Fet per: Juan Camilo De Los Ríos Algorismia i Programacio Audiovisual (APA)
Data: 1-06-2023

SNAKE GAME ESEIAAT / HARDCORE SNAKE GAME

Objectiu:

L'objectiu d'aquest projecte, ha sigut saber utilitzar les llibreries "turtle", "time" i "random" per a, de la manera menys complicada, aprendre a programar el típic joc de l'snake. He volgut presentar dos tipus de joc, ja que un és el convencional joc de l'snake, i l'altre li he afegit obstacles al joc que faran que la dificultat del joc augmenti i el jugador perdi més ràpid, fent el joc algo més interessant.

Disseny i desenvolupament de l'aplicació:

El disseny no és el més modern possible , ja que no s'han utilitzat les llibreries més adequades per a jocs (animacions 3D, com "pygame", per exemple).

Comentaré el primer exemple de joc i després el que se li ha modificat, per obtenir el segon exemple "HARDCORE SNAKE GAME".

Primer de tot he importat les tres llibreries esmentades ("turtle", "time" i "random"). Després he creat les variables per els "delays" necessaris del joc i els resultats inicialitzats a zero.

A continuació venen les configuracions dels elements de la llibreria "turtle" necessaris per a la visualització del joc com la finestra de joc, el cap de la serp, el menjar de la serp, el cos de la serp, el resultat del marcador i el text de game over, amb les seves propietats corresponents ("títol", "bgcolor", dimensions, etc.).

En la segona versió del joc (Hardcore mode), he creat un objecte de tipus "turtle" que contindrà les propietats del obstacle que apareixerà aleatòriament a la finestra cada cop que la serp vagi acumulant punts i que farà perdre més fácilment al jugador, ja que si el cap el toca, la serp mor i es "GAME OVER". He creat també una llista de tots els objectes acumulats.

Després de fer el "setup" dels elements importants del joc, venen les funcions que fan que aquests elements interactuïn entre si donant consistència a les accions del jugador al jugar. Les funcions primeres son les que s'utilitzen per a canviar la direcció del cap de la serp (def adalt(), def abaix(), def esquerre(), def dreta()), a continuació, la funció de moviment (def moviment()), on s'especifica els píxels que volem que el cap es mogui dins les coordenades del marc de la finestra. Per últim la configuració del teclat, conforme s'utilitza la funció onkeypress() amb paràmetres de direcció i moviment creats anteriorment. Cal remarcar que el segon paràmetre s'ha d'escriure en majúscules per fer referència a una tecla del teclat.

Per últim es troba el bucle principal del joc (While:), on primer de tot actualitzarem la pantalla constantment i a partir d'aquí construirem iteradors que donin pas a les característiques principals del joc. Aquests són:

- Col·lisió del cap amb el marc de la finestra: donades les especificacions correctes, el jugador perdrà la partida quan el cap colisioni amb qualsevol coordenada del marc de la finestra, i tot es reinicia altre cop per tornar a començar una partida nova. Els segments aconseguits del cos de la serp es perden, el resultat torna a zero, mantenint el millor resultat i apareixent el títol de "GAME OVER".

- Col·lisió entre el cap de la serp y el menjar: aquí he fet que al xocar el cap de la serp amb el punt vermell que apareix (menjar), el menjar vagi reapareixent aleatòriament per la finestra de joc i apareixi un nou segment de cos que s'acumuli a la llista del cos creada anteriorment, que el resultat augmenti 10 punts i que s'actualitzi si és necessari el millor resultat. En la segona versió del joc, he fet que cada cop que la serp menji, aparegui un nou obstacle de forma aleatoria dins del marc de joc, per així augmentar el nivell de dificultat del joc.

- Moviment del cos de la serp: per moure el cos de la serp, eh iterat els components de la llista. Per donar una sensació d'animació, utilitzem bucle for, però encara sense teledirigirlos cap al cap de la serp. Aquest bucle es per que els segments del cos de la serp es segueixin entre ells un cop s'han afegit al cap de la serp. Després s'obté les coordenades x i y del segment anterior perquè l'últim element es mogui o segueixi al anterior i amb la instrucció "segments_cos[index].goto(x,y)" mourem l'índex actual cap a les coordenades de l'element anterior i per tant seguirlo. A continuació amb el bucle if, es de teledirigeix els segments del cos cap al cap de la serp. Finalment crido la funció de moviment (moviment()) creada anteriorment.

- Col·lisió entre el cap i el mateix cos de la serp: aquí iterem un altre bucle for per cada element de la llista, per fer que quan es toqui el cap amb el cos (distància més petita que els 20 pixels de dimensió per defecte del "square shape" (cap i cos)), el cap es pari, els segments es perdin, el resultat torni a baixar a zero mantenint el millor resultat i surti per pantalla el "GAME OVER".

- La funció sleep de la allibreria "time", que utilitza com a paràmetre "posposar" creat al principi, s'utilitza per a que el programa no s'executi tan rápid.

- Col·lisió entre el cap i l'obstacle: en la segona versió del joc he hagut de crear també la part en que el cap col·lisiona amb l'obstacle. He intentat que al tocar el cap de la serp l'objecte, tots els objectes tornessin a desaparèixer, però no ho he aconseguit

malauradament. El resutat torna a zero, sense canviar el marcador de millor resultat, i apareix el “GAME OVER” per pantalla.

