cada integrante del grupo tendría una utilidad distinta para el trabajo y así no afectasen los cambios en el código. A últimas instancias para entregar el proyecto decidimos comprimir archivos y mejorar el código y por consecuencia hubo menos archivos,archivos de audio y carpetas.

- Juego
- Lógica
- Ranking
- Menu
- ALBERT ASX PROJECT 1.mp3
- ALBERT ASX PROJECT 2.mp3
- Images / Carpetas
- Songs /Carpetas

→Ahora empezaré con la explicación del programa de python,seguiré los puntos nombrados anteriormente. En la explicación del programa habrán capturas de pantalla, un texto explicando que hace y algún texto más extenso para algún comando.

Juego:

```
1   import pygame
2   import sys
3   from pygame.locals import *
4   from random import randint
5   import random
6
7   def cell2coord(row, col):
8       return (275 + (CELLSIZE * row), 125 + (CELLSIZE * col))
9
10  CELLSIZE = 50
11
12  def generar_posicion_aleatoria():
13      x = random.randint(0, 12)
14      y = random.randint(0, 12)
15      return x, y
16
17  serpiente = ""
18  fila = 6
19  col = 1
20  serpiente_direccion = 'derecha'
```

→**La línea 1** Importa el módulo principal pygame, que proporciona todas las funcionalidades principales para desarrollar juegos y aplicaciones multimedia.
→**La línea 2** Importa el módulo sys, el cual proporciona funciones y variables relacionadas con la interacción del programa con el sistema.
→**La línea 3** Importa todos los nombres (clases, funciones, constantes, etc.) del módulo 'locals' dentro del paquete pygame. Este módulo contiene definiciones de constantes para eventos y teclas especiales.
→**La línea 4** Importa la función randint del módulo random. La función randint se utiliza para generar números enteros aleatorios en un rango específico.
→**La línea 5** Importa el módulo random, que proporciona funciones para generar números aleatorios y realizar operaciones relacionadas con la aleatoriedad
→**La línea 7 y 8** Hace que la función cell2coord(row, col) tome dos argumentos: row y col, que representan las coordenadas de una celda en una cuadrícula. La función devuelve un par de coordenadas (x, y)

correspondientes a la posición en píxeles de esa celda en relación a un sistema de coordenadas específico.
→**La línea 10** Pone un valor constante a CELLSIZE que es 50 para determinar la ubicación de elementos gráficos.
→**La línea 12** La función generar_posicion_aleatoria() genera una posición aleatoria dentro de una cuadrícula de 13x13 (con índices que van desde 0 hasta 12).
→**La línea 13** Utiliza la función randint() del módulo random para generar un número entero aleatorio en el rango de 0 a 12 (ambos inclusive) y lo asigna a la variable x.
→**La línea 14** De manera similar a la línea anterior, esta línea genera otro número entero aleatorio en el rango de 0 a 12 y lo asigna a la variable y.
→**La línea 15** La función devuelve una tupla con los valores de x e y. Esto significa que cuando llames a la función obtendrás una tupla que contiene dos valores, la posición aleatoria en el eje x y en el eje y.
→**La línea 17** Esta variable inicialmente se establece como una cadena vacía.
→**La línea 18-20** Pone la serpiente en el mapa, la fila y columna donde aparecerá y hacia donde mira

```
22  def pintar_cabeza(ventana):
23      global serpiente
24      global serpiente_direccion
25      # SERPIENTE, IMAGEN Y POSICION
26      if serpiente_direccion == 'derecha':
27          serpiente = pygame.image.load("images/serpiente/cd.png")
28      elif serpiente_direccion == 'izquierda':
29          serpiente = pygame.image.load("images/serpiente/ci.png")
30      elif serpiente_direccion == 'arriba':
31          serpiente = pygame.image.load("images/serpiente/cb.png")
32      elif serpiente_direccion == 'abajo':
33          serpiente = pygame.image.load("images/serpiente/ca.png")
34      pos = cell2coord(col, fila)
35      posX = pos[0]
36      posY = pos[1]
37      ventana.blit(serpiente, (posX, posY))
38
39  manzana = ""
40  colman, filaman = generar_posicion_aleatoria()
41
42  def pintar_manzana(ventana):
43      global manzana
44      manzana = pygame.image.load("images/comida.png")
45      pos = cell2coord(colman, filaman)
46      posX = pos[0]
47      posY = pos[1]
48      ventana.blit(manzana, (posX, posY))
```

→**La línea 22** La función pintar_cabeza(ventana) se encarga de dibujar la cabeza de la serpiente en una ventana o superficie de juego.

→**La línea 23** indica que la variable serpiente se refiere a una variable global y no a una variable local dentro de la función.
→**La línea 24** Similar a la línea anterior, esta línea declara que serpiente_direccion se refiere a una variable global.
→**La línea 26-27** Si serpiente_direccion es igual a 'derecha', se carga la imagen de la serpiente con la cabeza mirando hacia la derecha.
(Las imágenes de la serpiente se cargan desde archivos en la carpeta "images/serpiente" con las extensiones de archivo correspondientes.)
→**La línea 28-29** Si serpiente_direccion es igual a 'izquierda', se carga la imagen de la serpiente con la cabeza mirando hacia la izquierda.
→**La línea 30-31** Si serpiente_direccion es igual a 'arriba', se carga la imagen de la serpiente con la cabeza mirando hacia arriba**.**
→**La línea 32-33** Si serpiente_direccion es igual a 'abajo', se carga la imagen de la serpiente con la cabeza mirando hacia abajo.
→**La línea 34** llama a la función cell2coord() con los argumentos col y fila para obtener las coordenadas (x, y) en píxeles correspondientes a la posición de la cabeza de la serpiente.
→**La línea 35-36** Las variables posX y posY se asignan con los valores x e y de la tupla pos, respectivamente.
→**La línea 37** Esta línea dibuja la imagen de la serpiente en la posición (posX, posY) en la ventana o superficie de juego especificada.
→**La línea 39** Esta variable se inicializa como una cadena vacía.
→**La línea 40** llama a la función generar_posicion_aleatoria() y se asignan los valores retornados a las variables colman y filaman. Esto se utiliza para obtener una posición aleatoria para la manzana.
→**La línea 42** Hace que la función pintar_manzana(ventana) se encargue de dibujar una imagen de manzana en una superficie de juego.
→**La línea 43** Indica que la variable manzana se refiere a una variable global y no a una variable local dentro de la función.
→**La línea 44** Se carga la imagen de la manzana desde el archivo "images/comida.png" y se asigna a la variable manzana.
→**La línea 45** Se llama a la función cell2coord() con los argumentos colman y filaman para obtener las coordenadas (x, y) en píxeles correspondientes a la posición de la manzana.
→**La línea 46-47** Las variables posX y posY se asignan con los valores x e y de la tupla pos, respectivamente.

