Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2019-2

26 de septiembre de 2019
Actividad Sumativa

Actividad 05

Interfaces Gráficas

Introducción

El pasado 20 de septiembre, estando en medio de una fonda, recordaste que era el tan esperado *Area 51 Raid*, así que hiciste tu mejor *Naruto Run*[1] para llegar al instante. Luego de una intensa batalla lograste escapar con una caja secreta, la cual en su interior contenía a un *alien* que necesita tu ayuda para escapar de la tierra. La clave para abrirla está oculta en el famoso juego **DCColgado 51**, y tú, como programador experto, deberás modelar este juego usando interfaces gráficas y encontrar su solución.

Modo de juego

El objetivo del *DCColgado 51* es simple: adivinar, letra por letra, una palabra secreta antes de que el colgado desaparezca. Cada letra que confirmes será considerada como un **intento**, y al alcanzar **seis intentos erróneos** (que no están en la palabra) perderás la partida. El usuario debe poder:

- Presionar una letra en el teclado, que aparecerá en pantalla.
- Confirmar la última tecla presionada como un **intento** para ingresar. Esta confirmación debe hacerse mediante algún *widget*, como un botón.
- Una vez que se ingresa un intento (letra) al juego, se debe actualizar un historial visible de letras ya intentadas. Del mismo modo, se deben indicar las letras que aún se pueden ingresar.

[1] Más información aquí.

- Si el intento ingresado estaba en la palabra por adivinar, se actualiza la palabra con la la letra. Si fue un intento erroneo, se debe actualizar una imagen del *alien* en una fase más cercana a la derrota.

Archivos Entregados

Para el desarrollo de esta actividad, tendrás a tu disposición el *back-end* **ya implementado** y tu **solo debes preocuparte de crear el *front-end* del programa**. Junto con esto, recibirás un pequeño esqueleto para que puedas instanciar tu programa. Los detalles de cada uno se entregan a continuación:

- ***Back-end:***
  - `DCColgado.py`: Corresponde al funcionamiento interno del juego y **ya está implementado**. Puedes probar el juego en la terminal ejecutando directamente este archivo.

    El archivo contiene la clase `DCColgado`, la cual posee la señal `respuesta_signal` que se encargará de **emitir** la información necesaria al *front-end* mediante un diccionario. Este diccionario, tendrá las siguientes llaves y valores:

    - `"msg"`: Mensaje (*string*) que indica un estado respecto a la consulta de letra o palabra.

      Ejemplo: `"Esta letra ya fue utilizada"`

    - `"usadas"`: Es un *string* con todas las letras que ya han sido utilizadas.

      Ejemplo: `"AGDPWR"`

    - `"disponibles"`: Es un *string* que indica las letras que aún pueden ser utilizadas.

      Ejemplo: `" BC EF HIJKLMNÑO Q STUV XYZ"`

    - `"palabra"`: Este *string* muestra la palabra en su estado actual, es decir, las letras ocultas y las visibles.

      Ejemplo: `"P A _ A _ R A"`

    - `"imagen"`: *Path* relativo (*string*) de la imagen del colgado que debe mostrarse.

      Ejemplo: `"images/1.png"`

    Para conectar la señal `respuesta_signal` con un método de la ventana que sea capaz de manejar esta información, debes completar el archivo `main.py` en la **línea 28**.
  - `main.py`: Corresponde al módulo que instancia `DCColgado` y `VentanaJuego`, que además realiza las conexiones de señales entre ellos.
- ***Front-end:***
  - `Ventanas.py`: Corresponde a un módulo semi vacío. Aquí eres libre de diseñar tu código como mejor te convenga, procurando que contenga lo pedido en **Modo de juego** y lo indicado más abajo:

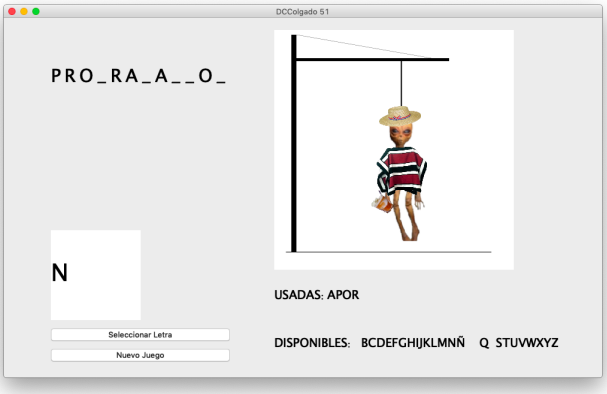

Figura 1: Ejemplo de ventana de juego

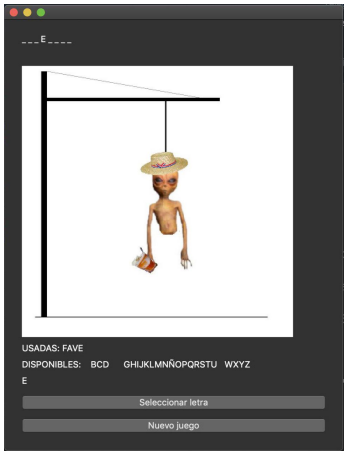

Figura 2: Otro ejemplo de ventana de juego

Como puedes ver, el juego debe estar compuesto por lo siguiente:

- La palabra completándose a medida que se ingresen las letras correctas.
- Una zona para la imagen del personaje.
- Una zona que muestre las letras ya usadas.
- Una zona que muestre las letras disponibles.
- Una sección para mostar un mensaje de ser necesario.
- Una sección que muestre la letra presionada que se pretende seleccionar.
- Un botón para confirmar la letra y que sea enviada al *back-end*.
- Un botón para iniciar un nuevo juego.