Prestacions:

En la versió primera del joc serien les mateixes que la del joc convencional de l’snake, i en la segona versió he afegit més dificultat al joc creant enemics o obstacles cada cop que la serp menja i es fa més gran.

Limitacions:

Les principals limitacions serien que no he pogut fer desaparèixer els obstacles creats al llarg de la partida, al col·lisionar amb el cap. També no he sapigut resoldre del tot el tema del GAME OVER amb el “finestra.tracer(0)”, ja que si s’escriu el comand, el text de “GAME OVER” no surt per pantalla i es queda al background de la finestra. El codi s'hagués pogut optimitzar, creant funcions per no escriure tant de codi.

CONCLUSIONS:

He sapigut portar a terme els coneixements bàsics de programació apresos a classe, utilitzant de manera no tan complicada tres llibreries i poder crear un joc simple però que compila bé i pot satisfer la necessitat del jugador de jugar al mític joc de l’snake i a sobre amb un grau més alt de dificultat.

Annex 1: com executar l'aplicació

Després de guardar el projecte, presionar Cntrl + B i començar a jugar o simplement compilar el codi per a començar a gaudir del joc.

Annex 2: codi de l’aplicació

```
# Joc de l'Snake fet a Python
# Fet per: Juan Camilo De Los Ríos

import turtle       # S'utilitza per a desenvolupar conceptes d'una manera més entretinguda
import time         # L'utilitzarem per a modificar els temps d'animació del joc
import random       # S'utilitza per a la reaparició aleatoria del menjar cada cop que es menjat per la serp
```

```
posposar = 0.11         # Per a retrasar 0.11 milèsimes de segon l'execució del programa
resultat = 0            # Variable que contindrà el resultat (score) actual
millor_resultat = 0     # Variable que contindrà el major resultat aconsguit fins el moment


# CONFIGURACIÓ DE LA FINESTRA DE JOC

finestra = turtle.Screen()                              # creem la finestra (element Screen()) on es desenvolupará el nostre joc (titol del joc, background, colors, etc.
finestra.title("SNAKE GAME ESEIAAT")            # li donem un títol al joc
finestra.bgcolor('blue')                        # canviem el color de fons de la finestra
finestra.setup(width = 800 , height = 800)      # redimensionem la finestra amb les mesures desitjades
#finestra.tracer(0)                             # fará que les animacions siguin una mica més agradables als nostres ulls i desactiva les actualitzacions de la pantalla



# CREACIÓ DEL CAP DE LA SERP

cap = turtle.Turtle()           # creem un objecte turtle perque es mostri algo a la pantalla
cap.speed(0)                    # per a que s'inicii la pantalla, el cap de la serp estigui des de l'inici
cap.shape("square")             # canviem la forma del cap a quadrada
cap.color("yellow")             # li donem un color al cap de la serp
cap.penup()                     # amb aquest comand, tot i que el cap de la serp es mogui, no hi haurá rastre o estela
cap.goto(0,0)                   # començará a la posició (0,0) de la pantalla
cap.direction = "stop"          # amb això direccionem el cap de la serp en la direcció que marquem amb les fletxes del teclat.

 # Amb l'stop, li diem que no volem que es mogui en cap direcció fins que es faci click a alguna direcció



# MENJAR PER A LA SERP

food = turtle.Turtle()          # creem un objecte turtle perque es mostri algo a la pantalla, en aquest cas, el menjar de la serp per a que creixi
food.speed(0)                   # per a que s'inicii la pantalla, el menjar estigui des de l'inici
food.shape("circle")            # canviem la forma del menjar a circular
food.color("red")               # li donem un color, en aquest cas vermell
```

```
food.penup()                # amb aquest comand, tot i que el food de la serp es mogui, no hi haurá rastre o estela
food.goto(0,111)            # començará a la posició (0,111) de la pantalla



# EL COS DE LA SERP:

# El cos de la serp, simplement són segments. Quan es toqui ("menji") amb el cap de la serp el menjar, s'anirà afegint un segment.
# L'estructura de dades més conevnient per realitzar aquesta acció, seria una LLISTA.


segments_cos = []  # es crea la llista que contindrá els segments de cos de la serp



# RESULTAT DEL MARCADOR

text = turtle.Turtle()  # Necessitarem un text que sigui igual a un obecte "turtle.Turtle()"
text.speed(0)           # El text no es mourà ja que romandrà quiet a una zona desitjada
text.color("yellow")    # Escollim el color desitjat
text.penup()            # Sense deixar rastre o estela
text.hideturtle()       # Fa invisible a la fletxeta
text.goto(0,300)        # Posicionem el marcador en l'eix y positiu


text.write(f" RESULTAT: 0                    MILLOR RESULTAT: 0 " ,          #  Amb el mètode .write(), fem un fstream dels resultats amb alineació centrada i escollim el tipus de font,
           align = "center" , font = ("Calibri", 24, "bold"))               #  que en aquest cas es una tupla (tipus, tamany, no negreta)



# TEXT DE GAME OVER

game_over = turtle.Turtle()     # Necessitarem un text que sigui igual a un obecte "turtle.Turtle()". Creem el text que ens informarà del GAME OVER al perdre
game_over.speed(0)              # El text no es mourà ja que romandrà quiet a una zona desitjada
game_over.color("red")          # Escollim el color desitjat
game_over.penup()               # Sense deixar rastre o estela
game_over.hideturtle()          # Fa invisible a la fletxeta
```

game_over.goto(0,111) # Posicionem el marcador en l'eix y positiu, una mica més adalt de la meitat

