TRABAJO PRÁCTICO
PROGRAMACIÓN
JUEGO "PALABRAS ENCOLUMNADAS"
TEMA: PYTHON & PYGAME

INTEGRANTES: GRUPO K
- Santana Seumal Francisco
- Wanuffelen Oviedo Luciana Daniela

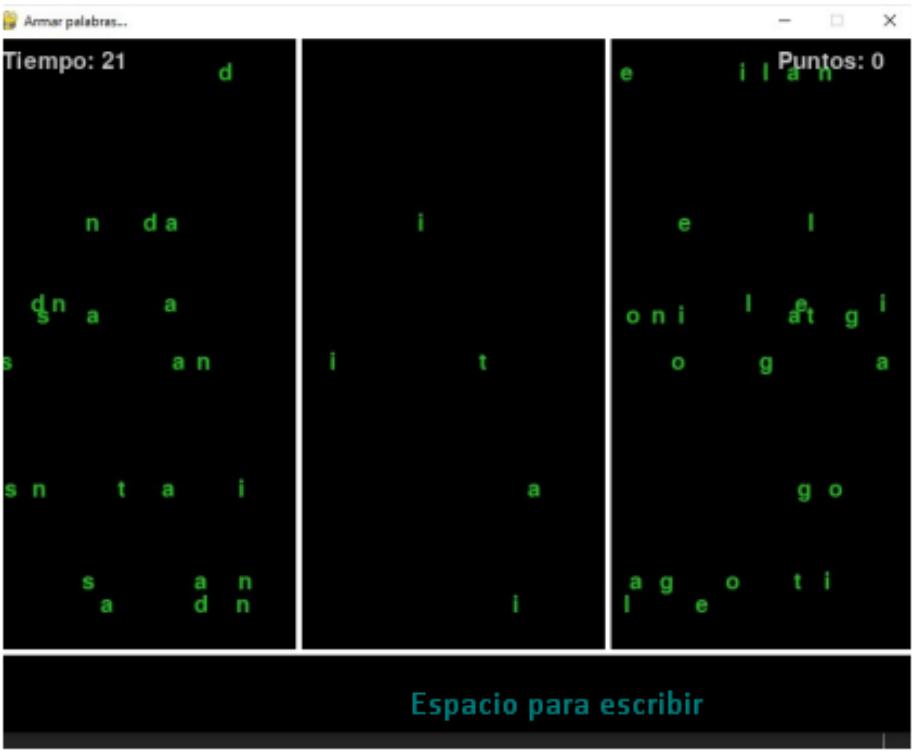
COMISIÓN: 02
TURNO: Tarde
UNIVERSIDAD: Universidad Nacional General Sarmiento
AÑO: 2021

INTRODUCCIÓN

Durante este trabajo estaremos explicando cómo fuimos realizando el juego propuesto por los profesores para la aprobación de la materia. Las funciones que tuvimos que realizar para hacer el funcionamiento básico del mismo, los problemas que tuvimos, como fuimos resolviéndolo, entre otras explicaciones que estarán por leer.

DESARROLLO

El juego es construido por tres columnas donde en cada una van bajando letras “aleatorias” y en posicionamiento de igual manera. Nosotros como jugadores tenemos que armar la palabra según las letras que aparezcan, en un tiempo determinado, en caso de adivinar una palabra se le entrega los puntos dependiendo de las letras que construyen esa palabra. De todas maneras hay ciertas reglas que hay que seguir para que se otorguen esos puntos, las iremos comentando a lo largo del desarrollo del trabajo.
En la pantalla contamos con un renglón donde es ese el lugar en el que debemos escribir la palabra.



Lo interesante de este juego es que aunque las letras formaban originalmente una palabra si al jugador se le ocurre otra opción, siempre y cuando se encuentre en el lemario creado y sea posible crearla a partir de las letras disponibles, será considerada válida. Dando con una jugada válida y se entregarán los puntos.

. REGLAS

- El juego es de un solo jugador, es decir single player.
- El objetivo del juego es escribir la mayor cantidad de palabras en el tiempo establecido que existan en el lemario español.
- El tiempo establecido es de 60 segundos.

