CAMPUS TECNOLÓGICO LOCAL SAN JOSÉ

INGENIERÍA EN COMPUTACIÓN-2018

IC-1803 TALLER DE PROGRAMACIÓN GR 41

Informe Técnico

Profesor: Eduardo Adolfo Canessa Montero

edcanessa@itcr.cr

Estudiantes:

Gabriel Núñez González

gnunez@estudiantec.cr

2022058171

Emilio Costaguta Araya

ecostaguta@estudiantec.cr

2022437795

San José, Costa Rica

Introducción:

Un laberinto según la Real Academia Española (RAE) es un, “lugar formado artificiosamente por calles y encrucijadas, para confundir a quien se adentre en él, de modo que no pueda acertar con la salida.”(RAE, s.f) Esto significa que es un acertijo que hace pensar de maneras diferentes el cerebro humano para llegar a su destino es este caso la salida. En este proyecto se quiere llegar a programar laberintos de manera aleatoria o ya sea que el usuario digite su laberinto, mediante el lenguaje de programación Python para que el usuario pueda interactuar con la interfaz.

“Python es un lenguaje de multiparadigma, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web.” (Anónimo, 2003). Esto significa que Python es un programa versátil que cuenta con todos los paradigmas, por lo tanto, para este proyecto se usó este lenguaje para desarrollar una calculadora científica.

Por lo tanto, para la programación de esta interfaz se usó la biblioteca pygame, la cual conlleva muchas funciones que se necesitan y útiles para la implementación de un juego como el laberinto, las cuales van desde una función para iniciar el programa hasta cuando el mouse está sobre un objeto o botón.

Apéndice 1:

- Entradas: las entradas serian el ancho y el alto del laberinto para que se genere aleatoriamente o un laberinto preconstruido
- Salidas: La salida en una ventana donde la matriz que hayamos puesto se convierta en sprites y alguno se puedan atravesar y otros no, par asi tener el laberinto
- Restricciones: El inicio y el fin deben de estar en los extremos y no puede ser adyacentes
- Resumen: Lo primero que vemos es como llamamos las imágenes que utilizamos de sprite y luego creamos una función para la colisión de las paredes del laberinto, esta función lo que hace es que cadas vez que detecta un True pone un rectángulo transparente de 80x80px.

```
5   ########  SPRITES ##########################################################################################
6   class muro(pygame.sprite.Sprite):                                                                          #
7       def __init__(self):                                                                                    #
8           super().__init__()                                                                                 #
9           self.image=pygame.image.load("ParedAmarilla.jpg").convert_alpha()                                  #
10          self.rect=self.image.get_rect()                                                                    #
11  class suelo(pygame.sprite.Sprite):                                                                         #
12      def __init__(self):                                                                                    #
13          super().__init__()                                                                                 #
14          self.image=pygame.image.load("Nuevo suelo2.png").convert_alpha()                                   #
15          self.rect=self.image.get_rect()                                                                    #
16  class PJ (pygame.sprite.Sprite):                                                                           #
17      def __init__(self):                                                                                    #
18          super().__init__()                                                                                 #
19          self.image=pygame.image.load("PJ_DS.png").convert_alpha()                                          #
20          self.rect=self.image.get_rect()                                                                    #
21  class inicio(pygame.sprite.Sprite):                                                                        #
22      def __init__(self):                                                                                    #
23          super().__init__()                                                                                 #
24          self.image=pygame.image.load("InicioColor.jpg").convert_alpha()                                    #
25          self.rect=self.image.get_rect()                                                                    #
26  class Final(pygame.sprite.Sprite):                                                                         #
27      def __init__(self):                                                                                    #
28          super().__init__()                                                                                 #
29          self.image=pygame.image.load("FinalColor.jpg").convert_alpha()                                     #
30          self.rect=self.image.get_rect()                                                                    #
31          self.rect=self.image.get_rect()                                                                    #
32  Logo=pygame.image.load("Logo.png")
33  YouWin=pygame.image.load("You Win.jpg")
34  Bosque=pygame.image.load("Fondo bosque.jpg")  #fondo de pantalla                                           #
35  ##########################################################################################################
```

Luego declaramos la mayoría de las funciones que usaremos a lo largo de todo el codigo, tales como el eje “x” y “y” el alto y el ancho, colores y el final e inicio del laberinto. Despues de eso ponemos todos los parámetros de nuestra ventana que su tamaño se adapta al del laberinto que se genera.

```
#  parametros de la ventana   ##########################################################
WIDTH, HEIGHT=(n*80)+400,m*80 #variable para el tamano                                  #
FPS= 30                                                                                 #
WIN= pygame.display.set_mode((WIDTH,HEIGHT))  #variable que contiente la venntana      #
pygame.display.set_caption("Laberinto.io") #nombre de la ventana                        #
clock=pygame.time.Clock()                                                               #
##########################################################################################
```

Declaramos los esprites en variables para poder utilizarlos y ponerlos en la ventana.

```
##  LISTA DE SPRITES  ###################
lista_Sprite_Muro= pygame.sprite.Group()  #
muro=muro()                               #
lista_Sprite_Muro.add(muro)               #
lista_sprite_suelo=pygame.sprite.Group()  #
suelo=suelo()                             #
lista_sprite_suelo.add(suelo)             #
lista_sprite_inicio=pygame.sprite.Group() #
inicio=inicio()                           #
lista_sprite_inicio.add(inicio)           #
lista_sprite_final=pygame.sprite.Group()  #
Final=Final()                             #
lista_sprite_final.add(Final)             #
lista_arbol_sprite=pygame.sprite.Group()  #
PJ=PJ()                                   #
lista_arbol_sprite.add(PJ)                #
#########################################
```

Y luego nuestro generador de matrices automatico, lo que basicamente es un creador de laberintos, donde True es un muro y False es suelo por donde puede pasar el jugador.