#game_over.write(f" GAME OVER " , align = "center" , font = ("Impact", 55, "bold")) --- (Aquesta linia l'he ficat per si no es veia el GAME OVER al morir, comprovar que s'escriu bé pepr pantalla)

FUNCIONS PER A CAMBIAR LA DIRECCIÓ DEL CAP DE LA SERP

def adalt(): # Funció per a que la direcció de la funció vagi cap adalt
  cap.direction = "up"

def abaix(): # Funció per a que la direcció de la funció vagi cap abaix
  cap.direction = "down"

def esquerre(): # Funció per a que la direcció de la funció vagi cap a l'esquerre
  cap.direction = "left"

def dreta(): # Funció per a que la direcció de la funció vagi cap a la dreta
  cap.direction = "right"

FUNCIONS PER EL MOVIMENT DE LA SERP

def moviment():

  if cap.direction == "down": # si la direcció del cap és "down" ("abaix"), en comptes de "stop", es direccionará cap abaix.
    y = cap.ycor() # per a que s'envagi cap abaix, hem de modificar el nostre eix Y i que el cap baixi. Obtenim així la coordenda Y del cap de la serp i la guardem a la variable y.
    cap.sety(y - 20) # aquí fem un set de la cordenada en que es mogui 18 píxels cap abaix cada cop que s'activi la funció.

```
    if cap.direction == "up":       # si la direcció del cap és "up" ("adalt"), en comptes de "stop", es direccionará cap adalt.



y = cap.ycor()                  # per a que s'envagi cap adalt, hem de modificar el nostre eix Y i que el cap pugi. Obtenim així la coordenda Y del cap de la serp i la guardem a la variable y.
        cap.sety(y + 20)            # aquí fem un set de la cordenada en que es mogui 18 píxels cap adalt cada cop que s'activi la funció.

    if cap.direction == "right":    # si la direcció del cap és "right" ("dreta"), en comptes de "stop", es direccionará cap a la dreta e n l'eix X.
        x = cap.xcor()              # per a que s'envagi cap a ladreta, hem de modificar el nostre eix X i que el cap giri. Obtenim així la coordenda X del cap de la serp i la guardem a la variable x.
        cap.setx(x + 20)            # aquí fem un set de la cordenada en que es mogui 18 píxels cap a la dreta cada cop que s'activi la funció.

    if cap.direction == "left":     # si la direcció del cap és "left" ("esquerre"), en comptes de "stop", es direccionará cap a l'esquerra en l'eix X.
        x = cap.xcor()              # per a que s'envagi cap a l'esquerre, hem de modificar el nostre eix X (absices de la pantalla) i que el cap giri. Obtenim així la coordenda X del cap de la serp i la guardem a la variable x.
        cap.setx(x - 20)            # aquí fem un set de la cordenada en que es mogui 18 píxels cap a l'esquerre cada cop que s'activi la funció.



# CONFIGURACIÓ DEL TECLAT

finestra.listen()                       # li diem a la pantalla que estigui atenta i escoltant les ordres del teclat
finestra.onkeypress(adalt, "Up")        # si es prem una tecla del teclat, els parámetres que li pasaré a la funció son als que el cap de la serp reaccionará.
finestra.onkeypress(abaix, "Down")      # La primera lletra de la vocal del segon parámetre, ha de ser majúscula per fer referència a una tecla del teclat.
finestra.onkeypress(esquerre, "Left")
finestra.onkeypress(dreta, "Right")
```

Fet per: Juan Camilo De Los Ríos Algorismia i Programacio Audiovisual (APA)
Data: 1-06-2023

FUNCIÓ BUCLE WHILE PRINCIPAL

```
while True:                    # creem un bucle principal, ja que será uhn bucle infinit que fins que no li donem ordre de que surti del joc, mai acabará.
    finestra.update()          # actualitzarem la pantalla constantment, conforme es faci "run" del bucle s'anirá actualizant
```

