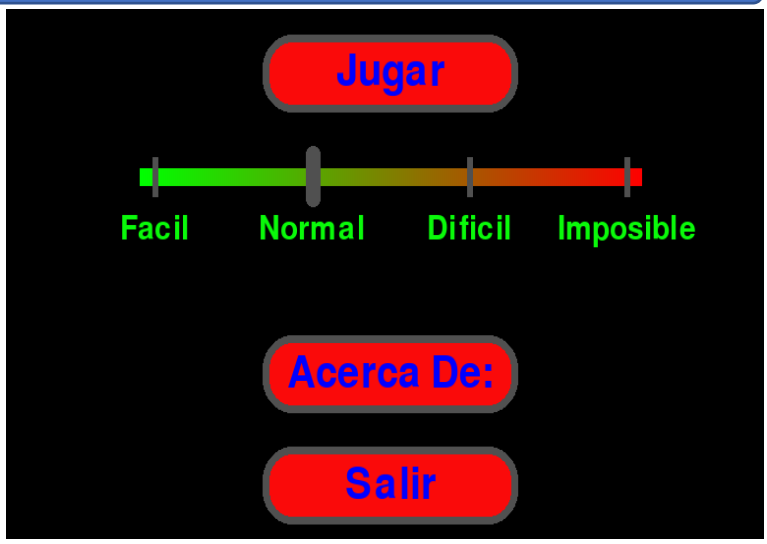
JUEGO - LA ESCONDIDA - PANTALLA DE INICIO.

El objetivo del trabajo practico final, es generar un juego que nos permita adivinar la palabra que el módulo selecciona aleatoriamente.
Para comenzar le vamos a pedir al usuario que seleccione el nivel de dificultad con el que desea realizar la partida.
Una vez seleccionado va a poder presionar el botón jugar y comenzar la partida.
El participante va a contar con (cinco) intentos o una cantidad de tiempo que va a depender del nivel de dificultad seleccionado.
En el botón acerca de... Va a poder encontrar la explicación de cómo se realizó el juego.



La siguiente función se desarrolló en la hoja "extras". Para lograr esta pantalla inicial, se utilizó la función "draw" del módulo PyGame, que nos permite utilizar diferentes complementos (cuadrados, rectángulos, círculos, etc).
La sintaxis se compone de la siguiente manera:

```
#rectangulo Jugar
REC3 = pygame.Rect(REC3_X1,REC3_Y1,REC3_X12,REC3_Y12)
pygame.draw.rect(screen,COLOR_ROJO,REC3,0,3
pygame.draw.rect(screen,COLOR_GRIS_OSCURO,REC37,30)
```

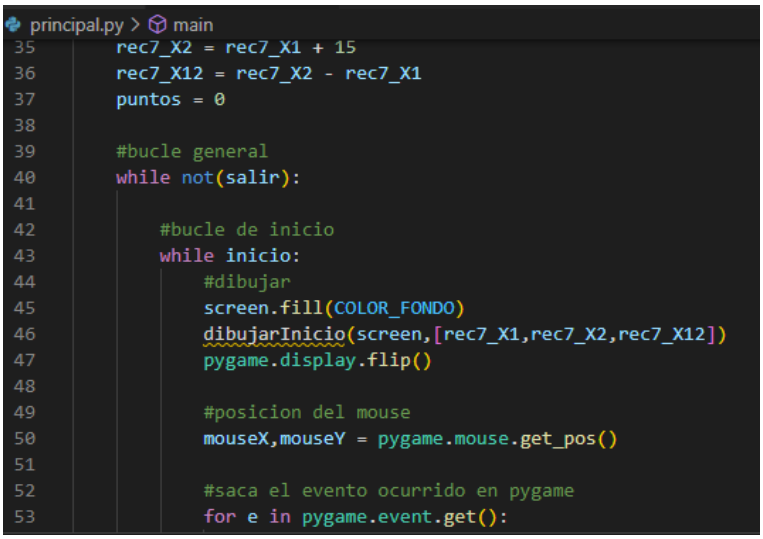
Dentro de la misma hoja "extras" se confecciono la línea de dificultad. Para que esta línea funcione, se le agrego un if que permite ver entre 1 y 4 que dificultad selecciono el usuario. Parte del código es:

```
#separadores de dificultad
cont = 0
   for LINEA_POS_X in LINEAS_DIFICULTAD_X:
      if cont == 0:
         dificultad = "Facil"
      elif cont == 1:
         dificultad = "Normal"
pygame.draw.line(screen,COLOR_GRIS_OSCURO,(LINEA_POS_X,REC6_Y1-10),(LINEA_POS_X,REC6_Y2+10),6)
screen.blit(EtiquetaDificultad,(LINEA_POS_X-EtiquetaDificultad.get_width()//2,REC6_Y1+40))
```