- El usuario puede cambiar de columna para crear la palabra pero solo una vez. Es decir si en la columna que estamos jugando no está la letra para construir la palabra, sino que está en alguna de las otras dos. Seguimos con alguna de ellas pero no podemos volver a elegir otra columna, nos estancamos en ella por el resto del tiempo.
- Los puntos se entregan dependiendo de las letras que construyan la palabras:
    + Cada vocal otorga 1 punto.
    + Cada consonante otorga 2 puntos, salvo las difíciles.
    + Las consonantes difíciles (j, k, q, w, x, y, z) otorgan 5 puntos

. CONSTRUCCIÓN DEL JUEGO

principal.py:

En ella configuramos la función main, en ella se programó la pantalla, el tiempo, todas las listas que usaremos a lo largo del juego, inicializamos los puntos y la candidata (es el input en forma de cadena). También abrimos el lemario en extensión txt y con un ciclo for vamos guardando todo lo que esté en el. De todas maneras tuvimos que hacer unos retoques en la codificación de la apertura del documento para componer la lista de forma comoda:

```
with
open("C:/Users/usuario/Documents/programacion/TP-en-columnas/lemarioMasCorto.txt","
r") as archivo:
        no="\n"
        abc = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','ñ','o','p','q','r','s','t','u','v','w','x','y','z']
        for linea in archivo:
            lista.append(linea)
            for caracter in range(len(lista)):
                lista[caracter] = lista[caracter].replace(no,"")
                for dato in range(len(abc)):
                    if lista[caracter]==abc[dato]:
                        lista[caracter] = lista[caracter].replace(abc[dato],"")
                if lista[caracter]=="":
                    lista.pop(caracter)
```

También cargamos funciones como cargarListas, que como dice el nombre cargamos las listas de las tres columnas y las listas de las posiciones dependiendo de la lista que hicimos con el lemario. Luego está la función dibujar que es como está configurada la pantalla junto con las listas, los puntos, la candidata y el tiempo.

Luego se especifican los fps, y como se cierra la pantalla (por el tiempo o haciendo click en la cruz) y el color de la misma.

Al final de la página ejecutamos la única función de la hoja, o sea ejecutamos main.

Otra parte importante es que la única hoja la importamos en las otras hojas.

configuracion.py:

En ella no contamos con ninguna función sino todos los detalles de la página. Como el tamaño de la letra, los segundos de reproducción de la pantalla, todos los colores que usaremos en la pantalla en formato rgb, el alto y ancho de la página.

funcionesVACIAS.py:

En ella es donde nosotros como alumnos debemos realizar las funciones principales para hacer el funcionamiento básico del juego, contamos con:

+ cargarListas:
con los siguientes parámetros dados: lista, listaIzq, listaMedio, listaDer, posicionesIzq, posicionesMedio, posicionesDer.
En esta función tenemos que lograr elegir una palabra al azar del lemario creado, tomar sus letras y repartirlas entre las tres listas. Luego otorgarles una posición al azar, dependiendo en qué columna se encuentre la letra. El código que hicimos para poder hacer posible fue el siguiente:

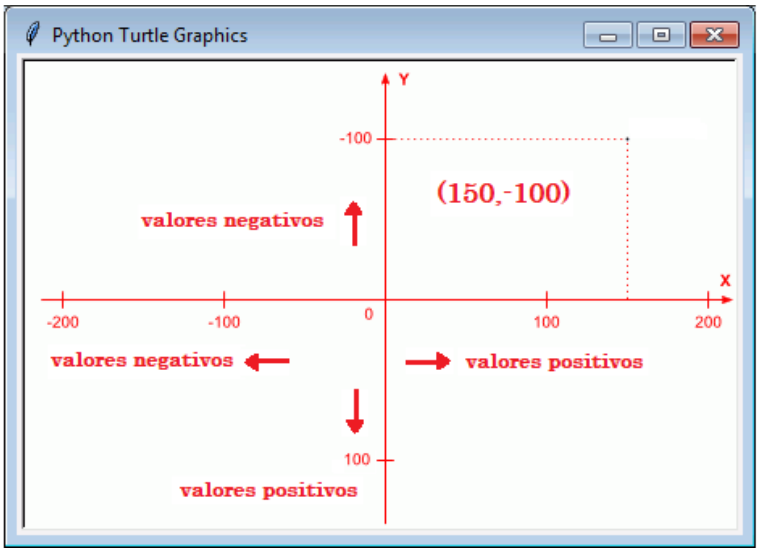
```
def cargarListas(lista, listaIzq, listaMedio, listaDer, posicionesIzq, posicionesMedio, posicionesDer):
    if (len(lista)>0):
        palabra = lista[random.randint(0, len(lista)-1)]
        palabra1=""
        palabra2=""
        palabra3=""
        if (len(palabra)>2):
            division1= random.randint(1, len(palabra)-2)
            for i in range(division1):
                palabra1=palabra1+palabra[i]
            for letra in palabra1:
                borde_desde = 1
                borde_hasta = PRIMER_LINEA_VERTICAL-15
                        guardarLetraEnListas(listaIzq, posicionesIzq, letra, borde_desde, borde_hasta)

            division2= random.randint(division1+1, len(palabra)-1)
            for i in range(division1, division2):
                palabra2=palabra2+palabra[i]
            for letra in palabra2:
                borde_desde = PRIMER_LINEA_VERTICAL+15
                borde_hasta = SEGUNDA_LINEA_VERTICAL-15
                  guardarLetraEnListas(listaMedio, posicionesMedio, letra, borde_desde, borde_hasta)

            for i in range(division2, len(palabra)):
                palabra3=palabra3+palabra[i]
            for letra in palabra3:
                borde_desde = SEGUNDA_LINEA_VERTICAL+15
                borde_hasta = ANCHO-15
                        guardarLetraEnListas(listaDer, posicionesDer, letra, borde_desde, borde_hasta)
```