COL·LISIÓ AMB EL MARC DE LA FINESTRA (GAME OVER)

```
    if (cap.xcor() > 390 or cap.xcor() < -390 or cap.ycor() > 390 or cap.ycor() < -390):    # Mentre que les coordenades del cap (20 pixels) sobrepassin els límits de la coordenada x i y, negatius i positius (-390,390, més el cap de la serp (20))
        time.sleep(1)                                                                       # s'adorm el programa un segon
         cap.goto(0,0)                                                                      # reset al programa fent qque el cap torni a la posició d'origen (0,0)
         cap.direction = "stop"                                                             # i la direcció estigui en mode "stop"

        # AMAGUEM ELS SEGMENTS OBTINGUTS, AL MORIR
            for segment in segments_cos:            # Per cada segment que estigui a la llista "segments_cos"
                segment.goto(1111,1111)     # els fem anar ben lluny (fora dels marcs de la finiestra (1111,1111))

         segments_cos.clear()               # Borrem els segments de la llista, però fora del bucle for, ja que sinó els segments quedarien dins de la finestra

        # Reset del resultat del marcador
        resultat = 0

         # Actualitzem l'fstream amb els resultats, natejant el text (.clear()) perque no es solapin els increments del resultat
        text.clear()
          text.write(f" RESULTAT: {resultat}                    MILLOR RESULTAT: {millor_resultat} " , align = "center" , font = ("Calibri", 24, "bold"))
```

```
# Aquí escribim amb un fstream, el famòs "GAME OVER" per indicar que s'ha perdut la partida

        game_over.write(f" GAME OVER " , align = "center" , font = ("Impact", 55, "bold"))
        time.sleep(1)
        game_over.clear()



# COL·LISIÓ ENTRE EL CAP Y EL MENJAR DE LA SERP

    # aquí mirem la distanca entre els dos objectes, que serien el cap de la serp i el menjar
    # li donem 20 unitats, ja que les mesures del quadrat i cercle per defecte son 20 x 20 pixels



 if cap.distance(food) < 20:                # si la distanca es menor al tamany dels dos objectes, significará que s'han tocat
         x = random.randint(-380, 380)       # creem números random per assignarlos a la nostra X i Y
        y = random.randint(-380, 380)       # creem un núm. enter amb possibilitat de que surti dels -380 fins als 380 per a x i y, per no estar tan aprop del marge de la finestra
        food.goto(x,y)                      # per actualitzar la possició del menjar


         nou_segment = turtle.Turtle()           # creem un objecte turtle perque es mostri algo a la pantalla
         nou_segment.speed(0)                  # per a que s'inicii la pantalla, el nou segment del cos de la serp estigui des de l'inici
        nou_segment.shape("square")           # la forma del cos també serà quadrada
        nou_segment.color("orange")           # li donem un color al cos de la serp
         nou_segment.penup()                  # amb aquest comand, tot i que el nou_segment de la serp es mogui, no hi haurá rastre o estela
           segments_cos.append(nou_segment)         # cada cop que es crei el segment de cos, s'anirà afegint a la llista "segments_cos"


        # Escurçament del retard

        posposar = posposar - 0.001
```

Fet per: Juan Camilo De Los Ríos Algorismia i Programacio Audiovisual (APA)
Data: 1-06-2023

```
 # AUGMENT DEL RESULTAT DEL MARCADOR

        resultat = resultat + 10              # Augmentem el resultat 10 punts, cada cop que la serp menji el menjar

       if resultat > millor_resultat:         # Si el resultat és major que el millor resultat
          millor_resultat = resultat          # El millor resultat s'actualitzarà

        # Actualitzem l'fstream amb els resultats, natejant el text (.clear()) perque no es solapin els increments del resultat
       text.clear()
         text.write(f" RESULTAT: {resultat}                    MILLOR RESULTAT: {millor_resultat} " , align = "center" , font = ("Calibri", 24, "bold"))



# MOVIMENT DEL COS DE LA SERP



# Per moure el cos de la serp, iterarem els components de la llista. El primer índex de la llista sempre es el 0. Els paràmetres
   # del bucle for seran: l'últim element (segments_totals -1), fins al primer que és el 0 que no es inclòs dins del bucle, i l'últim
  # paràmetre será que vagi decreixent el valor (-1).

     segments_totals = len(segments_cos)  # métode per obtindre de forma entera (int), els segments de cos totals

       # Per donar una sensació d'animació, iterem aquest bucle for, però encara sense teledirigirlos cap al cap de la serp.
     # Aquest bucle es per que els segments de cos de la serp es segueixin entre ells un cop s'han afegit al cap de la serp
   for index in range (segments_totals - 1, 0, -1):

         # otenim les coordenades x i y del segment anterior perque l'últim element es mogui o segueixi al anterior
       x = segments_cos[index - 1].xcor()
       y = segments_cos[index - 1].ycor()
        segments_cos[index].goto(x,y)       # amb aquesta instrucció mourem el index actual cap a les coordenades de l'element anterior i per tant seguirlo

   # Amb aquest bucle if, acabarem de teledirigir els segments del cos cap al cap de la serp
```

```
    if segments_totals >  0:        # Mirem que la llista no estigui buida, ja que si ho està, donaria error
        x = cap.xcor()            # Obtenim la coordenada x del cap de la serp
        y = cap.ycor()            # Obtenim la coordenada y del cap de la serp
        segments_cos[0].goto(x,y)   # Fem que es mogui cap a on està el cap de la serp



    moviment()      # iniciem la funció de moviment


    # COL·LISIÓ ENTRE EL CAP Y EL MATEIX COS DE LA SERP (SEGMENTS DEL COS)

    # Iterem cada element de la llista "segments_cos"
    for segment in segments_cos:
        if segment.distance(cap) < 20:      # Fem que la distancia amb el segments del cos no estigui tan aprop del cap de la serp
            time.sleep(1)               # Una petita pausa d'1 seg per la colisió
            cap.goto(0,0)               # Fem que el cap de la serp torni al seu lloc d'origen (0,0)
            cap.direction = "stop"        # Aturem la direcció del cap


            # TORNEM A AMAGAR ELS SEGMENTS OBTINGUTS, AL MORIR

                for segment in segments_cos:          # Per cada segment que estigui a la llista "segments_cos"
                    segment.goto(1111,1111)     # els fem anar ben lluny (fora dels marcs de la finiestra (1111,1111))

                segments_cos.clear()              # Borrem els segments de la llista, però fora del bucle for, ja que sinó els segments quedarien dins de la finestra


            # Reset del resultat del marcador
            resultat = 0

             # Actualitzem l'fstream amb els resultats, natejant el text (.clear()) perque no es solapin els increments del resultat
            text.clear()
            text.write(f" RESULTAT: {resultat}                    MILLOR RESULTAT: {millor_resultat} " , align = "center" , font = ("Calibri", 24, "bold"))
```

```
        game_over.write(f"GAME OVER " , align = "center" , font = ("Impact", 55, "bold"))
        time.sleep(2)
        game_over.clear()


    time.sleep(posposar)        # per a que el programa no s'executi tan rápid
```