→**La línea 48** Esta línea dibuja la imagen de la manzana en la posición (posX, posY) en la ventana o superficie de juego especificada.

```
50  def abrir_juego():
51      global serpiente
52      global serpiente_direccion
53      global fila
54      global col
55      global colman
56      global filaman
```

→**La línea 50** La función abrir_juego() en tu programa de Python se encarga de inicializar el juego

→**La línea 51-56** indica que la variables se refieren a una variable global y no a una variable local dentro de la función.

```
58      # INICIACION DE PYGAME
59      pygame.init()
60      # VENTANA TAMAÑO
61      ventana = pygame.display.set_mode((1200, 900))
62
63      # VELOCIDAD POR SEGUNDO A CUANTO SE MUEVE LA SERPIENTE
64      FPS = 10
65      reloj = pygame.time.Clock()
66
67      pintar_cabeza(ventana)
68
69      # CUADRICULA, IMAGEN Y POSICION
70      cuadricula = pygame.image.load("images/cuadricula.png")
71      cuadriculpos = (275, 125)
72
73      # FONDO, IMAGEN Y POSICION
74      superfondo = pygame.image.load("images/superfondo.png")
75      superfondopos = (0, 0)
76
77      # BORDE, IMAGEN Y POSICION
78      borde = pygame.image.load("images/borde.png")
79      bordepos = (266, 116)
80
81      # PRUEBA BOTON QUIT, IMAGEN Y POSCION
82      quit = pygame.image.load("images/quit.png")
83      quitpos = (800, 800)
84
85      # EL MARCO QUE SI PULSAS DENTRO, SALES
86      #                   posicion   ,  tamaño
87      botonquit = pygame.Rect(800, 800, 300, 85)
```

→**La línea 59** Inicializa el módulo pygame y prepara el entorno para utilizar sus funciones y objetos.
→**La línea 61** Se crea una ventana con un tamaño de 1200 píxeles de ancho y 900 píxeles de alto utilizando la función set_mode() de pygame. La variable ventana se utiliza para hacer referencia a esta ventana.
→**La línea 64-65** Se define la velocidad de cuadros por segundo (FPS) a 10 y se crea un objeto reloj utilizando la función Clock() de pygame. Este objeto se utilizará más adelante para controlar la velocidad de actualización del juego.
→**La línea 67** Se llama a la función pintar_cabeza() pasando la variable ventana como argumento.

→**La línea 70-71** Carga una imagen de una cuadrícula desde el archivo "images/cuadricula.png" y se almacena en la variable cuadrícula. La tupla (275, 125) representa la posición en píxeles donde se colocará la imagen de la cuadrícula en la ventana.
→**La línea 74-75** Carga una imagen de fondo desde el archivo "images/superfondo.png" y se almacena en la variable superfondo. La tupla (0, 0) representa la posición en píxeles donde se colocará la imagen del fondo en la ventana.
→**La línea 78-79** Carga una imagen de un borde desde el archivo "images/borde.png" y se almacena en la variable borde. La tupla (266, 116) representa la posición en píxeles donde se colocará la imagen del borde en la ventana.
→**La línea 82-83** Carga una imagen de un botón de salida desde el archivo "images/quit.png" y se almacena en la variable quit. La tupla (800, 800) representa la posición en píxeles donde se colocará la imagen del botón de salida en la ventana.
→**La línea 87** Se crea un objeto de tipo Rect utilizando la función Rect() de pygame. Este objeto representa un rectángulo y se utiliza para definir un área sensible al clic dentro de la ventana. El rectángulo se posiciona en (800, 800) y tiene un ancho de 300 píxeles y una altura de 85 píxeles.

```
89          # BUCLE DE EL JUEGO
90          while True:
91              # ACTIVACION DE IMAGENES
92              ventana.blit(superfondo, superfondopos)
93              ventana.blit(cuadricula, cuadriculpos)
94              pintar_cabeza(ventana)
95              ventana.blit(borde, bordepos)
96              pintar_manzana(ventana)
97              ventana.blit(quit, quitpos)
98
99              # SI LAS POSICIONES TOCAN ESAS COORDENADAS, SE CIERRA EL JUEGO
100             if col < 0 or col > 12 or fila < 0 or fila > 12:
101                 exit()
102
103             # SI SE TOCA UNA TECLA, PASA ALGO, RESPECTIVAMENTE
104             for evento in pygame.event.get():
105                 if evento.type == QUIT:     # AQUI PARA CERRAR EL PROGRAMA
106                     pygame.quit()
107                     sys.exit()
108
109                 # AQUI LA PARTE QUE HACE FUNCIONAR EL QUIT
110                 if evento.type == pygame.MOUSEBUTTONDOWN:
111                     if botonquit.collidepoint(evento.pos):
112                         pygame.quit()
113                         quit()
114
```

→**La línea 90** Crea un bucle infinito, lo que significa que el contenido dentro del bucle se ejecutará continuamente hasta que el programa se cierre.

→**La línea 92-97** Se encargan de dibujar diferentes elementos en la ventana en cada iteración del bucle. Se dibujan el fondo, la cuadrícula, la cabeza de la serpiente, el borde, la manzana y el botón de salida en la ventana.

→**La línea 100-101** Verifica si las variables col y fila están fuera de los límites permitidos (0-12) para el juego de la serpiente. Si alguna de estas condiciones es verdadera, se llama a la función exit() para salir del programa.

