SrabbleAR: Informe Final

Martin Souto - Nicolas Hermida

Índice

Introducción

En este informe se detalla el proceso de implementación del juego ScrabbleAR, realizado por los alumnos Nicolas Hermida y Martin Souto. Se va a describir todo el proceso de desarrollo del programa, así como los problemas surgidos y las soluciones a los mismos.

Reglas de juego

El juego consiste en ingresar palabras en el tablero para sumar puntos. Cada jugador (el usuario y la máquina) tiene un atril de letras con las cuales debe formar las palabras, cada letra tiene un distinto puntaje y hay una cantidad distinta de las mismas. Para formar las palabras no se pueden utilizar las letras ya ingresadas en otros turnos en el tablero, excepto en el primer turno que se utiliza la letra de inicio.

Tablero

Hay tres tableros de tamaños distintos, uno pequeño, uno mediano y otro grande. El tablero en el que se va a jugar la partida se selecciona al azar. Cada uno de los tableros contiene casillas especiales y estas son:

- **x2**: Duplica el puntaje de la letra ingresada en esta casilla.
- **x3**: Triplica el puntaje de la letra ingresada en esta casilla.
- **-1**: Le resta un punto al puntaje de la letra ingresada en esta casilla.
- **-2**: Le resta dos puntos al puntaje de la letra ingresada en esta casilla.
- **-3**: Le resta tres puntos al puntaje de la letra ingresada en esta casilla.

Niveles

Hay tres niveles, estos son fácil, medio y difícil. Cada uno tiene distintas características:

- **Fácil**: Las palabras que se pueden ingresar son todas las que el módulo Pattern considere válidas. Por defecto, el tiempo de juego en este nivel es de siete minutos.
- **Medio**: En este nivel solo se pueden ingresar adjetivos o verbos. Por defecto, el tiempo de juego en este nivel es de diez minutos.
- **Difícil**: Las categoría de palabras válidas en este nivel se elige al azar, las opciones son adjetivos o verbos. Por defecto, el tiempo de juego en este nivel es de doce minutos.

Configuración

Cada nivel tiene por defecto distinto tiempo de juego, puntos que vale cada letra y cantidad de letras que hay en la bolsa de fichas. Estos valores pueden ser modificados en la ventana de configuración por el usuario.

Primer turno

Se elige al azar quién es el que inicia el juego, este jugador debe utilizar la letra ubicada en la casilla de inicio para formar su palabra. Si este jugador pasa el turno, el otro jugador es el que debe utilizar esta letra.

Turnos

En cada turno el jugador puede elegir entre ingresar una palabra o cambiar sus fichas. Si intenta ingresar una palabra, debe insertar una por una las letras y luego confirmar para verificar que la palabra sea válida. En cambio, si elige cambiar sus fichas, selecciona las fichas que desea cambiar y estas son intercambiadas por otras de la bolsa de fichas. Cada

jugador tiene tres cambios de fichas, luego, si pasa tres veces el turno, se le dará al jugador un cambio de fichas extra.

Fin de la partida

La partida puede finalizar por tres razones:

- No hay suficientes fichas en la bolsa para completar el atril de un jugador.
- Se acaba el tiempo que dura la partida
- Se presiona el botón de finalizar partida.

Ante cualquiera de estos casos, se suma el puntaje de las letras que quedaron en el atril de cada jugador y se le resta a su puntaje. El jugador con el mayor puntaje gana la partida.

Temas estudiados

Para el desarrollo de este programa se tuvieron que estudiar distintos conceptos sobre el lenguaje de programación Python. Entre ellos se encuentran: estructuras de datos (cadenas, listas, tuplas, conjuntos y diccionarios), estructuras de control (if, elif, else, while, for in), funciones, funciones Lambda, clases, objetos y, por último, módulos y paquetes.

A su vez, se utilizaron módulos externos a Python los cuales son PySimpleGUI y Pattern. Para su correcta implementación se debió estudiar a fondo su comportamiento y definiciones.

PySimpleGUI

Este módulo de software libre permite crear una GUI (Graphical User Interface) personalizada. Brinda un manejo más sencillo sobre el módulo Tkinter de Python, ya que abstrae al usuario de varios errores posibles que se pueden dar usando Tkinter. En la

documentación de este módulo[1] se muestra la siguiente frase: "[...] GUI Development does not have to be difficult nor painful. It can be (and is) FUN [...]".

Durante el desarrollo del programa se debieron estudiar las definiciones e implementaciones de los distintos elementos que nos provee este módulo[2].

Pattern

En la documentación de este módulo[3], se define a Pattern como: "[...]a web mining module for the Python programming language. It has tools for data mining (Google, Twitter and Wikipedia API, a web crawler, a HTML DOM parser), natural language processing (part-of-speech taggers, n-gram search, sentiment analysis, WordNet), machine learning (vector space model, clustering, SVM), network analysis and <canvas> visualization.". Esto quiere decir que Pattern tiene varias funciones, pero la que se debió utilizar para la realización del trabajo es la de procesamiento del lenguaje.