Codi de Hardcore Snake Game:

```
# Joc de l'Snake fet a Python, per princiants, amb obstàcles (HARDCORE MODE)
# Fet per: Juan Camilo De Los Ríos

import turtle       # S'utilitza per a desenvolupar conceptes d'una manera més entretinguda
import time         # L'utilitzarem per a modificar els temps d'animació del joc
import random       # S'utilitza per a la reaparició aleatoria del menjar cada cop que es menjat per la serp


posposar = 0.11         # Per a retrasar 0.1 milèsimes de segon l'execució del programa
resultat = 0            # Variable que contindrà el resultat (score) actual
millor_resultat = 0     # Variable que contindrà el major resultat aconsguit fins el moment


# CONFIGURACIÓ DE LA FINESTRA DE JOC

finestra = turtle.Screen()                              # creem la finestra (element Screen()) on es desenvolupará el nostre joc (titol del joc, background, colors, etc.
finestra.title("SNAKE GAME ESEIAAT")            # li donem un títol al joc
finestra.bgcolor("blue")                        # canviem el color de fons de la finestra
finestra.setup(width = 800 , height = 800)      # redimensionem la finestra amb les mesures desitjades
#finestra.tracer(0)                             # fará que les animacions siguin una mica més agradables als nostres ulls i desactiva les actualitzacions de la pantalla


# CREACIÓ DEL CAP DE LA SERP

cap = turtle.Turtle()           # creem un objecte turtle perque es mostri algo a la pantalla
cap.speed(0)                    # per a que s'inicii la pantalla, el cap de la serp estigui des de l'inici
```

```
cap.shape("square")          # canviem la forma del cap a quadrada
cap.color("yellow")          # li donem un color al cap de la serp
cap.penup()                  # amb aquest comand, tot i que el cap de la serp es mogui, no hi haurá rastre o estela
cap.goto(0,0)                # començará a la posició (0,0) de la pantalla
cap.direction = "stop"       # amb això direccionem el cap de la serp en la direcció que marquem amb les fletxes del teclat.
                             # Amb l'stop, li diem que no volem que es mogui en cap direcció fins que es faci click a alguna direcció



# CREACIÓ MENJAR PER A LA SERP

food = turtle.Turtle()       # creem un objecte turtle perque es mostri algo a la pantalla, en aquest cas, el menjar de la serp per a que creixi
food.speed(0)                # per a que s'inicii la pantalla, el menjar estigui des de l'inici
food.shape("circle")         # canviem la forma del menjar a circular
food.color("red")            # li donem un color, en aquest cas vermell
food.penup()                 # amb aquest comand, tot i que el food de la serp es mogui, no hi haurá rastre o estela
food.goto(0,111)             # començará a la posició (0,111) de la pantalla



# CREACIÓ DELS OBSTÀCLES QUE SERAN ELS ENEMICS

obstacle = turtle.Turtle()   # creem un objecte turtle perque es mostri algo a la pantalla
obstacle.speed(0)            # per a que s'inicii la pantalla, el cap de la serp estigui des de l'inici
obstacle.shape("square")     # forma del obstacle a quadrada
obstacle.color("black")      # li donem un color al obstacle (negre)
obstacle.penup()             # amb aquest comand no hi haurá rastre o estela
obstacle.goto(0,150)         # començará a la posició (0,0) de la pantalla
obstacle.direction = "stop"  # amb això direccionem el primer obstacle a la posició desitjada.
                             # Amb l'stop, li diem que no volem que es mogui en cap direcció

obstacles = []               # Creem la llista per acumular obstacles
```

EL COS DE LA SERP:

El cos de la serp, simplement són segments. Quan es toqui ("menji") amb el cap de la serp el menjar, s'anirà afegint un segment.
L'estructura de dades més conevnient per realitzar aquesta acció, seria una LLISTA.

```
segments_cos = []  # es crea la llista que contindrá els segments de cos de la serp
```