→**La línea 104** Se utiliza para obtener los eventos de entrada del usuario en cada iteración del bucle.

→**La línea 105-107** Esta condición verifica si el evento actual es de tipo QUIT, lo cual indica que el usuario ha solicitado cerrar el programa (por ejemplo, haciendo clic en el botón de cerrar de la ventana). Si se cumple esta condición, se llama a las funciones pygame.quit() y sys.exit() para cerrar correctamente el programa.
→**La línea 110-113** Esta condición verifica si el evento actual es de tipo MOUSEBUTTONDOWN, lo cual indica que se ha producido un clic del mouse. Si se cumple esta condición, se verifica si las coordenadas del evento (evento.pos) se encuentran dentro del área del rectángulo del botón de salida (botonquit.collidepoint(evento.pos)). Si es así, se llama a las funciones pygame.quit() y quit() para cerrar el programa.

```
115                 # AQUI EL MOVIMIENTO DE LA SERPIENTE CON SU FOTO
116                 elif evento.type == KEYDOWN:
117                     if evento.key == K_a and serpiente_direccion != 'derecha':
118                         serpiente_direccion = 'izquierda'
119                     elif evento.key == K_d and serpiente_direccion != 'izquierda':
120                         serpiente_direccion = 'derecha'
121                     elif evento.key == K_w and serpiente_direccion != 'abajo':
122                         serpiente_direccion = 'arriba'
123                     elif evento.key == K_s and serpiente_direccion != 'arriba':
124                         serpiente_direccion = 'abajo'
125 
126             # MOVIMIENTO DE LA SERPIENTE
127             if serpiente_direccion == 'derecha':
128                 col += 1
129             elif serpiente_direccion == 'izquierda':
130                 col -= 1
131             elif serpiente_direccion == 'arriba':
132                 fila -= 1
133             elif serpiente_direccion == 'abajo':
134                 fila += 1
135 
136             # Si la posición de la serpiente coincide con la posición de la manzana, generar una nueva posición para la manzana
137             if (fila, col) == (filaman, colman):
138                 colman, filaman = generar_posicion_aleatoria()
139 
140             pygame.display.update()
141             reloj.tick(FPS)
142 
143     abrir_juego()
```

→**La línea 116** Esta condición verifica si se ha presionado una tecla en el evento actual.
→**La línea 117-118** Esta condición verifica si la tecla presionada es la tecla "a" y la dirección actual de la serpiente no es hacia la derecha. Si se cumple esta condición, se actualiza la dirección de la serpiente a la izquierda.
→**La línea 119-120** Similar a la línea anterior, esta condición verifica si la tecla presionada es la tecla "d" y la dirección actual de la serpiente no es hacia la izquierda. Si se cumple esta condición, se actualiza la dirección de la serpiente a la derecha.

→**La línea 121-122** Esta condición verifica si la tecla presionada es la tecla "w" y la dirección actual de la serpiente no es hacia abajo. Si se cumple esta condición, se actualiza la dirección de la serpiente hacia arriba.
→**La línea 123-124** Similar a la línea anterior, esta condición verifica si la tecla presionada es la tecla "s" y la dirección actual de la serpiente no es hacia arriba. Si se cumple esta condición, se actualiza la dirección de la serpiente hacia abajo.
→**La línea 127-134** Actualiza la posición de la serpiente según su dirección. Dependiendo de la dirección de la serpiente, se incrementa o decrementa el valor de col o fila. Por ejemplo, si la serpiente se mueve hacia la derecha, se incrementa el valor de col en 1.
→**La línea 137-138** Verifica si la posición de la serpiente coincide con la posición de la manzana. Si es así, se llama a la función generar_posicion_aleatoria() para generar una nueva posición para la manzana.
→**La línea 140-141** Actualiza la ventana con los cambios realizados, y reloj.tick(FPS) se utiliza para limitar la velocidad del bucle a la cantidad de cuadros por segundo (FPS).
→**La línea 143** Llama la función abrir_juego() para iniciar el juego y ejecutar el bucle principal.

Lógica:

En este archivo está el apartado lógico del juego

→**La línea 1** Proporciona funciones relacionadas con la generación de números aleatorios. Puedes usar funciones como random.randint(a, b) para generar un número entero aleatorio en el rango de a a b, random.random() para obtener un número flotante aleatorio entre 0 y 1, y muchas otras funciones para diferentes tipos de generación de números aleatorios.
→**La línea 2** Proporciona clases y funciones para trabajar con fechas y horas en Python. Puedes utilizar la clase datetime para representar una fecha y hora específica, y luego utilizar métodos y atributos de esta clase para realizar operaciones y manipulaciones con fechas y horas.

```
5   def mover_cabeza(serp, dir):
6       x = serp[0][0]
7       y = serp[0][1]
8       if dir == "w":
9           x -= 1
10      if dir == "s":
11          x += 1
12      if dir == "a":
13          y -= 1
14      if dir == "d":
15          y += 1
16      serp.insert(0, [x, y])
17
18  def desplazar_cola(serp):
19      serp.pop()
20
21  def mostrar_serp(serp, tablero):
22      tablero[serp[0][0]][serp[0][1]] = 8
23      # dibuixar cap
24      for celda in serp [1:]:
25          tablero[celda[0]][celda[1]] = 1
26          #dicuixar cua
```

→**La línea 5-18** Esta función recibe una lista serp que representa la serpiente y una dirección dir (puede ser "w", "s", "a" o "d"). La función actualiza la posición de la cabeza de la serpiente en función de la dirección proporcionada. Si la dirección es "w", se decrementa el valor de la coordenada x de la cabeza en 1. Si la dirección es "s", se incrementa el valor de la coordenada x de la cabeza en 1. Si la dirección es "a", se decrementa el valor de la coordenada y de la cabeza en 1. Si la dirección es "d", se incrementa el valor de la coordenada y de la cabeza en 1. Luego, se inserta la nueva posición de la cabeza al principio de la lista serp.

→**La línea 18** Esta función recibe una lista serp que representa la serpiente.

→**La línea 19** La función elimina el último elemento de la lista serp, lo que corresponde al desplazamiento de la cola de la serpiente.

