Trabajo Practico Programacion

Introducción:

Este trabajo se trata de un juego, en el que se muestra en pantalla una palabra (que se toma de un archivo), y el usuario debe escribirla en el pie de la pantalla, separando en sílabas.

Para ello hemos desarrollado un conjunto de funciones que permiten controlar los datos y reglas del juego.

Integrantes del grupo:

- Albarenga, Ignacio Agustín.
- Pedraza, Joaquin Ezequiel.
- Rubio, Martín Oscar.

Desarrollo:

Hemos desarrollado las siguientes funciones básicas para el correcto funcionamiento del juego:

1. lectura.
2. nuevaPalabra.
3. silabasTOpalabra.
4. palabraTOsilaba.
5. esCorrecta.
6. puntaje.

Explicación de cada función:

1. ***lectura(archivo, salida)*** sirve para poder leer el archivo y devolver una lista con todo su contenido adentro, a esta función le pasamos 2 variables, **archivo**, para que lea el archivo donde se encuentran las palabras en las que se basa el juego, y **salida**, para que guarde cada palabra en una lista, para luego, poder elegir al azar una palabra entre toda esa lista.

```
def lectura(archivo, salida):
    with archivo as txt:
        for lineas in txt:
            salida.extend(lineas.split())
```

2. ***nuevaPalabra(lista)*** tiene como función elegir una palabra al azar de una lista, a esta función le pasamos la **lista** para que mediante un tipo de sorteo, elija una palabra al azar.

```
def nuevaPalabra(lista):
    azar = random.choice(lista)
    while "ñ" in azar or len(azar)<1:
        azar = random.choice(lista)
    return azar
```

3. ***silabasTOpalabra(silaba)*** esta función lo que hace es recorrer la palabra separada en silabas con un guion “-” de por medio, y crea una nueva palabra separada en silabas con espacio, intercambiando el guión por un espacio.

```
def silabasTOpalabra(silaba):
    palabraSinGuiones = ""
    for caracter in silaba:
        if caracter == "-":
            palabraSinGuiones += " "
        else:
            palabraSinGuiones += caracter
    return palabraSinGuiones
```

4. ***palabraTOsilaba(palabra)*** en esta función utilizamos la función hecha por los profesores llamada “separador” la cual toma como parámetro la palabra a separar en silabas.

```
def palabraTOsilaba(palabra):
    palabraSeparadaEnSilabas = separador(palabra)
    return palabraSeparadaEnSilabas
```

5. ***esCorrecta(palabraEnSilabasEnPantalla, palabra)*** devuelve si la palabra ingresada está separada en silabas correctamente o no, a esta función le pasamos 2 variables, **palabraEnSilabasEnPantalla,** que es la palabra que ingresa el usuario y **palabra**, que es la palabra original. Lo que hace esta función es que a cada variable la manda a la funcion silabasTOpalabra, para poder dejarlas en igualdad de condiciones, después mediante el condicional if, preguntamos si son iguales, en el caso de que sean iguales devuelve True, y en el caso contrario devuelve False.

```
def esCorrecta(palabraEnSilabasEnPantalla, palabra):
    palabraSinAgregados = silabasTOpalabra(palabraEnSilabasEnPantalla).lower()
    palabraOriginalSinAgregrados = silabasTOpalabra(palabra).lower()
    if palabraSinAgregados == palabraOriginalSinAgregrados:
        return True
    return False
```

6. ***puntaje(palabra)*** devuelve el puntaje de una palabra, a esta función le pasamos **palabra**, que sería la palabra que ingresa el usuario, y lo que hace es preguntar si el largo de la palabra es mayor a 6 devuelve True, y en caso contrario False, esto lo utilizamos para sumar más puntos o no, todo desde el programa principal.

```
def puntaje(palabra):
    if len(palabra)>6:
        return True
    return False
```

Dificultades en el desarrollo de las funciones básicas:

1. En la función **lectura(archivo, salida)** solo funcionaba con el archivo "lemario.txt", para solucionar esto cambiamos el parámetro *"lemario.txt"* por *archivo* y le eliminamos el return al final ya que no era necesario que retorne ninguna salida.

2. En la función **nuevaPalabra(lista)** habíamos llamado a la función *lectura* pero no era necesario ya que de eso se encargaba el programa principal, así que le eliminamos esa llamada.