IMPORTANTE: El archivo `Ventanas.py` ya trae declarada la clase `VentanaJuego`, la cual contine dos señales:

1) `enviar_letra_signal`: Una señal que envía al *back-end* la información correspondiente a la letra seleccionada o la palabra completa en caso del *bonus* mediante un diccionario. Este diccionario debe contener al menos la *key* `'letra'` donde el valor asociado será la letra seleccionada.

2) `reiniciar_signal`: Una señal que no recibe argumentos, la cual solo indica al *back-end* que debe buscar otra palabra y volver los atributos como los intentos o las letras utilizadas a sus valores iniciales.

Cada vez que una de estas señales sea activadas, el *back-end* enviará una respuesta mediante la señal `respuesta_signal` mencionada anteriormente.

- `images/`: Corresponde a el directorio donde se encuentran las imágenes que deberás usar para mostrar los intentos restantes en el juego.

Notas

- **Sólo se evaluará lo que sea visible a nivel de interfaz**, por lo que debes mostrar todo lo necesario en ella y no en la consola.

- Recuerda implementar la comunicación *back-front* mediante el uso de **señales**.
- Considera que en tu *front-end* no puede existir contenido relacionado al funcionamiento del juego, por lo que está prohibido implementar cosas relacionadas a este en dicho módulo.

Requerimientos

- (3 pts) Interfaz
  - (1 pts) Se visualiza una ventana con todos los elementos correspondientes.
  - (1 pts) Se agrega una letra al historial de letras usadas y se elimina de las disponibles.
  - (1 pts) Se muestran los intentos restantes mediante la actualización de la imagen del personaje.
- (2 pts) Botones y teclas
  - (1 pt) Se visualiza la tecla presionada en la interfaz al presionarla.
  - (0.5 pts) Se puede enviar la letra presionada al *back-end* mediante un botón.
  - (0.5 pts) Se puede iniciar una nueva partida mediante un botón.
- (1 pt) Señales
  - (1 pt) Se implementa el uso de señales y son manejadas correctamente en el *front-end*
- (0.5) *Bonus*
  - (0.3 pts) Se implementa el *pop-up* de Ingresar palabra completa.
  - (0.2 pts) Se implementa el *pop-up* de Victoria y Derrota.

Bonus: Pop-ups (5 décimas)

Esta actividad considera 2 *bonus* distintos, para los cuales **está permitido editar el *back-end*** solo agregando funcionalidades. Para conseguir sus décimas respectivas, deberás implementar *pop-ups* según lo siguiente:

- **(3 décimas) Ingresar palabra completa:** Debe poder seleccionarse una opción para ingresar la palabra completa de una vez, lo que hará que el usuario gane o pierda instantáneamente.

  Para enviar la palabra completa hacia el *back-end* se debe utilizar la señal `enviar_letra_signal` antes mencionada, donde la palabra ingresada irá en el diccionario junto a la *key* `'palabra'`. En ese caso, la *key* `'letra'` debe ir con un *string* vacío.

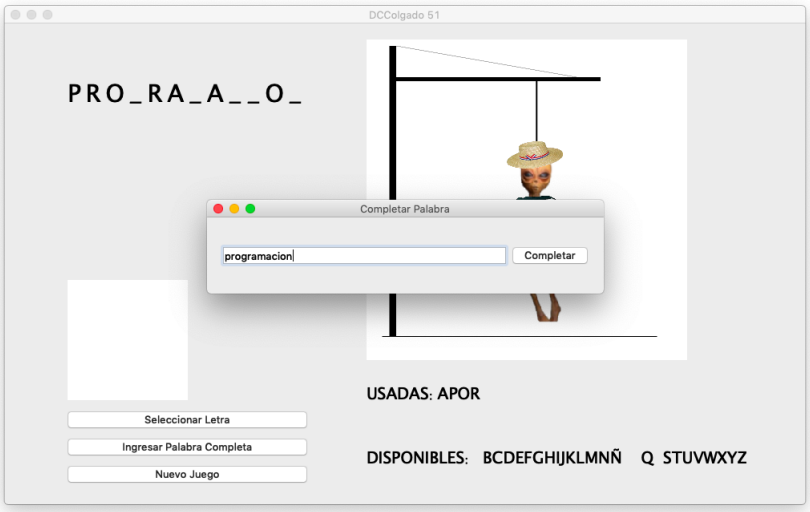

Figura 3: Ejemplo del *pop-up* para la palabra completa.

- **(2 décimas) Victoria y Derrota:** Cuando se adivine la palabra y cuando se pierdan todos los intentos, deberán mostrarse las animaciones[2] `win.gif` y `lose.gif` respectivamente en una ventana *pop.up*.

Figura 4: Ejemplo del *pop-up* en una victoria.

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC05/`
- **Hora del *push*:** 16:45

[2]La clase `QMovie` de `PyQt5.QtGui` te puede ser de utilidad.