→**La línea 21-25** Esta función recibe la lista serp que representa la serpiente y una matriz tablero que representa el tablero del juego. La función actualiza el valor de las celdas del tablero que están ocupadas por la serpiente. En primer lugar, se establece el valor 8 en la celda correspondiente a la cabeza de la serpiente (tablero[serp[0][0]][serp[0][1]] = 8). Luego, se tira sobre el resto de las celdas de la serpiente y se establece el valor 1 en esas celdas (tablero[celda[0]][celda[1]] = 1), lo que representa el cuerpo de la serpiente en el tablero.

```
29      def savepoints():
30          archivo = open("puntuacion.txt", "a")
31          archivo.write(str(datetime.datetime.now())+','+str(nManzanas)+"\n")
32          archivo.close()
33
34
35      def mostrar():
36          tablero = mostrar_tablero()
37          mostrar_serp(serp, tablero)
38          mostrar_manzana(manzpos, tablero)
39          for fila in tablero:
40              print(fila)
41
42
43      def generar_posicion_aleatoria():
44          x = random.randint(0, 8)
45          y = random.randint(0, 8)
46          return [x, y]
```

→**La línea 29-32** Esta función se encarga de guardar los puntos obtenidos en un archivo de texto llamado "puntuacion.txt". Abre el archivo en modo de apertura ("a") para agregar nuevos contenidos al final del archivo. Luego, escribe una cadena que contiene la fecha y hora actual obtenida de datetime.datetime.now(), seguida de una coma y el número de manzanas nManzanas. Finalmente, cierra el archivo.

→**La línea 35-40** Esta función muestra el estado actual del juego. Utiliza otras funciones para mostrar el tablero, la serpiente y la manzana. Llama a la función mostrar_tablero() para obtener el estado actual del tablero y luego llama a mostrar_serp() y mostrar_manzana() para mostrar la serpiente y la manzana respectivamente. Finalmente, itera sobre las filas del tablero e imprime cada fila.

→**La línea 43-46** Esta función genera una posición aleatoria en forma de una lista [x, y]. Utiliza la función random.randint(a, b) para generar dos números enteros aleatorios x e y en el rango de 0 a 8. Luego, devuelve la lista [x, y] que representa la posición aleatoria.

```
50  def crear_manzana(serp):
51      manzpos = generar_posicion_aleatoria()
52      if manzpos in serp:
53          return crear_manzana(serp)
54      else:
55          return manzpos
56
57  def mostrar_manzana(manzpos, tablero):
58      tablero[manzpos[0]][manzpos[1]] = 2
59
60
61  def mostrar_tablero():
62      # insertar imatge taulell
63      return [
64          [0, 0, 0, 0, 0, 0, 0, 0, 0],
65          [0, 0, 0, 0, 0, 0, 0, 0, 0],
66          [0, 0, 0, 0, 0, 0, 0, 0, 0],
67          [0, 0, 0, 0, 0, 0, 0, 0, 0],
68          [0, 0, 0, 0, 0, 0, 0, 0, 0],
69          [0, 0, 0, 0, 0, 0, 0, 0, 0],
70          [0, 0, 0, 0, 0, 0, 0, 0, 0],
71          [0, 0, 0, 0, 0, 0, 0, 0, 0],
72          [0, 0, 0, 0, 0, 0, 0, 0, 0]
73      ]
74
75  serp = [[4, 4]]
76  dir = "O"
77
78
79  manzpos = crear_manzana(serp)
80  nManzanas = 0
81  dirant= ""
82  posant= [0,0]
```

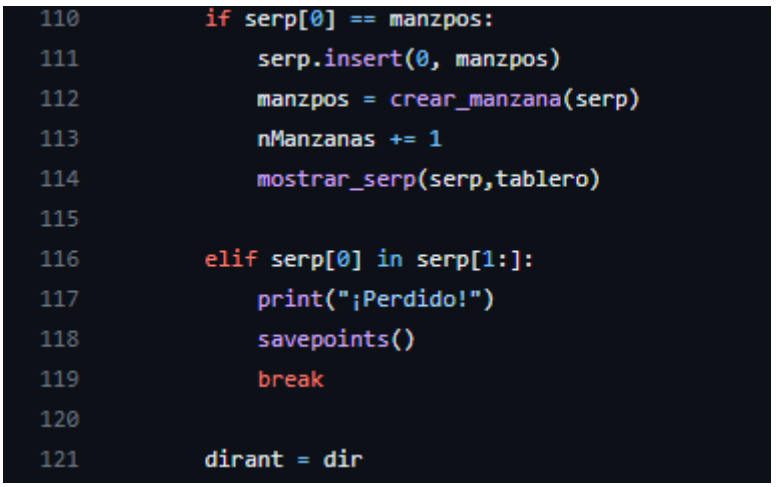
→**La línea 50-55** Esta función genera una posición aleatoria para la manzana en el tablero. Recibe la lista serp que representa la serpiente actual. Primero, genera una posición aleatoria utilizando la función generar_posicion_aleatoria(). Luego, verifica si la posición generada se encuentra dentro de la serpiente. Si la posición está ocupada por la serpiente, vuelve a llamar recursivamente a la función para generar una nueva posición. Si la posición generada no está ocupada por la serpiente, la devuelve como la posición de la manzana.
→**La línea 57-58** Esta función actualiza el tablero para mostrar la manzana en la posición manzpos. Recibe la posición de la manzana y la matriz tablero que representa el tablero del juego. Asigna el valor 2 en el tablero en la posición correspondiente a la manzana
(tablero[manzpos[0]][manzpos[1]] = 2).
→**La línea 61-82** Esta función crea y devuelve una matriz que representa el tablero del juego. La matriz es una cuadrícula de 9x9 donde inicialmente todas las celdas tienen el valor 0.
Serp representa la serpiente inicial con una posición [4, 4], dir es una variable que almacena la dirección actual de la serpiente nManzanas es una variable que almacena el número de manzanas recogidas y posant y dirant es una variable para almacenar información anterior sobre la posición y dirección.

```
84      while True:
85
86          tablero = mostrar_tablero()
87          mostrar()
88          dir = input("dir:")
89          if dir == "w" and dirant == "s":
90              dir = "s"
91
92          if dir == "s" and dirant == "w":
93              dir= "w"
94
95          if dir == "a" and dirant == "d":
96              dir = "d"
97
98          if dir == "d" and dirant == "a":
99              dir = "a"
100
101         mover_cabeza(serp, dir)
102         desplazar_cola(serp)
```

