**¡Atrápame si puedes!**

Estudiantes:

* Albornoz, Magalí.
* Gómez, Marcelo.
* Sequeira, Maximiliano.

***Detalles del Juego:***

“*Atrápame si puedes*” es un juego interactivo en el que el jugador, en un lapso de 60 segundos, deberá ingresar la mayor cantidad de palabras que aparecen en pantalla. En caso de hacerlo de manera correcta, sumará los puntos correspondientes a dicha palabra. Por ende, cuantos más aciertos, ¡más alta será su puntuación!

***Especificaciones del Programa:***

En cuanto a los archivos originales, se realizaron las siguientes modificaciones:

* Modificación del Ancho de pantalla.
* Modificación del fondo de pantalla de Juego.
* Se agregaron NIVELES de dificultad a elección del jugador.
* Se agregaron efectos de sonidos de ACIERTO y ERROR.\*\*

\*\* Para lograr insertar archivos musicales, se realizaron búsquedas de información online con las cuáles el equipo de trabajo obtuvo las funciones de la librería “pygame” que lo permitieron.

***Funciones del Programa:***

Nuestro juego cuenta con una gran base de datos en la que se encuentran cargados los nombres (palabras) que se ven reflejados en pantalla. Para hacer uso de ella, definimos la siguiente función:

def InitNombres():

archivo = open("nombres.txt")

lista=archivo.readlines()

nombres=[]

for nombre in lista:

nombreNuevo = ""

for pos in range(len(nombre)-1):

nombreNuevo += nombre[pos]

nombres.append(nombreNuevo)

archivo.close()

return nombres

De esta manera, logramos abrir un archivo de texto (\*.txt) y extraer el listado de nombres que tiene en su interior para guardarlo dentro de una *“lista”* que será la base del programa. A su vez, dentro de la función es necesario el uso de un *“for”* ya que los nombres extraídos contienen un *“\n”* al final de cada uno, y el mismo no debe verse reflejado en la pantalla.

Una vez que contamos con la lista de nombres, generamos una función para tomar, de forma aleatoria, un elemento de esa lista:

def nuevaPalabra(nombres):

NombresAleatorios=random.choice(nombres)

return cambioMayusPorMinus(NombresAleatorios)

Tal como se puede observar, dentro de la función *nuevaPalabra(nombres)* se debió utilizar otra función, *cambioMayusPorMinus(nombres*). Esto fue a causa de que los nombres contenidos en la lista original, comienzan con letra mayúscula. El problema que encontramos con esto es que, una vez que el programa corre, todo lo tecleado por el jugador es ingresado en letras minúsculas. De esa forma, las palabras nunca figuran como correctamente ingresadas.

La solución hallada se basa en la función *dameLetraApretada(key)* que se encuentra el archivo “extras”. Así, logramos que por cada letra mayúscula en una cadena, nos devuelva la misma letra, pero en minúscula y dentro de una nueva cadena.

Otra opción fue intentar que la primer letra ingresada por el jugador, comience con letra mayúscula, pero no fue posible lograrlo. Por este motivo, decidimos implementar la siguiente función:

def cambioMayusPorMinus(nombre):

nombreDeSalida=""

for char in nombre:

if char == "A":

nombreDeSalida= nombreDeSalida+"a"

elif char=="a":

nombreDeSalida=nombreDeSalida+char

**{ Similiar proceder para letras desde B hasta Y}**

elif char=="Z":

nombreDeSalida=nombreDeSalida+"z"

elif char=="z":

nombreDeSalida=nombreDeSalida+char

return nombreDeSalida

A continuación, se detalla la composición de la función *actualizar*. La misma se encarga de generar diferentes puntos (de forma aleatoria) tanto para las coordenadas correspondientes al eje X, como para el eje Y, que luego serán los puntos exactos donde aparecerán los nombres en pantalla. Dichas posiciones se almacenan en una matriz, la cual nos permitirá modificar las posiciones en el eje Y, para así lograr el descenso de las palabras a través del alto de pantalla.

A su vez, se encarga de obtener una nueva palabra para almacenar en la lista de palabras que aparecerán en pantalla.

En caso de que las palabras desciendan hasta alcanzar el valor de la línea de fondo, la función se encargará de quitarlas de pantalla. Para ello se realiza un “pop” simultáneo entre la lista de palabras en pantalla, y la lista con las posiciones que le corresponden a dicha palabra.

def actualizar(listaPalabras, posiciones, nombres):

x=random.randint(20,900)

y=random.randint(000,000)

nombre=nuevaPalabra(nombres)

listaPalabras.append(nombre)

posiciones.append([x,y])

pos=0

for i in listaPalabras:

posiciones[pos][1] += 30

if posiciones[pos][1] > 515:

listaPalabras.pop(pos)

posiciones.pop(pos)

pos+=1

En el siguiente caso, el enfoque fue puesto en realizar una función que se encargue de *quitar* una palabra y su respectiva posición en la pantalla, en caso de ser necesario. Para ello, tomamos una palabra en específico, y la comparamos con la lista de palabras que se muestran en pantalla, teniendo también en cuenta la lista de posiciones que les corresponden. Si la palabra ingresada se encuentra en la lista, simplemente se quita dicha palabra de la lista, como también su correspondiente posición. De esta manera, ya no aparecerá en pantalla.

def quitar(palabra, listaPalabras, posiciones):

pos=0

for i in listaPalabras:

if i == palabra:

listaPalabras.pop(pos)

posiciones.pop(pos)

pos+=1

return True

La función *esValida* cumple la tarea de determinar si una palabra ingresada por el jugador pertenece a la lista que almacena las palabras que se muestran en pantalla:

def esValida(palabra, listaPalabras):

if palabra in listaPalabras:

return True

En este caso, la función Puntos se encarga de asignar el valor correspondiente a las palabras correctamente ingresadas por el jugador.

• Las vocales otorgan 1 punto cada una

• Las consonantes otorgan 2 puntos, salvo las difíciles\*

• Las consonante difíciles \*(j, k, q, w, x, y, z) otorgan 5 puntos.

Se asignan cadenas (formadas por los grupos de letras de tipo similiar) a variables. De esta manera, al recorrer las palabras ingresadas por el jugador y encontrar letras correspondientes a las diferentes cadenas, se obtiene el puntaje esperado.

def Puntos(palabra,listaPalabras):

vocal="aeiou"

cons\_facil="bdfghlmnprstv"

cons\_dif="jkqwxyz"

puntaje=0

if esValida(palabra,listaPalabras):

for i in palabra:

if i in vocal:

puntaje+=1

if i in cons\_facil:

puntaje+=2

if i in cons\_dif:

puntaje+=5

return puntaje

La función *procesar* tiene la finalidad de combinar las diferentes funciones para evaluar las palabras ingresadas por el jugador. Realiza una validación con la función esValida, y en caso afirmativo, procede a dar un puntaje y quitar la palabra de la pantalla, utilizando las funciones antes detalladas. Además, se encarga de reproducir un sonido por cada acierto.

En caso negativo, no suma puntos y tampoco quita la palabra de pantalla, pero se encarga de reproducir un sonido marcando el error de tipeo.

def procesar(palabra, listaPalabras, posiciones):

if esValida(palabra,listaPalabras):

pygame.mixer.music.load("SONIDO3.mp3")

pygame.mixer.music.play()

puntaje= Puntos(palabra,listaPalabras)

quitar(palabra,listaPalabras,posiciones)

return puntaje

else:

pygame.mixer.music.load("ERROR.mp3")

pygame.mixer.music.play()

return 0

***Niveles:***

El sistema de niveles se basa en la elección de 5 (cinco) velocidades de descenso de las palabras en pantalla. Para ello, se decidió modificar la cantidad de *Frames* o cuadros por segundo que se encuentra en el archivo *configuracion.py*, dejando a elección del jugador el grado de dificultad que desea.

Pero, para evitar inconvenientes o una mala experiencia del jugador, se añadió un “*WHILE*”. El mismo no permite que se ingresen valores mayores a 5, o menores a 1. Dando una buena jugabilidad al programa.

NIVEL = (int(input("Ingrese el nivel de dificultad que desea - 1 a 5-")))

while (NIVEL>5) or (NIVEL<1):

NIVEL = ((int(input("Debe ingresar un valor entre 1 y 5"))))

FPS\_inicial = NIVEL

***Sonido:***

En un principio, se decidió colocar un sonido “de fondo” durante el tiempo que dure el juego. El problema encontrado fue que dicho sonido, continuaba sonando una vez transcurrido el tiempo establecido. Además, se implementó la utilización de sonidos para los casos de acierto y desacierto de las palabras ingresadas por el jugador. Nuevamente se generó un problema, en este caso, la reproducción de estos sonidos anulaba el sonido de fondo. Ante la no solución de dicho inconveniente, se procedió a quitar este último. De esta manera, encontramos que el juego se ejecuta de manera más “limpia”, sin excesos de sonidos que perturben al jugador, y por supuesto, sin sobresaltos en cuanto a los diferentes sonidos.

***Vidas:***

El juego cuenta con un máximo de cinco intentos fallidos al ingresar las palabras. Para lograr dicho que esto suceda, definimos la siguiente función:

def Vidas(palabra, listaPalabras):

if esValida(palabra,listaPalabras):

return 0

else:

return -1

Por empezar, declaramos la variable “vida” en el archivo *principal.py* y le asignamos el valor 5, siendo este nuestro máximo de vidas a l comenzar el juego.

Dentro de la función *Vidas*, nuevamente recurrimos a la función *esValida*. En caso de que las palabras ingresadas sean correctas, retorna 0, y en caso contrario, -1. De esta manera, con cada error restamos una vida del total asignado en el archivo *principal.py*

El problema encontrado durante la implementación fue que en todos los casos (de aciertos y errores) esta función restaba una vida. Revisando en detalle, se encontró que el error se hallaba en que el resto del programa quitaba las palabras de pantalla antes de que la nueva función pudiera evaluar si lo ingresado por el jugador era correcto. Es decir, cualquiera fuese el caso, el resultado sería negativo.

La solución fue reordenar las funciones, de manera tal que primero se evaluase el error y se descontara una vida, y luego se quitaran las palabras.

***Fin del Jueg***o:

Una vez que las vidas llegan a 0 (cero) o el tiempo de juego se acaba, implementamos la función *dibujarFIN* :

def dibujarFIN(screen, puntos):

defaultFont= pygame.font.Font( pygame.font.get\_default\_font(), 70)

renFin = defaultFont.render("FIN DEL JUEGO!!!", 1, COLOR\_TEXTO)

renFinPuntos = defaultFont.render("Puntos Totales: " + str(int(puntos)), 1, COLOR\_TEXTO)

screen.blit(renFin, (200, 200))

screen.blit(renFinPuntos, (210, 300))

Esta función se encarga de mostrar la última pantalla del juego con un mensaje de despedida y los puntajes alcanzados.