3. En la función **silabasTOpalabra(silaba)** tuvimos que modificar la forma de recorrer la variable ingresada ya que había un error de que si escribías la palabra sin espacios te lo daba como correcto. Lo solucionamos modificando el ciclo en donde se recorria caracter por caracter de la silaba ingresada, y cuando detectaba un guion, lo reemplaze por un espacio y lo guarde en una variable nueva llamada “palabraSinGuiones”, en el caso de que haya una letra la guardaba en esas misma variable.

4. En la función **esCorrecta(palabraEnSilabasEnPantalla, palabra)** al estar mal la función *silabasTOpalabra* estaba mal esta función y daba por correcta cualquier palabra ingresada que cumpla con el largo de la palabra original, una vez solventado el problema de silabasTOpalabra se soluciono el error en esta función.

Extras:

1. Fondo del Juego:
    a. Descargamos una imagen apta para la resolucion del juego y lo guardamos en una carpeta destinada para las imagenes.
    b. Luego en el archivo principal llamamos a la funcion de pygame (pygame.image.load("imagenes/background.jpg") y cargamos el fondo.

2. Sonidos:
    a. Al igual que con el fondo del juego, creamos una carpeta destinada a los sonidos, y le incluimos los sonidos de: Correcto, Incorrecto, Ganaste y Perdiste.
    b. Luego en el archivo principal llamamos a la funcion de pygame (pygame.mixer.Sound(“sonidos/...”)) dependiendo el estado del juego,por ejemplo (pygame.mixer.Sound(“sonidos/correcto.wav”)) si acertaste la silaba, (pygame.mixer.Sound(“sonidos/incorrecto.wav”)) si no acertaste la silaba, etc.

3. Cambio de color de la fuente de Tiempo y Puntos:
    a. Cuando el tiempo sea mayor a 30 segundos sera verde, cuando sea de 15 a 30 segundos sera amarillo y menor a 15 sera rojo. Mediante el uso de condicionales comparamos el estado del temporizador y le modificamos el color con un por ejemplo, si el tiempo es menor de 30 y mayor que 15 ponemos, ren = defaultFont.render("Tiempo: " + str(int(segundos)), 1, COLOR_YELLOW)
    b. Los puntos cuando es 1 y mayor a 1 sera verde, si el punto es 0, sera blanco y menor a 0 rojo. Por ejemplo si el punto es 0 ponemos screen.blit(defaultFont.render("Puntos: " + str(puntos), 1, COLOR_WHITE), (1138, 10))

4. Menu principal:
    a. Buscamos un menu en internet y lo adaptamos a nuestro juego, modificando el tipo de fuente, fondo, funcionalidad de cada opcion que traia (opciones y jugar)
    b. En el apartado JUGAR el usuario puede jugar con casi todas las palabras del lemario, menos con las de 1 caracter y las que tienen la letra “ñ”.
    c. Modificamos las opciones agregandole poder cambiar la dificultad del juego, donde el usuario puede elegir entre FACIL, MEDIO y DIFICIL.
    d. El apartado SALIR se encarga de cerrar la ventana.
    e. En las OPCIONES se encuentra el boton VOLVER que sirve para volver al menu principal.
    f. Le agregamos un icono, llamando a la funcion de pygame (pygame.display.set_icon(icono))

5. Modos de juego:
    a. En el apartado opciones del Menu principal, se encuentran las diferentes dificultades del juego.
    b. Cada una de estas variaciones cambian el maximo de caracteres de palabra y en el caso de MEDIO le agregamos un minimo y un maximo, es decir en dificultad MEDIO tiene como minimo 5 y de maximo 9 caracteres. En la dificultad FACIL tiene como maximo 5 caracteres y en DIFICIL un minimo de 10.
    c. Al igual que en la opcion JUGAR, ninguna variacion de dificultad se incluyen las palabras con la letra “ñ”.
    d. Para que el juego funcione tuvimos que crear nuevas funciones dependiendo de la dificultad.