→**La línea 84** El bucle while True permite que este segmento de código se ejecute continuamente, donde el usuario ingresa la dirección y la serpiente se mueve en consecuencia.
→**La línea 86** Llama a la función mostrar_tablero() para obtener la representación actual del tablero del juego y se asigna a la variable tablero.
→**La línea 87** Llama a la función mostrar() para mostrar el estado actual del juego, incluyendo el tablero, la serpiente y la manzana.
→**La línea 88** Solicita al usuario que ingrese la dirección deseada para mover la serpiente. La entrada se guarda en la variable dir.
→**La línea 89-99** Son condiciones que verifican si la dirección ingresada por el usuario es opuesta a la dirección anterior (dirant). Si es así, se ajusta la dirección para evitar que la serpiente se mueva en la dirección opuesta y se colapse.
→**La línea 101** Se llama a la función mover_cabeza() para mover la cabeza de la serpiente en la dirección indicada por el usuario. Esta función actualiza las coordenadas de la cabeza de la serpiente según la dirección especificada.
→**La línea 102** Se llama a la función desplazar_cola() para eliminar la última posición de la serpiente, es decir, se desplaza la cola.

```
104         if serp[0][0] < 0 or serp[0][0] >= len(tablero) or serp[0][1] < 0 or serp[0][1] >= len(tablero[0]):
105
106             print("¡Perdido!")
107             savepoints()
108             break
```

→**La línea 104** Este bloque de código verifica si la posición de la cabeza de la serpiente está fuera de los límites del tablero. Si se cumple alguna de estas condiciones, se ejecuta el siguiente código
→**La línea 106** Imprime el mensaje "¡Perdido!" en la consola. Esto indica que el juego ha terminado y el jugador ha perdido.
→**La línea 107** Llama a la función savepoints() para guardar la puntuación del juego. Esto sugiere que esta función se encarga de guardar los puntos o algún registro relacionado con el juego.
→**La línea 108** Rompe el if, lo que significa que el programa saldrá del bucle y finalizará la ejecución del juego.



```
110         if serp[0] == manzpos:
111             serp.insert(0, manzpos)
112             manzpos = crear_manzana(serp)
113             nManzanas += 1
114             mostrar_serp(serp,tablero)
115 
116         elif serp[0] in serp[1:]:
117             print("¡Perdido!")
118             savepoints()
119             break
120 
121         dirant = dir
```

→**La línea 110** Esta condición verifica si la posición de la cabeza de la serpiente coincide con la posición de la manzana (manzpos). Si es así, significa que la serpiente ha alcanzado la manzana y se ejecutan las siguientes acciones

→**La línea 111** Inserta la posición de la manzana al principio de la lista de posiciones de la serpiente. Esto hace que la serpiente crezca en longitud al agregar una nueva cabeza en la posición de la manzana.

→**La línea 112** Genera una nueva posición para la manzana llamando a la función crear_manzana(serp). Esto asegura que la nueva posición de la manzana esté fuera de la serpiente.

→**La línea 113** Incrementa el contador de manzanas (nManzanas) en uno para realizar un seguimiento de cuántas manzanas ha comido la serpiente.

→**La línea 114** Actualiza el tablero mostrando la serpiente con su nueva cabeza y longitud.

→**La línea 116** Esta condición verifica si la posición de la cabeza de la serpiente (serp[0]) se encuentra en alguna posición de su propio cuerpo (serp[1:]). Si esto sucede, significa que la cabeza de la serpiente chocó contra su propio cuerpo, lo que indica el final del juego. En este caso, se ejecutan las siguientes acciones:

→**La línea 117** Imprime el mensaje "¡Perdido!" en la consola para indicar que el jugador ha perdido.

→**La línea 118** Llama a la función savepoints() para guardar la puntuación del juego.

→**La línea 119** Rompe el if, finalizando la ejecución del juego.

→**La línea 121** Almacena la dirección actual (dir) en la variable dirant, para que en la siguiente iteración del bucle se tenga en cuenta la dirección anterior en caso de necesitar ajustar para evitar movimientos opuestos.

Ranking:

```
1    import pygame, sys
2    from pygame.locals import *
3    from random import randint
4    from juego import abrir_juego
```

→**La línea 1** Importa los módulos pygame y sys. pygame es una biblioteca que proporciona funcionalidades para el desarrollo de videojuegos en Python, mientras que sys proporciona funciones y variables relacionadas con la interacción con el sistema operativo.
→**La línea 2** Importa la función randint del módulo random. Esta función se utiliza para generar números enteros aleatorios.
→**La línea 3** Importa la función randint del módulo random. Esta función se utiliza para generar números enteros aleatorios
→**La línea 4** Importa la función abrir_juego desde el módulo juego. Esta línea asume que existe un archivo llamado "juego.py" en el mismo directorio que contiene la definición de la función abrir_juego. La función abrir_juego se ejecutará más adelante en el código.

```
6   def abrir_ranking():
7       # INICIACION DE PYGAME
8       pygame.init()
9
10      # PANTALLA VENTANA
11      W,H = 1200,900
12      menu = pygame.display.set_mode((W,H))
13
14      # VELOCIDAD POR SEGUNDO A CUANTO SE MUEVE ALGO
15      fps = 600
16      reloj = pygame.time.Clock()
17
18      # FUENTE DE LOS TEXTOS
19      fuente = pygame.font.Font(None, 20)
20
21      # FONDO DEL JUEGO
22      fondo = pygame.image.load("images/fondomenu.png")
23      # Variable de el fondo, x empieza en 0
24      x=0
25
26      # IMAGENES MENU Y SUS COORDS
27      score = pygame.image.load("images/score.png")
28      coordscore = (350, 50)
29      quit = pygame.image.load("images/quit.png")
30      coordsquit = (30, 700)
31
32      # ICONO Y TITULO
33      pygame.display.set_caption("SnakeThon")
34      icon = pygame.image.load('images/logo.png').convert()
35      pygame.display.set_icon(icon)
36
37      # TEXTO DE LA PANTALLA
38      text = "snapshot V1.1"
39      mensaje = fuente.render(text, 10, (255,255,255))
40      coordstextoversion = (1100, 880)
41
42      # EL MARCO QUE SI PULSAS DENTRO, PASA LO QUE DICE EN EL BUCLE
43      #                   posicion   ,  tamaño
44      botonquit = pygame.Rect(700,800,300, 85)
```

→**La línea 6** Define una ventana de visualización para mostrar un ranking de puntuaciones en un juego.