En la hoja "extra" podemos encontrar varias funciones vitales para el juego, principalmente, todo lo que refiere al motor grafico de este. Por ejemplo:

- Color de las letras que el usuario adivino o fallo.
- Longitud de la palabra ingresada.
- Segundos que pasaron desde que comenzó la partida.
- Muestra el abecedario.
- Detecta los tipos de errores introducidos, longitud de la palabra, palabra desconocida, ya ingresada.

- ¿Falta algo?



Función Principal. Esta función tiene la mayoría de los procesos que se van a utilizar durante el juego. Lo principal que podemos destacar, antes de comenzar a desarrollar el contenido, son los bucles que va a realizar el programa. Dentro de estos tenemos 5 niveles de bucle y el cierre de la partida.

Bucle general.

Bucle inicio.

Reiniciar valores.

Bucle del Juego.

Buble Final.

Cierre del juego.

Para comenzar el desarrollo de la hoja principal tenemos que comprender que es el más largo de todo el juego, por lo tanto, va a llevar mayor tiempo de explicación.

- **Bucle general**. Este va a permitir, a través de un while, que el juego no salga mientras se cumple la condición de no salir. Esta tabulación solo repite el nivel al hacer el cierre del juego.
- **Bucle Inicio**. En esta parte se realiza todo el proceso de inicio del motor gráfico. A partir de este código se inicia todo el background con los botones, movimientos del mouse y acciones que se realizan al seleccionar una acción.
- **Bucle reiniciar valores**. Este módulo funciona siempre para el inicio del juego, como los reintentos. Es decir, prepara la partida de acuerdo al nivel de dificultad seleccionado. Por ejemplo, si el usuario selecciona un nivel "fácil", el juego le va a proporcionar más tiempo que en un nivel difícil. La misma dificultad, actúa sobre el largo de las palabras que ingresan en el random.
    - Es importante recordar que en este bucle contamos con los valores básicos que puede alcanzar el juego. Por ejemplo, la para que selecciona el usuario, las que están en el diccionario, cuales son correctas, incorrectas, tienen algunas coincidencias, errores, los intentos que ya acumulo el participante y casi el más importante, sino no funcionaría nada, **la importación del "lemario"**.
    - Las sintaxis utilizadas en este módulo, son principalmente, while; for i in; if; elif; comandos pygame; listas.
- **Bucle del juego**. El bucle del juego contiene una de las condiciones más importantes, a continuación, mostramos la sintaxis de esta:

```
while segundos > fps/1000 and intentos > 0 and not gano:
    # 1 frame cada 1/fps segundos
    gameClock.tick(fps)
    totaltime += gameClock.get_time()
```

Es fundamental para el juego saber si el participante no cuenta con más tiempo, gano o agoto todos los intentos disponibles. En caso de no cumplir las tres condiciones simultáneamente, significa que perdió.

    - Dentro de esta función, se agregaron los efectos de sonido, en caso de perder o ganar. En ambos casos, se realiza un reset para que el juego pueda volver a comenzar "limpio".
- **Bucle Final**. En esta instancia se toma la respuesta de lo que sucedió en el juego, se guarda el resultado y se le da la opción al usuario de reiniciar una nueva partida, salir o volver al menú principal.
- **Cierre del Juego**. Vale aclarar que se incorporó una opción para que, si el usuario presiona la tecla salir en cualquier momento de la partida, independientemente de lo que suceda en la partida. Es decir, quizá mientras la partida está en curso, simplemente no desea continuar.

