Universidad Nacional de General Sarmiento

Introducción a la Programación

Trabajo Práctico – juego – "La palabra escondida"

- **Integrantes del grupo:**

-Brussa, Mariana (marianabrussaa@gmail.com)

-Gallardo, Rocio (rociogallardo110101@gmail.com)

- **Docentes:**

-Benesch, Alicia (abenesch@campus.ungs.edu.ar)

-Ebertz, Ximena (xebertz@campus.ungs.edu.ar)

-Omar, Argañaras (omar.arganaras@campus.ungs.edu.ar)

- **Comisión:** 11
- **Turno:** Mañana
- **Ciclo lectivo:** 2°_semestre - 2022

●índice:

●Introducción:

El juego “la palabra escondida” consiste en adivinar una palabra de 5 caracteres al azar. El usuario tendrá 5 chances para ingresar la palabra y 60 segundos antes de que termine la partida. Si el usuario no acertara la palabra el juego terminará y no podrá acumular puntos, mientras que si el usuario adivinara la palabra consecutivamente, es decir partida tras partida, este acumulará puntos y el record más alto quedara en pantalla, hasta que un nuevo usuario bata ese record. Tendrá la oportunidad de seguir jugando o salir del programa (la pantalla emitirá carteles de aviso).

Reglas del juego:

-por cada palabra ingresada el juego da pistas:

-Las letras que no estén en la palabra permanecerán marcadas en el teclado en color (**Rojo**).

-Las letras que no estén en la palabra permanecerán marcadas en el teclado en color (**Amarillas**).

-Las letras que no estén en la palabra permanecerán marcadas en el teclado en color (**Verdes**).

-no se puede ingresar una palabra con más de 5 caracteres.

-la palabra debe estar en el lemario.

-no puede ingresar dos veces la misma palabra.

●**Desarrollo del juego:**

Para desarrollar el juego se implementaron las siguientes funciones:

-lectura: esta función toma tres parámetros (un archivo de texto, una lista vacía y una longitud) lee el archivo de texto y en salida guarda todas las palabras con la longitud del tercer parámetro.

```
def lectura(archivo, salida, largo):
    palabras = archivo.readlines();
    for palabra in palabras:
        if (len(palabra)-1)==largo:
            salida.append(palabra[:-1])
```

-**nuevaPalabra:** esta función toma como parámetro una lista y devuelve una palabra al azar.

```
def nuevaPalabra(lista):
    palabra_al_azar=random.choice(lista)
    palabra=palabra_al_azar.lower()# convierto la palabra a minuscula para evitar errores
    return palabra
```

-**revision:** esta función toma como parámetros una palabra elegida al azar en la función nuevaPalabra y una palabra ingresada por el usuario, luego compara ambas y guarda los caracteres correctas, incorrectas y parcialmente correctas, para finalizar retorna True si la palabra que ingreso el ususrio es igual a la palabra que retorna nuevaPalabra, y False si no lo es.

```
def revision(palabraCorrecta, palabra, correctas, incorrectas, casi):
        if palabraCorrecta==palabra:
            return True
        else:
            for i in range(len(palabra)):
                if palabra[i]==palabraCorrecta[i]:
                    correctas.append(palabra[i])
                elif palabra[i]not in palabraCorrecta:
                    incorrectas.append(palabra[i])
                elif palabra[i] in palabraCorrecta and palabra[i]!=palabraCorrecta[i]:
                    casi.append(palabra[i])
            return False
```

-**leerPuntaje: la función lee un archivo de texto y guarda dos puntajes el puntaje actual en (1) y el record o máximo puntaje en (2).**

```python
def leerPuntaje(opcion): #se creo la funcion leer puntaje de archivo. 1 devuelve record a

    if not os.path.exists(ARCHIVO_RECORD): # Crea archivo si no existe
        with open(ARCHIVO_RECORD, 'w') as file:
            file.write("recordActual = 0\nrecordMaximo = 0\n")

    with open(ARCHIVO_RECORD, "r") as file:
        data = file.readlines()
        recordActual, recordMaximo = [d.split('=')[1].split('\n')[0] for d in data]

    recordActual = int(recordActual)
    recordMaximo = int(recordMaximo)

    if opcion:
        if opcion== 1:
            return recordActual
        if opcion== 2:
            return recordMaximo
```

-actualizarRacha: esta función fue creada para actualizar los puntajes, este archivo se va re-escribiendo a medida que el record incremente.

```python
def actualizarRacha(gano): # Mantener puntaje y escribir en archivo
    recordActual = leerPuntaje(1)
    recordMaximo = leerPuntaje(2)

    #Gano = True, calcular puntaje. False, resetear puntaje
    if gano:
        recordActual+=1
        if recordActual > recordMaximo:
            recordMaximo = recordActual
    else:
        recordActual=0

    with open(ARCHIVO_RECORD, 'w') as file:
        file.write("recordActual = " + str(recordActual)+"\nrecordMaximo = " + str(recordMaximo))
```

●Adicionales:

-se configuraron los colores de las letras correctas, incorrectas y casi, también las del teclado.