→**La línea 8-12** Inicia Pygame y establece las dimensiones de la ventana (W y H se establecen en 1200 y 900 respectivamente).

→**La línea 15-16** Establece la velocidad de actualización de la pantalla (fps) y crea un reloj para controlar la velocidad de fotogramas.
→**La línea 19** Carga una fuente de texto (fuente) con un tamaño de 20.
→**La línea 22** Carga una imagen de fondo (fondo) que se utilizará como fondo de la ventana.
→**La línea 27-30** Carga una imagen de fondo (fondo) que se utilizará como fondo de la ventana.
→**La línea 33-35** Establece el título de la ventana como "SnakeThon" y carga un icono (icon) para la ventana.
→**La línea 38-40** Crea un mensaje de texto para mostrar la versión del juego en la pantalla. El texto se renderiza utilizando la fuente y se establece en color blanco.
→**La línea 44** Crea un rectángulo interactivo (botonquit) que representa el botón de salida en la ventana. Si se hace clic dentro de este rectángulo, se realizan acciones específicas en el bucle del juego.

```
50          while True:
51              for evento in pygame.event.get():
52                  if evento.type == pygame.QUIT:
53                      pygame.quit()
54                      quit()
55
56                  # Verifica si se hizo clic en el botón
57                  if evento.type == pygame.MOUSEBUTTONDOWN:
58                      if botonquit.collidepoint(evento.pos):
59                          pygame.quit()
60                          quit()
```

→**La línea 50** El bucle while True permite que este segmento de código se ejecute continuamente
→**La línea 51** Dentro del bucle, se recorren todos los eventos generados por Pygame utilizando pygame.event.get(). Luego, se verifica el tipo de cada evento.
→**La línea 52-55** Si el evento es de tipo pygame.QUIT, significa que el usuario ha solicitado cerrar la ventana del juego. En este caso, se llama a pygame.quit() para cerrar Pygame y luego se llama a quit() para finalizar el programa.
→**La línea 57-60** El evento es pygame.MOUSEBUTTONDOWN, es que ha producido un clic del mouse. Se verifica si las coordenadas del clic (evento.pos) están dentro del rectángulo interactivo del botón de salida

(botonquit.collidepoint(evento.pos)). Si es así, se ejecuta el código para cerrar el juego de la misma manera que en el caso anterior.

```
62
63          #SE PUEDE DESGLOSaR PERO ES MEJOR ASI
64
65
66          # FONDO EN MOVIMIENTO
67          # SE DIVIDE EL VALOR DE X POR  EL ANCHO DE LA IMAGEN DEL FONDO Y DEVUELVE EL RESTO
68          x_bailonga = x % fondo.get_rect().width
69          menu.blit(fondo,(x_bailonga - fondo.get_rect().width,0))
70          if x_bailonga < W:
71              menu.blit(fondo,(x_bailonga,0))
72          x -= 1
73
74          # DIBUJA LAS IMAGENES
75          menu.blit(score, coordscore)
76          menu.blit(mensaje, coordstextoversion)
77          menu.blit(quit, coordsquit)
78
79          pygame.display.update()
80          reloj.tick(fps)
```

→**La línea 68-72** La variable x se utiliza para controlar el desplazamiento del fondo. Se calcula el valor de x_bailonga dividiendo x por el ancho de la imagen del fondo y obteniendo el resto. Esto permite crear un efecto de desplazamiento continuo del fondo. Luego se utiliza la función blit() para dibujar el fondo en la posición correspondiente.

→**La línea 75-77** A continuación, se utilizan las funciones blit() para dibujar las imágenes del puntaje (score), el mensaje de versión (mensaje), y el botón de salida (quit) en sus respectivas posiciones en la ventana del juego.

→**La línea 79** Después de dibujar todos los elementos en la pantalla, se llama a pygame.display.update() para actualizar la pantalla y mostrar los cambios.

→**La línea 80** Finalmente, se utiliza reloj.tick(fps) para controlar la velocidad de fotogramas del juego, limitando la cantidad de actualizaciones por segundo según el valor de fps.

Menú:

```
1   import pygame, sys
2   from pygame.locals import *
3   from random import randint
4   from juego import abrir_juego
5   from ranking import abrir_ranking
```

→**La línea 1** Importa el módulo principal de Pygame, que proporciona las funciones y clases necesarias para crear y gestionar juegos y importa el módulo sys de Python, que proporciona funciones y variables relacionadas con la interacción del sistema.
→**La línea 2** Importa todas las constantes y eventos definidos en el módulo pygame.locals. Estos son eventos específicos de Pygame, como QUIT, KEYDOWN, MOUSEBUTTONDOWN, etc.
→**La línea 3** Importa la función randint del módulo random. Esta función se utiliza para generar números enteros aleatorios.
→**La línea 4** Importa la función abrir_juego del módulo juego. Esto sugiere que hay un módulo llamado juego.py que contiene la definición de la función abrir_juego.
→**La línea 5** Importa la función abrir_ranking del módulo ranking. Esto sugiere que hay un módulo llamado ranking.py que contiene la definición de la función abrir_ranking.

```
7     # INICIACION DE PYGAME
8     pygame.init()
9
10    # PANTALLA VENTANA
11    W,H = 1200,900
12    menu = pygame.display.set_mode((W,H))
13
14    # VELOCIDAD POR SEGUNDO A CUANTO SE MUEVE ALGO
15    fps = 30
16    reloj = pygame.time.Clock()
17
18    # FUENTE DE LOS TEXTOS
19    fuente = pygame.font.Font(None, 20)
20
21    # FONDO DEL JUEGO
22    fondo = pygame.image.load("images/fondomenu2.png")
23    # Variable de el fondo, x empieza en 0
24    x=0
25
26    # IMAGENES MENU Y SUS COORDS
27    titulo = pygame.image.load("images/titulo.png")
28    coordstitulo = (10, -20)
29    play = pygame.image.load("images/play.png")
30    coordsplay = (450, 500)
31    rank = pygame.image.load("images/ranking.png")
32    coordsrank = (450, 610)
33    quit = pygame.image.load("images/quit.png")
34    coordsquit = (450, 720)
35
36    # ICONO Y TITULO
37    pygame.display.set_caption("SnakeThon")
38    icon = pygame.image.load('images/logo.png').convert()
39    pygame.display.set_icon(icon)
40
```