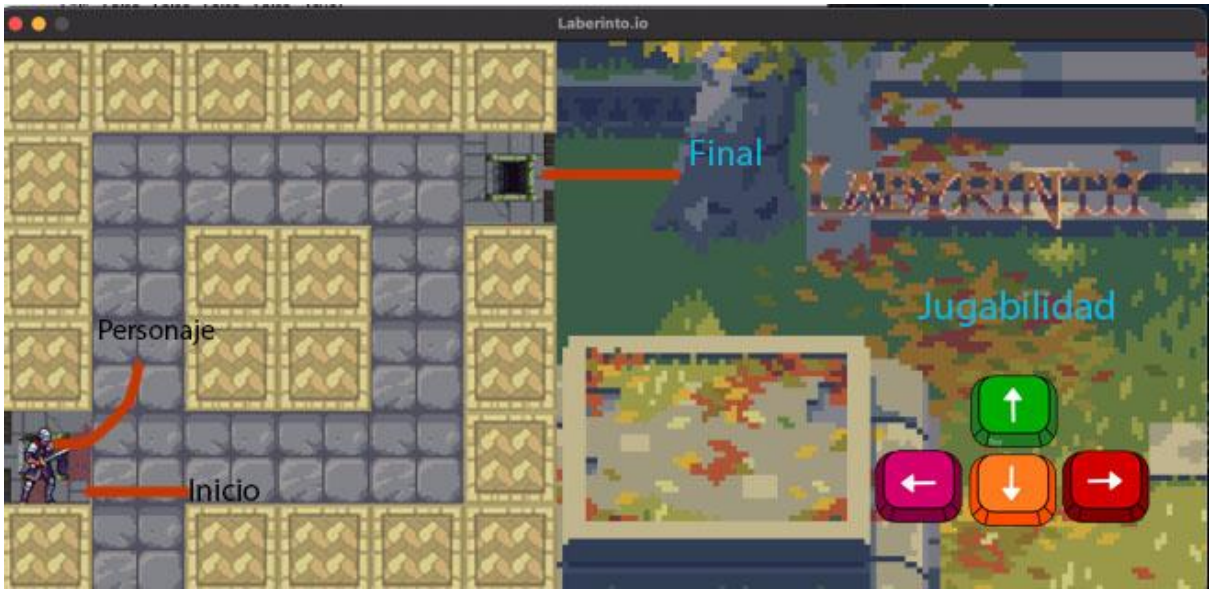
```
###  Generador de Matrices  ##################################
matriz=[]                                                    #
laberinto=[]                                                  #
def generador_matriz():                                       #
    global n                                                  #
    global m                                                  #
    print(n, m)#

    global matriz                                             #
    matriz=[[TrueFalse() for j in range(n)] for i in range(m)]  #
    print(matriz)
    global Inicio_n
    global Inicio_m
    global Final_n
    global Final_m
    print(Inicio_n, Inicio_m, Final_n, Final_m)
    matriz[Inicio_m][Inicio_n]="I"                            #
    matriz[0][1]=False                                        #
    matriz[Final_m][Final_n]="F"                              #
    matriz[-1][-2]=False                                      #
    return matriz                                             #
def TrueFalse():                                              #
    lista= random.choice([True, False])                       #
    return lista                                              #
laberinto= generador_matriz()                                 #
print(laberinto)                                             #
#############################################################
if matrizUsuario==[]:
    listaMatriz= ParedesLaberinto(laberinto)
else:
    laberinto= matrizUsuario
    listaMatriz= ParedesLaberinto(laberinto)
```

Y casi al final agregamos las funcionalidades del teclado y de la ventana en si, es decir que cuando le demos al botón QUIT de la venta esta se cierre, ademas configuramos el personaje para que se mueva con las flechas del teclado y que no se salga de los margenes de la ventana.

Y por último agregamos los sprites a la ventana

Apéndice 2:

Instrucciones



Conclusiones:

En conclusión, se diseñó mediante la investigación de la biblioteca pygame y todas sus funciones la interfaz gráfica y practica del laberinto, dando como resultado una interfaz donde el usuario puede digitar su laberinto favorito o que el propio programa genere una aleatorio con dimensiones que el mismo usuario requiera o guste.

Recomendaciones:

Al concluir este proyecto se recomienda por parte del grupo, investigar de manera minuciosa y astuta cualquier tipo de información que el programador ocupe, ya sean videos, artículos, etc; ya que por experiencia hay veces que no se tienen conocimiento suficiente de una biblioteca y de cómo implementarla de manera eficiente a un programa, en el caso de pygame no era algo nuevo pero en cada pero mediante la búsqueda de datos se logró conseguir el objetivo propuesto.

Literatura consultada:

- Python. (s. f.). *Pygame Front Page — pygame v2.1.1 documentation*. Pygame. https://www.pygame.org/docs/
- *laberinto pygame*. (2020, 29 diciembre). YouTube. https://www.youtube.com/watch?v=9ZkPsDXzrB0&t=993s
- *Qué es Python*. (2003, 19 noviembre). Desarrollo Web. https://desarrolloweb.com/articulos/1325.php