RESULTAT DEL MARCADOR

```
text = turtle.Turtle()  # Necessitarem un text que sigui igual a un obecte "turtle.Turtle()"
text.speed(0)           # El text no es mourà ja que romandrà quiet a una zona desitjada
text.color("yellow")    # Escollim el color desitjat
text.penup()            # Sense deixar rastre o estela
text.hideturtle()       # Fa invisible a la fletxeta
text.goto(0,300)        # Posicionem el marcador en l'eix y positiu

text.write(f" RESULTAT: 0                    MILLOR RESULTAT: 0 " ,      #  Amb el mètode .write(), fem un fstream dels resultats amb alineació centrada i escollim el tipus de font,
          align = "center" , font = ("Calibri", 24, "bold"))              #  que en aquest cas es una tupla (tipus, tamany, no negreta)
```

TEXT DE GAME OVER

```
game_over = turtle.Turtle()     # Necessitarem un text que sigui igual a un obecte "turtle.Turtle()". Creem el text que ens informarà del GAME OVER al perdre
game_over.speed(0)              # El text no es mourà ja que romandrà quiet a una zona desitjada
game_over.color("red")          # Escollim el color desitjat
game_over.penup()               # Sense deixar rastre o estela
game_over.hideturtle()          # Fa invisible a la fletxeta
game_over.goto(0,111)           # Posicionem el marcador en l'eix y positiu, una mica més adalt de la meitat

#game_over.write(f" GAME OVER " , align = "center" , font = ("Impact", 55, "bold"))      #  que en aquest cas es una tupla (tipus, tamany, negreta)
```

Fet per: Juan Camilo De Los Ríos Algorismia i Programacio Audiovisual (APA)
Data: 1-06-2023

FUNCIONS PER A CAMBIAR LA DIRECCIÓ DEL CAP DE LA SERP

```
def adalt():                # Funció per a que la direcció de la funció vagi cap adalt
    cap.direction = "up"


def abaix():                # Funció per a que la direcció de la funció vagi cap abaix
    cap.direction = "down"


def esquerre():             # Funció per a que la direcció de la funció vagi cap a l'esquerre
    cap.direction = "left"


def dreta():                # Funció per a que la direcció de la funció vagi cap a la dreta
    cap.direction = "right"
```

