Proyecto I – Scrabble

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos II (CE 2103)
Primer Semestre 2019
Valor 25%

Objetivo General

➔ Diseñar e implementar un juego multiplayer.

Objetivos Específicos

➔ Implementar listas enlazadas y algunas variaciones
➔ Investigar y desarrollar una aplicación en el lenguaje de programación C++
➔ Investigar acerca de programación orientada a objetos en C++.
➔ Aplicar **patrones de diseño** en la solución de un problema.

Descripción del Problema

Scrabble es un juego de mesa en el cual cada jugador intenta ganar más puntos mediante la construcción de palabras sobre un tablero de 15x15 casillas. Las palabras pueden formarse horizontal o verticalmente y se pueden cruzar siempre y cuando aparezcan en el diccionario estándar.

El juego se realiza entre 2, 3 o 4 jugadores (en general se juega de a 2), sobre un tablero de 15x15 casillas, en las que cada jugador coloca sus fichas. Cada jugador recibe un número específico de fichas (o letras) extraídas aleatoriamente. Las letras se encuentran numeradas con su respectivo valor, obteniéndose por cada palabra formada una puntuación que depende tanto del valor de las letras empleadas como de la posición de dichas letras dentro del tablero.

En total hay 100 fichas, 98 marcadas con letras y dos en blanco (sin puntos, actúan como comodines usándose para reemplazar letras). Según su frecuencia de aparición, las letras tienen más o menos puntos, siempre las de mayor frecuencia valen menos.

La cantidad de fichas por letra y puntaje es:

- 2 fichas en blanco (0 puntos)
- *1 punto*: **A** ×12, **E** ×12, **O** ×9, **I** ×6, **S** ×6, **N** ×5, **L** ×4, **R** ×5, **U** ×5, **T** ×4
- *2 puntos*: **D** ×5, **G** ×2
- *3 puntos*: **C** ×4, **B** ×2, **M** ×2, **P** ×2
- *4 puntos*: **H** ×2, **F** ×1, **V** ×1, **Y** ×1
- *5 puntos*: **CH** ×1, **Q** ×1
- *8 puntos*: **J** ×1, **LL** ×1, **Ñ** ×1, **RR** ×1, **X** ×1
- *10 puntos*: **Z** ×1,

Scrabble es un juego multiplayer y cliente-servidor. El servidor lleva el control de la puntuación de los jugadores y del estado actual del tablero. El servidor tiene un diccionario de palabras (un archivo con todas las posibles palabras) para validar que las palabras propuestas por los jugadores sean correctas.

Se les solicita implementar el juego este juego en C++, en una arquitectura cliente-servidor. A continuación los detalles.

El cliente es una aplicación de escritorio. Muestra una ventana inicial en la que el jugador puede crear un nuevo juego en el servidor (especificando cuántas personas pueden jugar) o unirse a un juego existente. Cuando se crea un nuevo juego, se genera un código de 6 dígitos que se puede compartir con otras personas para que se unan. Si un jugador trata de unirse a un juego con el máximo de personas, se muestra un error. Al unirse a un juego, se debe especificar también el nombre del jugador. La aplicación cliente tiene un archivo de configuración en el que se especifica el IP, puerto del servidor y teléfono del experto en palabras. Más sobre esto adelante. Una vez que se ha unido al juego, el cliente se conecta al servidor para obtener las fichas asignadas. El cliente muestra las fichas y la cuadrícula al usuario. El usuario puede entonces arrastrar fichas hacia la cuadrícula y presionar el botón scrabble! para jugar su turno. Cuando se presiona el botón, el servidor valida la palabra propuesta contra un diccionario de palabras válidas. Si la palabra es válida, se le conceden los puntos al jugador. Si la palabra no es correcta, el servidor se lo indica al jugador y le da dos opciones: intentar otra vez o consultar al experto. Si se consulta al experto, el servidor envía un SMS al experto para que valide la palabra. El experto puede responder al SMS aceptando o rechazando la palabra. SI es aceptada, la palabra se agrega al diccionario y se le da el puntaje al jugador. Cada jugada válida de un jugador, actualiza la cuadrícula en el server, el cual envía la nueva palabra al resto de jugadores para que la muestren en pantalla. El jugador que se quede sin letras, gana.	Toda la comunicación con el servidor es por socket y en formato JSON Es un properties file La cuadrícula es una matriz. Las palabras en el cliente se maneja como listas enlazadas. El servidor lleva el rastro de palabras en la cuadrícula. El servidor debe validar cruces entre palabras.

Tanto el cliente como el servidor deben utilizar alguna biblioteca para serializar/deserializar JSON.

Documentación requerida

1. Internamente, el código se debe documentar utilizando DoxyGen y se debe generar el HTML de la documentación.
2. Dado que el código se deberá mantener en GitHub, la documentación externa se hará en el Wiki de GitHub. El Wiki deberá incluir:
    a. Breve descripción del problema
    b. **Planificación y administración del proyecto:** se utilizará la parte de project management de GitHub para la administración de proyecto. Debe incluir:
        ➔ Lista de features e historias de usuario identificados de la especificación
        ➔ Distribución de historias de usuario por criticalidad
        ➔ Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental
        ➔ Descomposición de cada user story en tareas.

➔ Asignación de tareas a cada miembro del equipo.

c. Diagrama de clases en formato JPEG o PNG
d. Diagrama de arquitectura en formato JPEG o PNG.
e. Diagrama de componentes en formato JPEG o PNG.
f. Descripción de las estructuras de datos desarrolladas.
g. Descripción detallada de los algoritmos desarrollados.
h. Problemas encontrados en forma de bugs de *github*: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.

Aspectos operativos y evaluación:

1. **Fecha de entrega: De acuerdo al cronograma del curso**
2. El proyecto tiene un valor de 25% de la nota del curso.
3. El trabajo es **en grupos de 3 personas**.
4. Es obligatorio utilizar un GitHUB.
5. Es obligatorio integrar toda la solución.
6. El código tendrá un valor total de 75%, la documentación 20% y la defensa 5%.
7. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
8. Se evaluará que la documentación sea coherente, acorde a la dificultad/tamaño del proyecto y el trabajo realizado, se recomienda que realicen la documentación conforme se implementa el código.
9. La nota de la documentación es proporcional a la completitud del proyecto.
10. La documentación se revisará según el día de entrega en el cronograma.
11. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
12. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la cita revisión oficial
13. Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
    a. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
    b. Si no se utiliza un manejador de código se obtiene una nota de 0.
    c. Si la documentación no se entregan en la fecha indicada se obtiene una nota de 0.
    d. Sí el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
    e. El código debe ser desarrollado en C++, en caso contrario se obtendrá una nota de 0.
    f. Si no se siguen las reglas del formato de email se obtendrá una nota de 0.
    g. La nota de la documentación debe ser acorde a la completitud del proyecto.
14. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de clases, documentación interna y la documentación en el manejador de código.
15. Cada estudiante tendrá como máximo 15 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
16. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.

17. Cada estudiante es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberán avisar al menos 2 días antes de la revisión a el profesor para coordinar el préstamo de estos.
18. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.