→**La línea 8** Inicializa el módulo Pygame para su uso. Esta función debe llamarse antes de utilizar cualquier otra función de Pygame.
→**La línea 11** Define las dimensiones de la ventana del menú principal. W representa el ancho de la ventana (1200 píxeles) y H representa la altura de la ventana (900 píxeles).
→**La línea 12** Crea una ventana del menú principal con las dimensiones especificadas. La variable menú se utiliza para acceder y manipular la ventana del menú.
→**La línea 15** Establece la velocidad de actualización de la ventana en 30 fotogramas por segundo. Esto determina la suavidad del movimiento y la actualización de la pantalla.
→**La línea 16** Crea un objeto de reloj que se utiliza para controlar la velocidad de actualización de la ventana. El objeto reloj se utiliza para limitar la velocidad de cuadros (FPS) del juego.
→**La línea 19** Crea un objeto de fuente con un tamaño de 20 píxeles. Esta fuente se utilizará para renderizar y mostrar texto en la ventana del menú.
→**La línea 22** Carga una imagen de fondo del menú principal desde el archivo "fondomenu2.png". La imagen se almacenará en la variable fondo.
→**La línea 24** Inicializa la variable x con el valor 0. Esta variable se utiliza para controlar el desplazamiento del fondo en el eje horizontal.
→**La línea 27** Carga la imagen del título del juego desde el archivo "titulo.png". La imagen se almacenará en la variable titulo.
→**La línea 28** Establece las coordenadas de la posición del título en la ventana del menú principal. En este caso, el título se ubicará en las coordenadas (10, -20), lo que determinará su posición en la ventana.
→**La línea 29** Carga la imagen del botón de juego desde el archivo "play.png". La imagen se almacenará en la variable play.
→**La línea 30** Establece las coordenadas de la posición del botón de juego en la ventana del menú principal. En este caso, el botón de juego se ubicará en las coordenadas (450, 500).
→**La línea 31** Carga la imagen del botón de juego desde el archivo "ranking.png". La imagen se almacenará en la variable ranking.
→**La línea 32** Establece las coordenadas de la posición del botón de ranking en la ventana del menú principal. En este caso, el botón de ranking se ubicará en las coordenadas (450, 610).

→**La línea 33** Carga la imagen del botón de salida desde el archivo "quit.png". La imagen se almacenará en la variable quit.
→**La línea 34** Establece las coordenadas de la posición del botón de salida en la ventana del menú principal. En este caso, el botón de salida se ubicará en las coordenadas (450, 720).
→**La línea 37-39** Establece el título de la ventana del juego como "SnakeThon" y se carga y establece el icono de la ventana utilizando la imagen del logotipo del juego.

```
41      # TEXTO DE LA PANTALLA
42      text = "snapshot V1.1"
43      mensaje = fuente.render(text, 10, (255,255,255))
44      coordstextoversion = (1100, 880)
45
46      # EL MARCO QUE SI PULSAS DENTRO, PASA LO QUE DICE EN EL BUCLE
47      #                   posicion   ,  tamaño
48      botonquit = pygame.Rect(450,720,300, 85)
49      #                   posicion   ,  tamaño
50      botonplay = pygame.Rect(450,500,300, 85)
51      #                   posicion   ,  tamaño
52      botonranking = pygame.Rect(450, 610,300, 85)
```

→**La línea 42** Se define el texto que se mostrará en la pantalla. En este caso, el texto es "snapshot V1.1".
→**La línea 43** Crea un objeto de texto renderizado utilizando la fuente especificada anteriormente
→**La línea 44** Establece las coordenadas de la posición del texto de la versión en la ventana del menú principal.
→**La línea 48** Crea un rectángulo que representa el marco del botón de salida en la ventana del menú principal. El rectángulo se crea utilizando las coordenadas de posición (450, 720)
→**La línea 50** Crea un rectángulo que representa el marco del botón de jugar en la ventana del menú principal. El rectángulo se crea utilizando las coordenadas de posición (450,500)
→**La línea 52** Crea un rectángulo que representa el marco del botón de ranking en la ventana del menú principal. El rectángulo se crea utilizando las coordenadas de posición (450, 610)

```
61      while True:
62          for evento in pygame.event.get():
63              if evento.type == pygame.QUIT:
64                  pygame.quit()
65                  quit()
66
67              # Verifica si se hizo clic en el botón
68              if evento.type == pygame.MOUSEBUTTONDOWN:
69                  if botonquit.collidepoint(evento.pos):
70                      pygame.quit()
71                      quit()
72              # Verifica si se hizo clic en el botón
73              if evento.type == pygame.MOUSEBUTTONDOWN:
74                  if botonplay.collidepoint(evento.pos):
75                      abrir_juego()
76              if evento.type == pygame.MOUSEBUTTONDOWN:
77                  if botonranking.collidepoint(evento.pos):
78                      abrir_ranking()
79          #SE PUEDE DESGLOSaR PERO ES MEJOR ASI
80
81
82          # FONDO EN MOVIMIENTO
83          # SE DIVIDE EL VALOR DE X POR  EL ANCHO DE LA IMAGEN DEL FONDO Y DEVUELVE EL RESTO
84          x_bailonga = x % fondo.get_rect().width
85          menu.blit(fondo,(x_bailonga - fondo.get_rect().width,0))
86          if x_bailonga < W:
87              menu.blit(fondo,(x_bailonga,0))
88          x -= 1
```

→**La línea 61** Es un bucle que se ejecuta continuamente y maneja los eventos generados por el usuario en la ventana del menú principal del juego.