+ Guardar en listas:
con los siguientes parámetros dados: lista, posiciones, letra, borde_izquierdo, borde_derecho.

En ella nos encargamos de las posiciones. Elegimos una posición aleatoria del eje x, pero en el eje y establecemos un número fijo mayor a cero. ¿Por qué? porque si elegimos y=0 las letras se colocan debajo de los marcadores y nos quitan visión.

```
def guardarLetraEnListas(lista, posiciones, letra, borde_izquierdo, borde_derecho):
    pos_y = 20
    lista.append(letra)
    pos_x = random.randint(borde_izquierdo, borde_derecho)
    posiciones.append([pos_x, pos_y])
```

+ bajar:
con los siguientes parámetros dados: lista, posiciones.
En ella debemos lograr que las letras bajen, para ello debemos sumarle x valor a los valores del eje y. No debemos tocar el eje, en este caso. Una vez que lleguen a "tocar el piso", debemos eliminar la palabra de la lista.
Así es como funcionan los ejes en pygame:



Pasamos a ver el código:

```
def bajar(lista, posiciones):
if len(lista) > 0:
        movimiento_y = 2
        for pos_x_y in posiciones:
            pos_x_y[1] += movimiento_y  # baja la posicion en y de la letra
            if pos_x_y[1] > ALTO-90:  # si la letra llego al piso
                lista.pop(0)
                posiciones.pop(0)
```

+ actualizar:
con los siguientes parámetros dados: lista, listaIzq, listaMedio, listaDer, posicionesIzq , posicionesMedio, posicionesDer.

Esta hace que la anterior función se haga realidad, es decir esta actualiza todas las listas, elimina las que tocaron el suelo y carga nuevas opciones a las listas que tiene como parámetros.

```
def actualizar(lista, listaIzq, listaMedio, listaDer, posicionesIzq, posicionesMedio,
posicionesDer):
    bajar(listaIzq, posicionesIzq)
    bajar(listaMedio, posicionesMedio)
    bajar(listaDer, posicionesDer)
    if random.randint(1, 20) == 1:
        cargarListas(lista, listaIzq, listaMedio, listaDer, posicionesIzq,
posicionesMedio, posicionesDer)
```

+ estaCerca:
  con los siguientes parámetros dados: elem, lista. Con ella lo que hacemos es evitar los solapamientos, es decir que no existan sobreposiciones entre las letras.
+ Puntos:
  con los siguientes parámetros dados: elem, lista.
  En ella tratamos de darle un puntaje a la candidata. En esta parte del trabajo aún no sabemos si la candidata es correcta o no, en la función siguiente sabremos eso.
  Otorgamos el puntaje acorde de las letras que construyen la palabra como vimos en las reglas:
  + por vocal el programa otorga 1 punto,
  + por consonante difícil (j, k, q, w, x, y, z) otorga 5 puntos,
  + por consonante diferente a las consonantes difíciles otorga 2 puntos.

  Pasamos a ver el código:

```
def Puntos(candidata):
    vocales = "aeiou"
    cons_dificiles = "jkqwxyz"
    total_puntos = 0
    for letra in candidata:
        if letra in vocales:
            total_puntos += 1
        elif letra in cons_dificiles:
            total_puntos += 5
        else:
            total_puntos += 2
    return total_puntos
```

+ procesar:
  con los siguientes parámetros dados: lista, candidata, listaIzq, listaMedio, listaDerecha.
  Como dijimos en el anterior punto este iba a ser la función para averiguar si era correcta o no la candidata. En caso de ser verdadera se devuelve el puntaje visto en la anterior función o cero en caso contrario. Pero ¿cómo sabemos que es verdadera?, lo veremos en la próxima función. Pero antes eso debemos ver este código:

```
def procesar(lista, candidata, listaIzq, listaMedio, listaDer, posicionesIzq,
posicionesMedio, posicionesDer):
sound= pygame.mixer.Sound("sonido.mp3")
   if esValida(lista, candidata, listaIzq, listaMedio, listaDer, posicionesIzq,
posicionesMedio, posicionesDer):
      sound.play()
      return Puntos(candidata)
   else:
      return 0
```

+ esValida:

  con los siguientes parámetros dados: lista, candidata, listaIzq, listaMedio, listaDerecha.