FUNCIONS PER EL MOVIMENT DE LA SERP

```
def moviment():

    if cap.direction == "down":     # si la direcció del cap és "down" ("abaix"), en comptes de "stop", es direccionará cap abaix.
        y = cap.ycor()              # per a que s'envagi cap abaix, hem de modificar el nostre eix Y i que el cap baixi. Obtenim així la coordenda Y del cap de la serp i la guardem a la variable y.
        cap.sety(y - 20)            # aquí fem un set de la cordenada en que es mogui 18 píxels cap abaix cada cop que s'activi la funció.

    if cap.direction == "up":       # si la direcció del cap és "up" ("adalt"), en comptes de "stop", es direccionará cap adalt.
        y = cap.ycor()              # per a que s'envagi cap adalt, hem de modificar el nostre eix Y i que el cap pugi. Obtenim així la coordenda Y del cap de la serp i la guardem a la variable y.
        cap.sety(y + 20)            # aquí fem un set de la cordenada en que es mogui 18 píxels cap adalt cada cop que s'activi la funció.

    if cap.direction == "right":    # si la direcció del cap és "right" ("dreta"), en comptes de "stop", es direccionará cap a la dreta e n l'eix X.
        x = cap.xcor()              # per a que s'envagi cap a ladreta, hem de modificar el nostre eix X i que el cap giri. Obtenim així la coordenda X del cap de la serp i la guardem a la variable x.
```

```
        cap.setx(x + 20)            # aquí fem un set de la cordenada en que es mogui 18 píxels cap a la dreta cada cop que s'activi la funció.

    if cap.direction == "left":     # si la direcció del cap és "left" ("esquerre"), en comptes de "stop", es direccionará cap a l'esquerra en l'eix X.
        x = cap.xcor()              # per a que s'envagi cap a l'esquerre, hem de modificar el nostre eix X (absices de la pantalla) i que el cap giri. Obtenim així la coordenda X del cap de la serp i la guardem a la variable x.
        cap.setx(x - 20)            # aquí fem un set de la cordenada en que es mogui 18 píxels cap a l'esquerre cada cop que s'activi la funció.



# CONFIGURACIÓ DEL TECLAT

finestra.listen()                   # li diem a la pantalla que estigui atenta i escoltant les ordres del teclat
finestra.onkeypress(adalt, "Up")            # si es prem una tecla del teclat, els parámetres que li pasaré a la funció son als que el cap de la serp reaccionará.
finestra.onkeypress(abaix, "Down")      # La primera lletra de la vocal del segon parámetre, ha de ser majúscula per fer referéncia a una tecla del teclat.
finestra.onkeypress(esquerre, "Left")
finestra.onkeypress(dreta, "Right")



# FUNCIÓ BUCLE WHILE PRINCIPAL


while True:                     # creem un bucle principal, ja que será uhn bucle infinit que fins que no li donem ordre de que surti del joc, mai acabará.
    finestra.update()           # actualitzarem la pantalla constantment, conforme es faci "run" del bucle s'anirá actualizant


   # COLISIÓ AMB EL MARC DE LA FINESTRA (GAME OVER)

    if (cap.xcor() > 390 or cap.xcor() < -390 or cap.ycor() > 390 or cap.ycor() < -390):    # Mentre que les coordenades del cap (20 pixels) sobrepassin els límits de la coordenada x i y, negatius i positius (-390,390, més el cap de la serp (20))
        time.sleep(1)                                                   # s'adorm el programa un segon
         cap.goto(0,0)                                                  # reset al programa fent qque el cap torni a la posició d'origen (0,0)
```

```
 cap.direction = "stop"                                    # i la direcció estigui en mode "stop"


          # AMAGUEM ELS SEGMENTS OBTINGUTS, AL MORIR
             for segment in segments_cos:            # Per cada segment que estigui a la llista "segments_cos"
                  segment.goto(1111,1111)      # els fem anar ben lluny (fora dels marcs de la finiestra (1111,1111))

         segments_cos.clear()              # Borrem els segments de la llista, però fora del bucle for, ja que sinó els segments quedarien dins de la finestra

        # Reset del resultat del marcador
        resultat = 0


         # Actualitzem l'fstream amb els resultats, natejant el text (.clear()) perque no es solapin els increments del resultat

        text.clear()
          text.write(f" RESULTAT: {resultat}                    MILLOR RESULTAT: {millor_resultat} " , align = "center" , font = ("Calibri", 24, "bold"))


         # Aquí escribim amb un fstream, el famòs "GAME OVER" per indicar que s'ha perdut la partida

        game_over.write(f" GAME OVER " , align = "center" , font = ("Impact", 55, "bold"))
        time.sleep(2)
        game_over.clear()




    # COLISIÓ ENTRE EL CAP Y EL MENJAR DE LA SERP

    # aquí mirem la distanca entre els dos objectes, que serien el cap de la serp i el menjar
    # li donem 20 unitats, ja que les mesures del quadrat i cercle per defecte son 20 x 20 pixels

     if cap.distance(food) < 20:              # si la distanca es menor al tamany dels dos objectes, significará que s'han tocat
          x = random.randint(-380, 380)       # creem números random per assignarlos a la nostra X i Y
```

```
y = random.randint(-380, 380)          # creem un núm. enter amb possibilitat de que surti dels -380 fins als 380 per a x i y, per no estar tan aprop del marge de la finestra
    food.goto(x,y)                     # per actualitzar la possició del menjar


     nou_segment = turtle.Turtle()          # creem un objecte turtle perque es mostri algo a la pantalla
     nou_segment.speed(0)                   # per a que s'inicii la pantalla, el nou segment del cos de la serp estigui des de l'inici
    nou_segment.shape("square")             # la forma del cos també serà quadrada
    nou_segment.color("orange")             # li donem un color al cos de la serp
     nou_segment.penup()                    # amb aquest comand, tot i que el nou_segment de la serp es mogui, no hi haurá rastre o estela
      segments_cos.append(nou_segment)      # cada cop que es crei el segment de cos, s'anirà afegint a la llista "segments_cos"


     # L'obstacle canvia de posició a una posició random cada cop que la serp aconsegueix menjar

     x = random.randint(-290, 290)          # creem números random per assignarlos a la nostra X i Y
     y = random.randint(-290, 290)          # creem un núm. enter amb possibilitat de que surti dels -290 fins als 290 per a x i y, per no estar tan aprop del marge de la finestra
    obstacle.goto(x,y)                      # per actualitzar la possició de l'obstacle

     nou_obstacle = turtle.Turtle()         # creem un objecte turtle perque es mostri algo a la pantalla
     nou_obstacle.speed(0)                  # per a que s'inicii la pantalla, el nou segment del nou obstacle no es mogui
    nou_obstacle.shape("square")            # la forma del cos també serà quadrada
    nou_obstacle.color("black")             # li donem un color al obstacle
    nou_obstacle.penup()                    # amb aquest comand, no hi haurá rastre o estela
     nou_obstacle.goto(x,y)                 # un nou segment apareix en una posició aleatoria dins el marc de la finetra de joc
    obstacles.append(nou_obstacle)          # cada cop que es crei l'obstacle, s'anirà afegint a la llista "obstacles"


    # Escurçament del retard

    posposar = posposar - 0.001
```