-se creó ARCHIVO_RECORD para guardar el actual y mayor puntaje.

```
COLOR_LETRAS = (255,255,255)#color blanco para el teclado en pantalla
COLOR_FONDO = (0,0,0)
COLOR_TEXTO = (255,255,255)
COLOR_TIEMPO_FINAL = (200,20,10)
COLOR_AZUL = (0, 0, 255)
COLOR_VERDE=(0,255,0)#color de letras correctas
COLOR_ROJO=(255,0,0)#color de letras incorrectas
COLOR_AMARILLO=(200,200,0)#color de letras parcialmente correctas

ARCHIVO_RECORD = "record.txt" # EN ESTE ARCHIVO DE TEXTO SE GUARDA EL MAYOR PUNTAJE
```

-se agregaron los colores de las letras casi, correctas e incorrectas en el teclado.

```
#muestra las palabras anteriores, las que se fueron arriesgando
    pos =1
    for palabra in listaDePalabrasUsuario:
        screen.blit(defaultFontGrande.render(palabra, 1, COLOR_LETRAS), (ANCHO//2-len(palabra)*TAMANNO_LETRA_GRANDE//4,20 + 80 * pos))
        pos += 1

#muestra el abcdario
#el color del avecederario fue reconfigurado para que tenga el mismo color que el texto
    abcdario = ["qwertyuiop", "asdfghjkl", "zxcvbnm"]
    y=0
    for abc in abcdario:
        x = 0
        color=COLOR_LETRAS
        for letra in abc: # se agregaron los colores de las letras correctas, incorrectas y casi en la configuracion
            if letra in correctas:
                color = COLOR_VERDE
            elif letra in incorrectas:
                color=COLOR_ROJO
            elif letra in casi:
                color=COLOR_AMARILLO
            else:
                color=COLOR_LETRAS
            screen.blit(defaultFont.render(letra, 1, color), (10 + x, ALTO/1.5 + y))
            x += TAMANNO_LETRA
        y += TAMANNO_LETRA
elif gano: # se agrego musica de victoria cuando el jugador acierta

    pygame.mixer.music.load('Musica victoria.mp3')
    pygame.mixer.music.set_volume(0.08)
    pygame.mixer.music.play()
    actualizarRacha(True)
```

-se agrego musica de victoria cuando el jugador acierta y musica de derrota cuando el jugador pierde.

```
    screen.blit(defaultFontGrande.render("¡FELICIDADES, GANO!",1,COLOR_VERDE),(5,140))# cartel de ganador con la palabra correcta
    screen.blit(defaultFontGrande.render("LA PALABRA ES:",1,COLOR_VERDE),(100,220))
    screen.blit(defaultFontGrande.render("{"+palabraCorrecta.upper()+"}",1,COLOR_VERDE),(250,300))
    screen.blit(defaultFont.render("presione enter para continuar jugando, o [x] si desea salir del juego", 1, COLOR_TEXTO), (70,550))

    #MOSTRAR RACHA ACTUAL Y RACHA MAXIMA DE ARCHIVO
    screen.blit(defaultFont.render("Racha actual:"+str(leerPuntaje(1)),1,COLOR_TEXTO),(300,400))
    screen.blit(defaultFont.render("Record de racha:"+str(leerPuntaje(2)),1,COLOR_TEXTO),(300,450))


else: # se agrego musica de derrota cuando el jugador pierde
    pygame.mixer.music.load('Musica derrota.mp3')
    pygame.mixer.music.set_volume(0.08)
    pygame.mixer.music.play()
    actualizarRacha(False)

    screen.blit(defaultFontGrande.render("PERDIO, LA PALABRA",1,COLOR_ROJO),(10,140)) # cartel de perdedor con la palabra correcta
    screen.blit(defaultFontGrande.render("ES",1,COLOR_ROJO),(350,220))
    screen.blit(defaultFontGrande.render("{"+palabraCorrecta.upper()+"}",1,COLOR_ROJO),(250,300))
    screen.blit(defaultFont.render("presione enter para continuar jugando, o [x] si desea salir del juego", 1, COLOR_TEXTO), (70,550))#A

     #MOSTRAR RACHA ACTUAL Y RACHA MAXIMA DE ARCHIVO
    screen.blit(defaultFont.render("Racha actual:"+str(leerPuntaje(1)),1,COLOR_TEXTO),(300,400))
    screen.blit(defaultFont.render("Record de racha:"+str(leerPuntaje(2)),1,COLOR_TEXTO),(300,450))
```

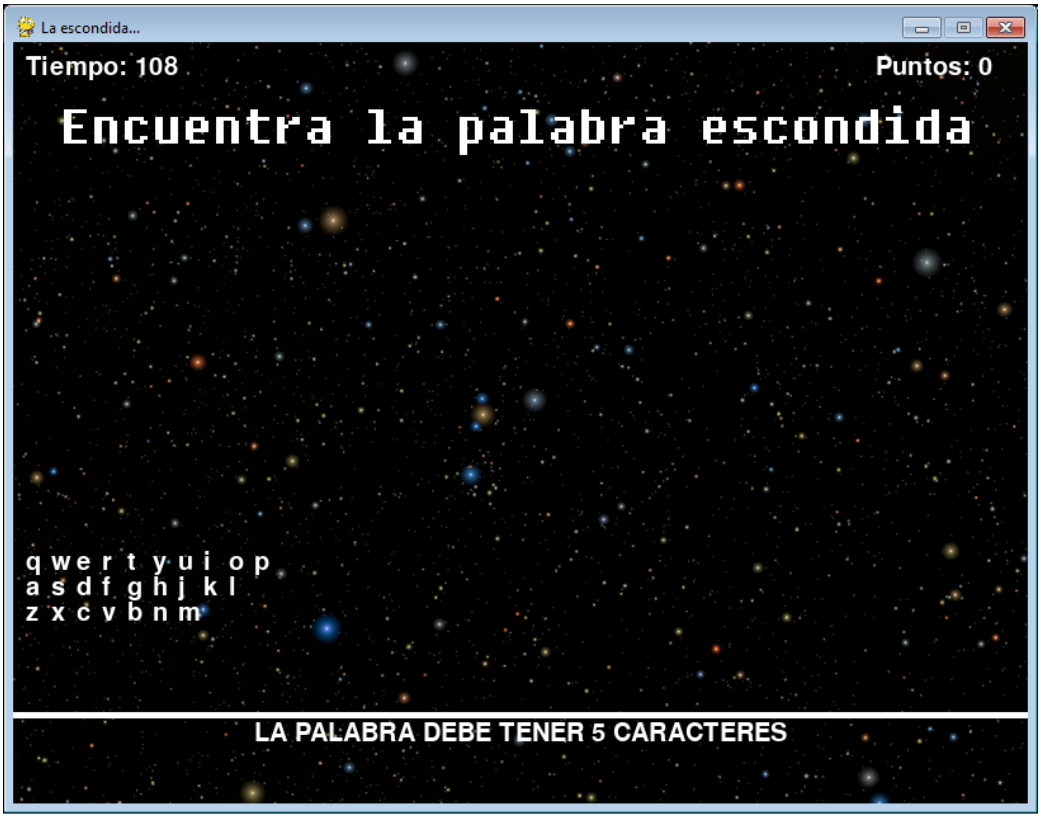
-se dibujó el cartel con el largo de la palabra, se dibujaron las reglas del juego cuando ingrese algo que las infrinja (mediante error se dibujan, para ello se modificó el programa principal) y los carteles de ganador y perdedor.

```
    #Verifica si la palabra ingresada por el usuario coincide con la longitud de la palabra correcta
    if len(palabraUsuario) == len(palabraCorrecta):
        #Verifica si la palabra ingresada está incluida en el diccionario
        enDiccionario = False
        for palabraDicc in listaPalabrasDiccionario:
            if palabraUsuario == palabraDicc:
                enDiccionario = True
        if enDiccionario == True:
            error = 0
            #Verifica si la palabra ingresada ya había sido ingresada anteriormente
            repetida = False
            for palabraRepe in ListaDePalabrasUsuario:
                if palabraUsuario == palabraRepe:
                    repetida = True
            if repetida == False:
                gano = revision(palabraCorrecta, palabraUsuario, correctas, incorrectas, casi)
                ListaDePalabrasUsuario.append(palabraUsuario)
                palabraUsuario = ""
                intentos -= 1

            else:
            #si la palabra está repetida, le avisa al usuario
                error = 3
                palabraUsuario = "" # vacio la palabra para que el usuario no tenga que borrar letra por letra en los tres condiciones de error
        else:
        #si la palabra no está en el diccionario, le avisa al usuario
            error = 2
            palabraUsuario = ""
    else:
    #si la palabra ingresada no tiene el largo que establece el juego, le avisa al usuario
        error = 1
        palabraUsuario = ""
```

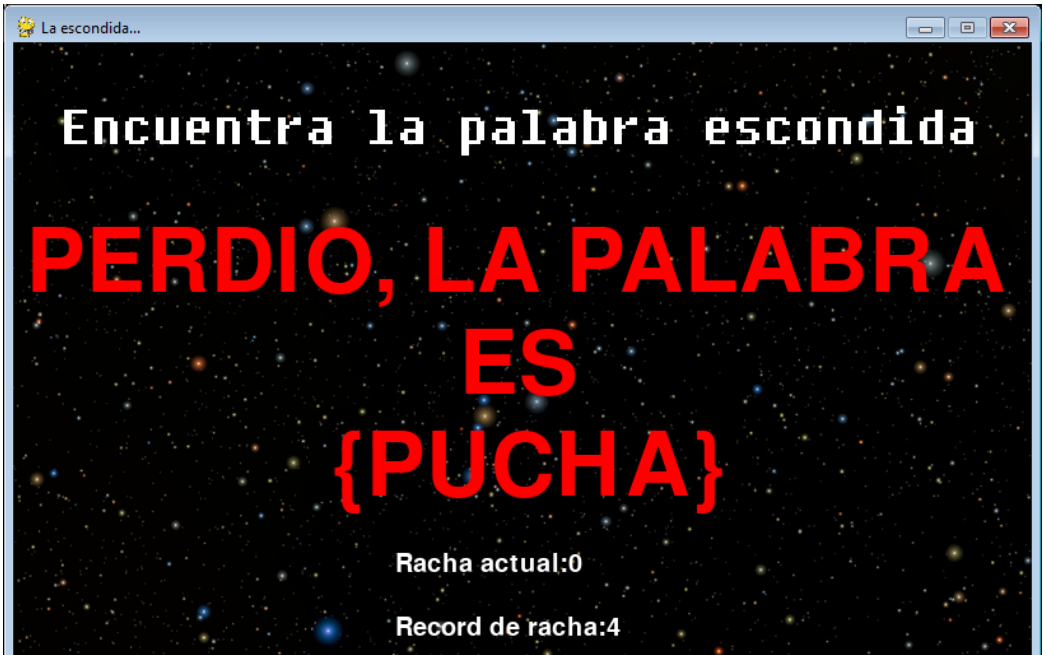
-se dibujó aviso para salir o para continuar con el juego.

```
def dibujar(screen, listaDePalabrasUsuario, palabraUsuario, puntos, segundos, gano,
                correctas, incorrectas, casi, error, intentos, palabraCorrecta):
    defaultFont= pygame.font.Font( pygame.font.get_default_font(), TAMANNO_LETRA)
    defaultFontGrande= pygame.font.Font( pygame.font.get_default_font(), TAMANNO_LETRA_GRANDE)

    fondo = pygame.image.load("fondo.png")
    fondo = pygame.transform.scale(fondo, (ANCHO, ALTO))
    screen.blit(fondo, (0, 0))

    #Linea Horizontal
    pygame.draw.line(screen, (255,255,255), (0, ALTO-70) , (ANCHO, ALTO-70), 5)
    if not gano and segundos>0 and intentos>0:
    #muestra lo que escribe el jugador
        screen.blit(defaultFont.render(palabraUsuario, 1, COLOR_TEXTO), (190, 570))
        screen.blit(defaultFont.render("LA PALABRA DEBE TENER "+str(LARGO)+" CARACTERES", 1, COLOR_TEXTO), (190, 535))#avisa c
    #muestra el puntaje
        screen.blit(defaultFont.render("Puntos: " + str(leerPuntaje(1)), 1, COLOR_TEXTO), (680, 10))
        #dibujo en pantalla los posibles errores
        if error == 1:
            screen.blit(defaultFont.render("La palabra no tiene " + str(LARGO) + " caracteres", 1, COLOR_TEXTO), (450, 580))
        elif error == 2:
            screen.blit(defaultFont.render("La palabra no está en el diccionario", 1, COLOR_TEXTO), (450, 580))
        elif error == 3:
            screen.blit(defaultFont.render("La palabra ya se había ingresado", 1, COLOR_TEXTO), (450, 580))
            #muestra los segundos y puede cambiar de color con el tiempo
        if(segundos<15):
            ren = defaultFont.render("Tiempo: " + str(int(segundos)), 1, COLOR_TIEMPO_FINAL)

        else:
            ren = defaultFont.render("Tiempo: " + str(int(segundos)), 1, COLOR_TEXTO)
        screen.blit(ren, (10, 10))
```

-se cambió el fondo de la pantalla.



-cartel perdedor con puntaje actual y record más palabra correcta y aviso de continuar o salir.



-cartel ganador con puntaje actual y record más palabra correcta y aviso de continuar o salir.