  Debemos revisar si la candidata está efectivamente en la lista. En caso de ser así devuelve True y en caso contrario False. Y de esa forma el programa sabrá si sí o no retornar los puntajes y borrarla de las listas.

```
if candidata in lista:
      pos_izq_usadas = []
      pos_medio_usadas = []
      pos_der_usadas = []
      col = 1
      i = 0
      while i < len(candidata):
         l = candidata[i]  # letra
         if col == 1:
            if l in listaIzq:
               pos_izq_usadas.append(listaIzq.index(l))
               i += 1
            else:
               col = 2
         elif col == 2:
            if l in listaMedio:
               pos_medio_usadas.append(listaMedio.index(l))
               i += 1
            else:
               col = 3
         else:
            if l in listaDer:
               pos_der_usadas.append(listaDer.index(l))
               i += 1
            else:
               return False
      borrarLetrasDeListas(listaIzq, posicionesIzq, pos_izq_usadas)
      borrarLetrasDeListas(listaMedio, posicionesMedio, pos_medio_usadas)
      borrarLetrasDeListas(listaDer, posicionesDer, pos_der_usadas)
      return True
   else:
      return False
```

+ borraLetrasDeLIstas:

con los siguientes parámetros dados: lista, posiciones, pos_usadas.
En ella vamos borrando las letras de aquellas palabras que fueron escritas por el usuario, que pasaron por todas las funciones anteriores. Pasamos a ver el código:

```
def borrarLetrasDeListas(lista, posiciones, pos_usadas):
    pos_usadas.sort()
    cont_pop = 0
    for pos in pos_usadas:
        lista.pop(pos-cont_pop)
        posiciones.pop(pos-cont_pop)
        cont_pop += 1
```

extras.py:
En ella contamos con un par de funciones, como dameLetraApretada que en ella se codificó que cada vez que se presiona una tecla nos retornara la letra de la tecla que se presiona en formato cadena. E incluso si presionamos la barra de espacio o si no seleccionamos nada es un espacio vacío.
Otra función es escribirEnPantalla donde los parámetros son la pantalla, posición, tamaño y el color. En ella definimos la fuente de las letras, los colores de las mismas, y se imprimiran en la pantalla en función de lo ya codificado y la posición dada.
Y por último está la función dibujar con ciertos parámetros dados como la pantalla, todas las listas que contienen las letras de las palabras aleatorias y listas de las posiciones. Aca establecemos la fuente que definimos en la anterior para hacer uso de ella, luego se dibujan dos líneas verticales haciendo alusión a las tres columnas y una horizontal para hacer el espacio para escribir. Después tenemos especificada la candidata, y pasamos a llamar a la variable puntos junto con la palabra “puntos”, establecemos su color y la fuente.
También definimos que cuando el tiempo esté entre 60-31 segundos se pinte de blanco, cuando el tiempo esté entre 30-11 segundos será de color naranja y cuando sea menor a 10 será de color rojo.

```
if(segundos<11):
        ren3 = fuente.render("Tiempo: " + str(int(segundos)), 1, COLOR_TIEMPO_FINAL)
   else:
      if(segundos<31):
         ren3 = fuente.render("Tiempo: " + str(int(segundos)), 1,
COLOR_TIEMPO_MEDIO)
      else:
         ren3 = fuente.render("Tiempo: " + str(int(segundos)), 1, COLOR_TEXTO)
```

Luego tenemos tres ciclos for que sirven para imprimir las letras en función de las posiciones dadas al azar.

.DIFICULTADES

Nuestro mayor enemigo durante toda la programación del juego fue el tiempo, ya que incluso una semana antes de la entrega estuvimos cambiandonos de grupo.

Otro factor que nos jugó en contra fue la inexperiencia de usar pygame, aunque nosotros como desarrolladores no tuvimos que codificar nada en este lenguaje fue complicado trabajar con él. Fuimos resolviendo junto con los profesores y haciendo uso de internet.
También fue la primera vez que tuvimos que trabajar con algo ya comenzado, con esto quiero decir que la gran mayoría del trabajo ya estaba desarrollado, nosotros solo debíamos desarrollar las funciones principales. Algunas veces surgían problemas y era complicado saber por qué se producían.
La falta de cooperación de los otros integrantes nos atrasó mucho, tanto de otros grupos en los que estuvimos o incluso en este grupo, ya que somos cuatro integrantes pero nomas cooperamos dos.

CONCLUSIÓN

El trabajo fue complicado pero gracias a mi compañero pudimos finalizarlo, y como consecuencia poder entregarlo completo y prolijo.
Una visualización del trabajo finalizado: