

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Fundamentos de programación

**Actividad asíncrona 10**

Alumna: Rivera González Frida Alison

M.I. Marco Antonio Martínez Quintana

No. lista: 40

Semestre: 2021-1

Grupo: 3

06/11/2020

**Pseudocódigo**

**Juegos arcade**

Juego snake

Codigo:  
  
  
Pantalla de carga (load\_Screen):

import Tkinter as tk # Python 2

import tkinter as tk # Python 3

import Imagenes #Libreria especial Creada por mi para la carga de Imagenes

import time #Libreria para el manejo de tiempos y cronometros

class load\_window():

root=object

def \_\_init\_\_(self): #Declaro el constructor de Clase

self.root = tk.Tk() #Creo la ventana Tk

# The image must be stored to Tk or it will be garbage collected.

imagen=Imagenes.get\_imagen("logo2.png",1000,1000) #Llamo la funcion get\_imagen para cargar una imagen a python

label = tk.Label(self.root, image=imagen,bg="black") #creo un label, el cual contendra la imagen

self.root.geometry("+470+200") #establesco la posicion de la pantalla de carga en el monitor

self.root.overrideredirect(True) #Le establesco la propiedad de una ventana traslucida

self.root.lift() #Le digo que al cargarse debera aparecer arriba de todas las ventanas creadas

self.root.wm\_attributes("-topmost", True) #le establesco que no debera poseer los botones de minimizar, maximizar y cierre

#self.root.wm\_attributes("-disabled", True) #Si no estuviera comentada y no existiese la linea anterior desabilitaria las funciones de los botones antes comentados

self.root.wm\_attributes("-transparentcolor", "black") #Le digo que la ventana debera actuar con transparencia en todo tono negro

label.pack() #cargo el label

# Establezco eventos para el raton

self.root.bind("", lambda e:self.exit()) #Si el mouse deja la ventana traslucida

self.root.bind("",lambda e:self.exit()) #o le doy click sobre la imagen

#debera ejecutar la funcion exit

self.root.mainloop()

def exit(self):

print "entro"

time.sleep(5) #espera 5s

self.root.destroy() #cierra esta ventana

import Snake #importa la libreria Snake correspondiente al archivo numero 2

"""Nota: La importancia de if \_\_name\_\_=='\_\_main\_\_':

en python es el siguiente:

Si no es colocado e importo dicho archivo como libreria, al momento de hacer la importacion todas

las instrucciones de dicho archivo se ejecutaran, en mi caso utilizo esa funcionalidad para ejecutar la ventana

del juego sin mas lineas de codigo, pues como podran observar en ese archivo, esa linea no la utilize.

De colocarla al momento de hacer la importacion, el codigo escrito no deberia ejecutarse como tal.

"""

c=load\_window() #Creo un objeto que a si vez crea la ventana de carga

Ahorcado

from funciones import \*

listado=[]

while True:

print("Seleccionar opción deseada")

print("1. Cargar palabras")

print("2. Jugar")

print("3. Salir")

opcion=int(input())

if opcion==1:

listado=cargarPalabras(listado)

elif opcion==2:

letrasAdivinadas=set()

intentosRestantes=6

palabra=seleccionarPalabra(listado)

print("La palabra tiene ", len(palabra), " letras.")

while intentosRestantes > 0:

if palabraCompleta(palabra, letrasAdivinadas):

print("¡Ganaste!")

break

mostrar=progresoDelJuego(palabra, letrasAdivinadas)

print(mostrar)

letra=leerLetra(letrasAdivinadas)

letrasAdivinadas.add(letra)

if letra not in palabra:

intentosRestantes-=1

print("¡NOP! Intentos restantes: ", intentosRestantes)

else:

print("¡SIP!")

elif opcion==3:

print("¡Hasta la próxima!")

break

else:

print("Opción inválida")

**funciones.py**

import random

def seleccionarPalabra(listadoPalabras):

return listadoPalabras[random.randint(0, len(listadoPalabras)-1)]

Buscaminas

import pygame

# Definimos algunos colores

NEGRO = (0, 0, 0)

BLANCO = (255, 255, 255)

VERDE = ( 0, 255, 0)

ROJO = (255, 0, 0)

# Establecemos el LARGO y ALTO de cada celda de la retícula.

LARGO = 20

ALTO = 20

# Establecemos el margen entre las celdas.

MARGEN = 5

# Creamos un array bidimensional. Un array bidimensional

# no es más que una lista de listas.

grid = []

for fila in range(10):

# Añadimos un array vacío que contendrá cada celda

# en esta fila

grid.append([])

for columna in range(10):

grid[fila].append(0) # Añade una celda

# Establecemos la fila 1, celda 5 a uno. (Recuerda, los números de las filas y

# columnas empiezan en cero.)

grid[1][5] = 1

# Inicializamos pygame

pygame.init()

# Establecemos el LARGO y ALTO de la pantalla

DIMENSION\_VENTANA = [255, 255]

pantalla = pygame.display.set\_mode(DIMENSION\_VENTANA)

# Establecemos el título de la pantalla.

pygame.display.set\_caption("Retículas y Matrices")

# Iteramos hasta que el usuario pulse el botón de salir.

hecho = False

# Lo usamos para establecer cuán rápido de refresca la pantalla.

reloj = pygame.time.Clock()

# -------- Bucle Principal del Programa-----------

while not hecho:

for evento in pygame.event.get():

if evento.type == pygame.QUIT:

hecho = True

elif evento.type == pygame.MOUSEBUTTONDOWN:

# El usuario presiona el ratón. Obtiene su posición.

pos = pygame.mouse.get\_pos()

# Cambia las coordenadas x/y de la pantalla por coordenadas reticulares

columna = pos[0] // (LARGO + MARGEN)

fila = pos[1] // (ALTO + MARGEN)

# Establece esa ubicación a cero

grid[fila][columna] = 1

print("Click ", pos, "Coordenadas de la retícula: ", fila, columna)

# Establecemos el fondo de pantalla.

pantalla.fill(NEGRO)

# Dibujamos la retícula

for fila in range(10):

for columna in range(10):

color = BLANCO

if grid[fila][columna] == 1:

color = VERDE

pygame.draw.rect(pantalla,

color,

[(MARGEN+LARGO) \* columna + MARGEN,

(MARGEN+ALTO) \* fila + MARGEN,

LARGO,

ALTO])

# Limitamos a 60 fotogramas por segundo.

reloj.tick(60)

# Avanzamos y actualizamos la pantalla con lo que hemos dibujado.

pygame.display.flip()

# Pórtate bien con el IDLE.

pygame.quit()