Fet per: Juan Camilo De Los Ríos Algorismia i Programacio Audiovisual (APA)
Data: 1-06-2023

```
# AUGMENT DEL RESULTAT DEL MARCADOR

        resultat = resultat + 10              # Augmentem el resultat 10 punts, cada cop que la serp menji el menjar

       if resultat > millor_resultat:         # Si el resultat és major que el millor resultat
          millor_resultat = resultat          # El millor resultat s'actualitzarà

        # Actualitzem l'fstream amb els resultats, natejant el text (.clear()) perque no es solapin els increments del resultat
       text.clear()
         text.write(f" RESULTAT: {resultat}                    MILLOR RESULTAT: {millor_resultat} " , align = "center" , font = ("Calibri", 24, "bold"))



   # COLISIÓ ENTRE EL CAP Y L'OBSTACLE


   for obstacle in obstacles:                    # bucle for per iterar la llista d'obstacles
         if cap.distance(obstacle) < 25:                 # si la distanca es menor al tamany dels dos objectes (cap i obstacle), significará que s'han tocat
           time.sleep(1)                         # s'adorm el programa un segon
           cap.goto(0,0)                         # reset al programa fent qque el cap torni a la posició d'origen (0,0)
           cap.direction = "stop"                # i la direcció estigui en mode "stop"

           # AMAGUEM ELS SEGMENTS OBTINGUTS, AL MORIR
             for segment in segments_cos:                # Per cada segment que estigui a la llista "segments_cos"
                 segment.goto(1111,1111)           # els fem anar ben lluny (fora dels marcs de la finiestra (1111,1111))
                  obstacles.clear()                # Borrem els obstacles que queden (no funciona i no s'envan)
             segments_cos.clear()                  # Borrem els segments de la llista, però fora del bucle for, ja que sinó els segments quedarien dins de la finestra


           # Reset del resultat del marcador
           resultat = 0
```

```
 # Actualitzem l'fstream amb els resultats, natejant el text (.clear()) perque no es solapin els increments del resultat

        text.clear()
        text.write(f" RESULTAT: {resultat}                    MILLOR RESULTAT: {millor_resultat} " , align = "center" , font = ("Calibri", 24, "bold"))

        obstacle.clear()

         # Aquí escribim amb un fstream, el famòs "GAME OVER" per indicar que s'ha perdut la partida

        game_over.write(f" GAME OVER " , align = "center" , font = ("Impact", 55, "bold"))
        time.sleep(2)
        game_over.clear()



# MOVIMENT DEL COS DE LA SERP

   # Per moure el cos de la serp, iterarem els components de la llista. El primer índex de la llista sempre es el 0. Els paràmetres
   # del bucle for seran: l'últim element (segments_totals -1), fins al primer que és el 0 que no es inclòs dins del bucle, i l'últim
  # paràmetre será que vagi decreixent el valor (-1).

     segments_totals = len(segments_cos)  # métode per obtindre de forma entera (int), els segments de cos totals

      # Per donar una sensació d'animació, iterem aquest bucle for, però encara sense teledirigirlos cap al cap de la serp.
    # Aquest bucle es per que els segments de cos de la serp es segueixin entre ells un cop s'han afegit al cap de la serp
   for index in range (segments_totals - 1, 0, -1):

        # otenim les coordenades x i y del segment anterior perque l'últim element es mogui o segueixi al anterior
      x = segments_cos[index - 1].xcor()
      y = segments_cos[index - 1].ycor()
```

```
segments_cos[index].goto(x,y)        # amb aquesta instrucció mourem el index actual cap a les coordenades de l'element anterior i per tant seguirlo

   # Amb aquest bucle if, acabarem de teledirigir el segments del cos cap al cap de la serp
    if segments_totals >  0:        # Mirem que la llista no estigui buida, ja que si ho està,, donaria error
       x = cap.xcor()            # Obtenim la coordenada x del cap de la serp
       y = cap.ycor()            # Obtenim la coordenada y del cap de la serp
       segments_cos[0].goto(x,y)   # Fem que es mogui cap a on està el cap de la serp



   moviment()       # iniciem la funció de moviment




   # COLISIÓ ENTRE EL CAP Y EL MATEIX COS DE LA SERP (SEGMENTS DEL COS)

   # Iterem cada element de la llista "segments_cos"
   for segment in segments_cos:
         if segment.distance(cap) < 20:       # Fem que la distancia amb el segments del cos no estigui tan aprop del cap de la serp
         time.sleep(1)                # Una petita pausa d'1 seg per la colisió
         cap.goto(0,0)                # Fem que el cap de la serp torni al seu lloc d'origen (0,0)
         cap.direction = "stop"          # Aturem la direcció del cap


         # TORNEM A AMAGAR ELS SEGMENTS OBTINGUTS, AL MORIR

               for segment in segments_cos:          # Per cada segment que estigui a la llista "segments_cos"
             segment.goto(1111,1111)        # els fem anar ben lluny (fora dels marcs de la finiestra (1111,1111))

            segments_cos.clear()               # Borrem els segments de la llista, però fora del bucle for, ja que sinó els segments quedarien dins de la finestra
           obstacle.clear()                # Intento borrar els obstacles de la llista, però no puc fe-los desaparèixer
         obstacle = 0
```

```
# Reset del resultat del marcador
        resultat = 0

          # Actualitzem l'fstream amb els resultats, natejant el text (.clear()) perque no es solapin els increments del resultat
        text.clear()
        text.write(f" RESULTAT: {resultat}                    MILLOR RESULTAT: {millor_resultat} " , align = "center" , font = ("Calibri", 24, "bold"))


          # Aquí escribim amb un fstream, el famòs "GAME OVER" per indicar que s'ha perdut la partida

        game_over.write(f"GAME OVER " , align = "center" , font = ("Impact", 55, "bold"))
        time.sleep(2)
        game_over.clear()


  time.sleep(posposar)       # per a que el programa no s'executi tan rápid
```