Explicación de las funciones extras:

1. **Fondo**: Definimos la variable background, y le pasamos la ruta en donde se encuentra el fondo.

```
background = pygame.image.load("imagenes/fondo.jpg")
```

2. **Sonidos**: Definimos cada variable para un sonido distinto, pasandole la ruta en donde se encuentra cada sonido.

```
sonidoCorrecto = pygame.mixer.Sound("sonidos/correcto.wav")
sonidoIncorrecto = pygame.mixer.Sound("sonidos/incorrecto.wav")
sonidoGanaste = pygame.mixer.Sound("sonidos/ganasteJuego.wav")
sonidoPerdiste = pygame.mixer.Sound("sonidos/perdisteJuego.wav")
pygame.mixer.music.set_volume(1.0)
```

3. **Cambio de color de la fuente de Tiempo y Puntos**: Tal como esta expresado en lineas arriba, asi quedaria en codigo.

```
if(puntos>=1):
    screen.blit(defaultFont.render("Puntos: " + str(puntos), 1, COLOR_GREEN), (1138, 10))
if(puntos==0):
    screen.blit(defaultFont.render("Puntos: " + str(puntos), 1, COLOR_WHITE), (1138, 10))
if(puntos<0):
    screen.blit(defaultFont.render("Puntos: " + str(puntos), 1, COLOR_RED), (1138, 10))
# ---- MUESTRA LOS SEGUNDOS Y CON LA APROXIMIDADA AL 0 CAMBIA AL ROJO. ---- #
if(segundos<30):
    ren = defaultFont.render("Tiempo: " + str(int(segundos)), 1, COLOR_YELLOW)
    if(segundos<15):
        ren = defaultFont.render("Tiempo: " + str(int(segundos)), 1, COLOR_RED)
else:
    ren = defaultFont.render("Tiempo: " + str(int(segundos)), 1, COLOR_GREEN)
screen.blit(ren, (10, 10))
```

4. **Menu principal**:

- PLAY: Se encarga de redireccionar al juego sin dificultad.

```
def play():
    while True:
        PLAY_MOUSE_POS = pygame.mouse.get_pos()

        SCREEN.fill(juegoOriginal())

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if PLAY_BACK.checkForInput(PLAY_MOUSE_POS):
                    main_menu()
        pygame.display.update()
```

- OPTIONS: Se encarga de mostrar las diferentes dificultades del juego y redirecciona a la dificultad elegida.

```
def options():
    while True:
        OPTIONS_MOUSE_POS = pygame.mouse.get_pos()

        SCREEN.blit(BG, (0, 0))

        OPTIONS_DIFICULTAD = Button(image=None, pos=(617, 100),
                            text_input="DIFICULTAD", font=get_font(80), base_color="White", hovering_color="Black")
        OPTIONS_DIFICULTAD.update(SCREEN)

        OPTIONS_DIFICULTAD = Button(image=None, pos=(619, 101),
                            text_input="DIFICULTAD", font=get_font(80), base_color="Black", hovering_color="Black")
        OPTIONS_DIFICULTAD.update(SCREEN)

        OPTIONS_FACIL = Button(image=None, pos=(602, 220),
                            text_input="FACIL", font=get_font(80), base_color="White", hovering_color="Black")
        OPTIONS_FACIL.update(SCREEN)

        OPTIONS_MEDIO = Button(image=None, pos=(626, 322),
                            text_input="MEDIO", font=get_font(80), base_color="White", hovering_color="Black")
        OPTIONS_MEDIO.update(SCREEN)

        OPTIONS_DIFICIL = Button(image=None, pos=(629, 424),
                            text_input="DIFICIL", font=get_font(80), base_color="White", hovering_color="Black")
        OPTIONS_DIFICIL.update(SCREEN)

        OPTIONS_VOLVER= Button(image=None, pos=(618, 570),
                            text_input="VOLVER", font=get_font(75), base_color="White", hovering_color="Green")
        OPTIONS_VOLVER.update(SCREEN)

        OPTIONS_FACIL = Button(image=None, pos=(600, 220),
                            text_input="FACIL", font=get_font(80), base_color="Green", hovering_color="Black")
        OPTIONS_FACIL.update(SCREEN)

        OPTIONS_MEDIO = Button(image=None, pos=(622, 322),
                            text_input="MEDIO", font=get_font(80), base_color="Yellow", hovering_color="Black")
        OPTIONS_MEDIO.update(SCREEN)

        OPTIONS_DIFICIL = Button(image=None, pos=(627, 424),
                            text_input="DIFICIL", font=get_font(80), base_color="Red", hovering_color="Black")
        OPTIONS_DIFICIL.update(SCREEN)

        OPTIONS_VOLVER= Button(image=None, pos=(620, 570),
                            text_input="VOLVER", font=get_font(75), base_color="Black", hovering_color="Green")
        OPTIONS_VOLVER.update(SCREEN)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if OPTIONS_VOLVER.checkForInput(OPTIONS_MOUSE_POS):
                    main_menu()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if OPTIONS_FACIL.checkForInput(OPTIONS_MOUSE_POS):
                    juegoFacil()
                if OPTIONS_MEDIO.checkForInput(OPTIONS_MOUSE_POS):
                    juegoMedio()
                if OPTIONS_DIFICIL.checkForInput(OPTIONS_MOUSE_POS):
                    juegoDificil()
        pygame.display.update()
```