→**La línea 62** Repite sobre todos los eventos generados por el usuario en la ventana del juego.

→**La línea 63-65** Verifica si el evento es del tipo "QUIT", lo que indica que el usuario ha solicitado cerrar la ventana del juego. En ese caso, se llama a pygame.quit() para cerrar la ventana de pygame y luego se llama a quit() para salir del programa.

→**La línea 68** Verifica si el evento es del tipo "MOUSEBUTTONDOWN", lo que indica que se ha realizado un clic del mouse.

→**La línea 69-71** Verifica si el botón de salida (botonquit) ha sido presionado. Compara la posición del evento (evento.pos) con el área del botón utilizando el método collidepoint(). Si el botón ha sido presionado, se llama a pygame.quit() para cerrar la ventana de pygame y luego se llama a quit() para salir del programa.

→**La línea 73** Verifica si el evento es del tipo "MOUSEBUTTONDOWN", lo que indica que se ha realizado un clic del mouse.
→**La línea 74-75** Verifica si el botón de juego (botonplay) ha sido presionado. Compara la posición del evento (evento.pos) con el área del botón utilizando el método collidepoint(). Si el botón ha sido presionado, se llama a la función abrir_juego() para iniciar el juego.
→**La línea 76** Verifica si el evento es del tipo "MOUSEBUTTONDOWN", lo que indica que se ha realizado un clic del mouse.
→**La línea 77-78** Verifica si el botón de ranking (botonranking) ha sido presionado. Compara la posición del evento (evento.pos) con el área del botón utilizando el método collidepoint(). Si el botón ha sido presionado, se llama a la función abrir_ranking() para abrir el ranking del juego.

```
82          # FONDO EN MOVIMIENTO
83          # SE DIVIDE EL VALOR DE X POR  EL ANCHO DE LA IMAGEN DEL FONDO Y DEVUELVE EL RESTO
84          x_bailonga = x % fondo.get_rect().width
85          menu.blit(fondo,(x_bailonga - fondo.get_rect().width,0))
86          if x_bailonga < W:
87              menu.blit(fondo,(x_bailonga,0))
88          x -= 1
```

→**La línea 84** Calcula el valor de x_bailonga como el resto de dividir x por el ancho de la imagen del fondo (fondo.get_rect().width). Esto permite crear un efecto de desplazamiento continuo del fondo.
→**La línea 85** Dibuja el fondo en la posición (x_bailonga-fondo.get_rect().width, 0) en la ventana del menú. Esta línea se utiliza para mostrar una copia del fondo desplazado hacia la izquierda.
→**La línea 86-87** Verifica si el valor de x_bailonga es menor que el ancho de la ventana (W). Si es así, se dibuja otra copia del fondo en la posición (x_bailonga, 0). Esto se hace para asegurar que el fondo se muestre de manera continua mientras se desplaza.
→**La línea 88** Decrementa el valor de x en 1 en cada iteración del bucle. Esto hace que el fondo se desplace hacia la izquierda en cada fotograma, ya que las coordenadas utilizadas para dibujar el fondo se actualizan en función de x.

```
90          # DIBUJA LAS IMAGENES
91          menu.blit(titulo,coordstitulo)
92          menu.blit(play,coordsplay)
93          menu.blit(rank,coordsrank)
94          menu.blit(mensaje, coordstextoversion)
95          menu.blit(quit, coordsquit)
96
97          pygame.display.update()
98          reloj.tick(fps)
```

→**La línea 91** Dibuja la imagen del título en la posición especificada por las coordenadas coordstitulo.
→**La línea 92** Dibuja la imagen del botón "Play" en la posición especificada por las coordenadas coordsplay.
→**La línea 93** Dibuja la imagen del botón "Ranking" en la posición especificada por las coordenadas coordsrank.
→**La línea 94** Dibuja el mensaje de la versión en la posición especificada por las coordenadas coordstextoversion.
→**La línea 95** Dibuja la imagen del botón "Quit" en la posición especificada por las coordenadas coordsquit.
→**La línea 97-98** Después de dibujar todas las imágenes en la ventana, se llama a pygame.display.update() para actualizar la pantalla y mostrar los cambios realizados. Luego, se utiliza reloj.tick(fps) para controlar la velocidad de fotogramas del menú principal según el valor de fps.

ALBERT ASX PROJECT 1/2.mp3:

→Estos ficheros de audio los juntaré en la misma explicación porque están relacionados. Queríamos hacer una banda sonora para nuestro juego, le pedimos al profesor y al centro si nos dejaban utilizar el aula del segundo piso, cuando teníamos todos los permisos el Albert y Daniel estuvieron haciendo la música con un chico muy majo que los ayudó. Queríamos hacer una canción similar a la del juego de donkey kong pero el único integrante que tenía conocimiento sobre música era el Albert que toca muy bien el piano no pudimos hacer lo que queríamos, e hicimos la música actual. Estamos orgullosos del resultado aunque no fuese nuestra idea principal y consideramos que queda bien con el ambiente del juego.

Carpetas:

→Como en el apartado anterior, juntaré el apartado de carpetas porque tienen la misma utilidad pero con fines distintos.

La gran mayoría de imágenes están hechas con un programa web llamado Piskel, ha sido nuestro salvador para crear la serpiente y los fondos.

→En la carpeta images tenemos bastantes archivos, por ejemplo

Tenemos imágenes que utilizamos antes y hemos cambiado por unas mejores, tenemos los botones, bordes,logo, en resumen, tenemos todo el apartado gráfico en la carpeta.

Esta carpeta considero que es la más importante, es dónde esta el cuerpo de la serpiente, son muchas imágenes distintas, como es una serpiente si mira para arriba es una imagen distinta que si mira hacia la derecha y cuando gira tiene una imagen según hacia donde se voltee.

→La carpeta songs no es tan extensa, son los audios que sonarán durante el juego.