Se utilizó el módulo Pattern.es el cual, según la documentación del mismo, contiene: [4]"a fast part-of-speech tagger for Spanish (identifies nouns, adjectives, verbs, etc. in a sentence) and tools for Spanish verb conjugation and noun singularization & pluralization." Es decir que este módulo reconoce las palabras del lenguaje español e identifica de qué tipo son (sustantivos, adjetivos, verbos, etc.). También se utiliza el módulo Parse el cual: [5] "[...]annotates words in the given string with their part-of-speech tags (e.g., NN for nouns and VB for verbs).[...]", es decir que dada una palabra o varias devuelve un string donde le asigna a cada palabra un tag dependiendo del tipo que sea.

Con estos módulos se logró analizar las palabras ingresadas tanto por el usuario como por la máquina al tablero, y así reconocer si estas estaban dentro de las categorías válidas para el nivel en el que se estaba jugando.

Desarrollo

El desarrollo del proyecto comenzó siguiendo el paradigma de programación imperativo para la creación de un primer tablero generado con PySimpleGUI, el cual funcionó de prueba para las acciones que se debían poder realizar en el juego. Luego se decidió cambiar el paradigma al de orientación a objetos y se crearon varias clases que representan los distintos elementos del tablero, por ejemplo la bolsa de fichas, las casillas, la fila de fichas, etc. Cada una de estas clases tiene funciones las cuales responden a las distintas acciones que realiza el usuario o la máquina a lo largo del juego.

El siguiente paso fue crear la configuración del juego. Para guardar las distintas configuraciones se decidió utilizar archivos json. Cada nivel (fácil, medio y difícil) tiene un archivo el cual almacena una lista la cual puede tener mínimo un diccionario y máximo dos, en el primero se encuentran los datos correspondientes a la configuración por defecto del nivel, mientras que en el segundo se encuentran los datos de la configuración modificada por el usuario, si es que existe. El juego siempre elige el último diccionario para cargar la configuración, por eso siempre que haya una versión modificada la utilizará. Al presionar el botón de restaurar valores se borra el diccionario con la configuración modificada.

Casi todos los elementos del programa tienen imágenes las cuales fueron creadas en su totalidad por los integrantes del grupo que desarrollaron este proyecto.

Problemas y soluciones

Durante el desarrollo del programa surgieron varios problemas e inconvenientes que se tuvieron que superar para poder terminar el juego.

Reloj

Una de las funcionalidades del juego es contar el tiempo en que transcurre la partida, teniendo un tiempo máximo en donde se acaba el juego si es que se llega. Al realizar esta funcionalidad, hubo un problema al contar los segundos debido a que cuando el usuario realizaba un evento (click en algún botón) el temporizador actualizaba mal el reloj. La solución a este problema fue calcular el tiempo que tarda el usuario en dar el click, luego se lo resta al tiempo de la partida y posteriormente se actualiza bien el reloj.

Importación

El segundo problema que surgió durante el desarrollo fue que al modularizar los distintos archivos de código que existían hubo algunos conflictos en la ruta de importación que se estaba utilizando debido a que un paquete solo conoce lo que posee en su interior y no otros paquetes externos. La solución a esto fue ejecutar todo desde el archivo principal ScrabbleAR.py

Sistemas operativos

El último problema surgido fue que al terminar el programa e intentar ejecutarlo en Linux todos los elementos de las ventanas se mostraban en una posición distinta a la que debían estar, lo que hacía que el juego se vea totalmente desprolijo. La solución a este problema fue diferenciar en qué sistema operativo (Windows o Linux) se estaba trabajando y para cada uno de ellos crear un layout distinto en el cual los objetos estaban donde deberían. Debido a esto, el programa pasó a funcionar correctamente tanto en linux como en windows.

Conclusiones

Las conclusiones generadas al finalizar el ScrabbleAR fueron que al realizar cualquier tipo de proyectos, ya sean en el ámbito informático o en otro, el tener una organización detallada de cómo se quiere llevar a cabo dicho proyecto es esencial para que este pueda desarrollarse de una manera ordenada y con el mínimo número de problemas. A su vez, al trabajar en equipo, siempre es conveniente que todos los miembros sean capaces de expresar su opinión respetando las de los demás, ya que todas las opiniones son válidas y sirven para mejorar el proyecto.

Referencias

[1]: PySimpleGUI - Home

[2]: PysimpleGUI - Call reference

[3]: Pattern - GitHub Documentation

[4]: Pattern.es - GitHub Documentation

[5]: Pattern.es: Parser - GitHub Documentation

Anexo 1: Guía de usuario

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto. Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.