- Menu principal: Es el encargado de mostrar todos los botones (JUGAR, OPCIONES y SALIR).

```
def main_menu():
    while True:
        icono = pygame.image.load("imagenes/icon.png")
        pygame.display.set_icon(icono)

        SCREEN.blit(BG, (0, 0))

        MENU_MOUSE_POS = pygame.mouse.get_pos()

        MENU_TEXT = get_font(100).render("MENU PRINCIPAL", True, "#000000")
        MENU_RECT = MENU_TEXT.get_rect(center=(640, 100))

        PLAY_BUTTON = Button(image=pygame.image.load("assets/Play Rect.png"), pos=(640, 250),
                             text_input="JUGAR", font=get_font(75), base_color="#d7fcd4", hovering_color="White")
        OPTIONS_BUTTON = Button(image=pygame.image.load("assets/Options Rect.png"), pos=(640, 400),
                             text_input="OPCIONES", font=get_font(75), base_color="#d7fcd4", hovering_color="White")
        QUIT_BUTTON = Button(image=pygame.image.load("assets/Quit Rect.png"), pos=(640, 550),
                             text_input="SALIR", font=get_font(75), base_color="#d7fcd4", hovering_color="White")

        SCREEN.blit(MENU_TEXT, MENU_RECT)

        for button in [PLAY_BUTTON, OPTIONS_BUTTON, QUIT_BUTTON]:
            button.changeColor(MENU_MOUSE_POS)
            button.update(SCREEN)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if PLAY_BUTTON.checkForInput(MENU_MOUSE_POS):
                    play()
                if OPTIONS_BUTTON.checkForInput(MENU_MOUSE_POS):
                    options()
                if QUIT_BUTTON.checkForInput(MENU_MOUSE_POS):
                    pygame.quit()
                    sys.exit()
        pygame.display.update()
```

5. **Modos de juego:** Aqui se encuentran todas las dificultades (FACIL, MEDIO y DIFICIL). Tal como esta dicho en su apartado en lineas anteriores, asi quedaria expresado en codigo.

```
def facil(lista):
    azar = random.choice(lista)
    while "ñ" in azar or len(azar)>5:
        azar = random.choice(lista)
    return azar

def medio(lista):
    azar = random.choice(lista)
    while "ñ" in azar or (len(azar)>9 or len(azar)<5):
        azar = random.choice(lista)
    return azar

def dificil(lista):
    azar = random.choice(lista)
    while "ñ" in azar or (len(azar)<10):
        azar = random.choice(lista)
    return azar
```

Dificultades en el desarrollo de las funciones extras:

- Menu principal:
  - o Al principio tuvimos problemas con la adaptación del codigo ya que no nos redirigia bien a cada opcion elegida, para eso tuvimos que crear funciones especiales para cada eleccion (juegoOriginal, juegoFacil,juegoMedio, y juegoDificil).
  - o Otro problema que nos surgio fue el temporizador, ya que empezaba a contar desde que iniciabas el juego, es decir que, desde el menu principal ya empezaba a descontar segundos, para esto creamos una variable nueva que sea otro temporizador y luego se lo sumabamos al tiempo total para que le sume la diferencia que serian los segundos perdidos.
- Modos de juego:
  - o Tuvimos un problema general en todas las dificultades, ya que dejaban pasar las palabras con la letra “ñ” y no cumplian la orden del largo de la palabra. Para esto creamos un bucle while, que recorre la palabra para fijarse si contiene la letra “ñ” y s no es el largo pedido, genere otra palabra al azar.