Hoja funciones vacías. Además de la función principal, en tema de importancia, tenemos la hoja de funciones vacías. Dentro de ella, se encuentran todas las interacciones que va a recibir el programa en el momento del juego.
Esta hoja es de vital importancia, ya que tiene la función de selección de palabra para iniciar la partida utilizando un random, el cual va a interactuar con la dificultad seleccionada por el usuario y de acuerdo a eso va a seleccionar, al azar, la palabra a adivinar.


```
def nuevaPalabra(lista,largo=0):
    numero=random.randrange(len(lista))
    if largo >0:
        while len(lista[numero]) != largo:
            numero=random.randrange(len(lista))
    return lista[numero]
```

A continuación, tenemos una función fundamental, que es la que recibe la palabra ingresada e interpreta el resultado. Es decir, si la palabra es correcta, incorrecta o si tiene aciertos parciales. Por ejemplo, si el usuario no encontró la palabra correcta, pero algunas letras coinciden suceden dos eventos. Si las letras coinciden y están en la posición correcta, se pintan de color verde, en caso de que estén en la palabra, pero en otra posición de color amarillo.

```
def revision(palabraCorrecta, palabra, correctas, incorrectas, casi):
    cont=0
    for i in palabra:
        if i in palabraCorrecta:
            if palabraCorrecta[cont]==palabra[cont] and i not in correctas:
                correctas.append(i)
            elif i not in correctas and i not in casi:
                casi.append(i)
        elif i not in incorrectas:
            incorrectas.append(i)
        cont+=1
```

```
def cambiarColorLetra(letra,correctas,casi,incorrectas):
    if letra in correctas:
        return COLOR_VERDE
    elif letra in casi:
        return COLOR_AMARILLO
    elif letra in incorrectas:
        return COLOR_ROJO
    else:
        return COLOR_TEXTO
```

```
#crear funcion puntajes para guardar
def guardar_puntajes(puntajes):
    archivo = open("resultados.cvs", "a")
    for nombre, puntos,  in puntajes:
        archivo.write(nombre+","+str(puntos)+"\n")
    archivo.close()

def recuperar_puntajes(nombre_archivo):
    lista = []
    archivo = open(nombre_archivo, "r")
    for línea in archivo:
        nombre, puntos = línea.rstrip("\\n").spl
        lista.append((nombre,int(puntos)))
    archivo.close()
    return lista
```

Hoja guardarecord. La hoja que mencionamos a continuación trabaja en conjunto con resultados.txt. El objetivo de este punto, es poder registrar los puntajes de los diferentes jugadoresy guardar las partidas personales a través de los puntajes conseguidos. Por último, vamos a poder visitar los diferentes puntajes, ordenados de mayor a menor.

Hoja configuración. En la siguiente hoja podemos principalmente mencionar que se cargaron todas las fuentes, colores y dibujos (rectángulos, etc).

```
#rectangulo SalirInicio
REC4_X1 = REC3_X1
REC4_X2 = REC3_X2
REC4_X12 = REC4_X2-REC4_X1
REC4_Y1 = CENTRO_PANTALLA_Y+100
REC4_Y2 = REC4_Y1 + 70
REC4_Y12 = REC4_Y2-REC4_Y1

#rectangulo AcercaDe:
REC5_X1 = REC3_X1
REC5_X2 = REC3_X2
REC5_X12 = REC5_X2-REC5_X1
REC5_Y1 = CENTRO_PANTALLA_Y
REC5_Y2 = REC5_Y1 + 50
REC5_Y12 = REC4_Y2-REC4_Y1
```

Conclusión.

En informe final tiene como objetivo principal identificar los puntos más importantes del código y sobre todo explicar su funcionamiento. Si bien se utilizaron funciones y fórmulas que quizás no se habían usado previamente, a través de investigación y puesta en marcha pudimos lograr mayor versatilidad en el funcionamiento del código.

Como conclusión podemos estar seguros que el trabajo nos permitió poner en práctica, todos los conceptos aprendidos durante la cursada y algunos puntos más. Por último, fue muy enriquecedor el trabajo en equipo a fin de poder absorber conceptos, brindarnos soporte y compartir conocimientos que otros compañeros tenían más afianzados.