Proyecto I – Juego de Memoria

Instituto Tecnológico de Costa Rica
Área Académica Ingeniería en Computadores
CE2103 - Algoritmos y Estructuras de Datos II
Primer Semestre 2022
Valor: 25%

Objetivo General

- ➔ Desarrollar una aplicación utilizando el lenguaje de programación C++.

Objetivos Específicos

- ➔ Aplicar conceptos de manejo de memoria.
- ➔ Investigar y desarrollar una aplicación en el lenguaje de programación C++
- ➔ Implementar una solución utilizando programación orientada a objetos en C++.
- ➔ Implementar una arquitectura Cliente-Servidor en C++
- ➔ Aplicar patrones de diseño en la solución de un problema.

Descripción del Problema

Se le solicita implementar un juego de memoria donde el jugador debe seleccionar parejas de tarjetas iguales dispuestas al azar al iniciar. No todas las tarjetas están cargadas en memoria, por lo que debe aplicar conceptos de paginación.

Cuando el juego inicia, se le solicitará ingresar el nombre de los dos jugadores. Posteriormente, se escoge aleatoriamente cual usuario tiene el primer turno y se muestra la pantalla principal del juego. La pantalla del juego se verá similar a la siguiente imagen:

Durante su turno, el jugador selecciona mediante el mouse, un par de tarjetas suponiendo que dichas contienen la misma imagen. Cuando el jugador selecciona cada una de las cartas, se rota mostrando al usuario la imagen contenida. Las tarjetas cuyas parejas han sido encontradas, se marcan con un algún indicador visual que muestre claramente que ya no están en juego. En todo momento se debe mostrar la cantidad de puntos de cada jugador. El juego soporta tres power ups propuestos por cada estudiante. La ocurrencia y la funcionalidad

de cada power up queda a criterio del estudiante.

El juego tiene dos componentes de software principales: el cliente y el servidor. El cliente es la interfaz gráfica que puede estar escrita en C++ o Java. **Este componente únicamente renderiza el estado del juego mantenido en el servidor**. **El cliente no tiene ninguna lógica de negocio**. Cada acción del usuario en la interfaz gráfica envía comandos al servidor para actualizar el estado del mismo.

El servidor es un programa aparte escrito en C++ con memoria limitada. Las tarjetas se modelan como una matriz (que puede ser sumamente grande) de objetos tipo Tarjeta. Para efectos de mostrar el manejo de la memoria, cada objeto tarjeta tiene internamente un array de bytes de la imagen de la tarjeta junto con cualquier otro dato necesario para la tarjeta. El servidor solo puede mantener la tercera parte de tarjetas en memoria. Es decir, si al iniciar hay 60 tarjetas, solo 20 se pueden mantener en memoria. Dicha cantidad debe ser dinámica y reducirse conforme las tarjetas se vayan eliminando cuando el jugador encuentre sus parejas. El algoritmo de reemplazo será definido por cada estudiante.

Cada vez que un jugador encuentre una pareja, el servidor cambiará aleatoriamente las tarjetas que están en memoria.

El servidor tiene una interfaz gráfica sencilla que indica en una tabla, cuáles páginas están en memoria y cuales están en disco. Indica también el puntaje de cada jugador para el juego actual. Muestra en todo momento el consumo de memoria, que debe ser claro que la memoria no aumenta al cargar nuevas tarjetas en memoria. También muestra la cantidad de page faults/page hit acumulados hasta el momento.

El cliente y el servidor se comunican mediante sockets. Cada estudiante deberá definir el protocolo de comunicación (formato, comandos, respuestas).

Cuando un jugador encuentra un par de tarjetas, y dichas tarjetas están en memoria, se le dará puntos adicionales.

Cada interacción que realice la lógica del servidor con la matriz de tarjetas, deberá considerar la paginación y cargar/descargar cualquier tarjeta para adherirse a la cantidad de tarjetas en memoria.

Documentación requerida

1. Internamente, el código se debe documentar siguiendo los estándares de documentación para cada lenguaje utilizado.
2. La documentación externa se hará en un documento que incluya lo siguiente (deberá entregarse un PDF):
    a. Breve descripción del problema.
    b. Diagrama de clases.
    c. Descripción de las estructuras de datos desarrolladas.
    d. Descripción del funcionamiento del algoritmo de paginación
    e. Protocolo de comunicación entre cliente y servidor
    f. Problemas encontrados en forma de bugs de *github*: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
3. **Planificación y administración del proyecto:** se utilizará Jira para la administración del proyecto. Debe incluir:
    a. Lista de features e historias de usuario identificados de la especificación.
    b. Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un

desarrollo incremental

c. Descomposición de cada user story en tareas.

Aspectos operativos y evaluación:

1. **Fecha de entrega: De acuerdo con el cronograma del curso y lo establecido en el TEC Digital**
2. El proyecto tiene un valor de 25% de la nota del curso.
3. El trabajo es en **individual.**
4. Deben entregar en el TEC Digital/classroom (según aplique) un documento con el link del repositorio de GitHub, Jira y el PDF de la documentación. En ambas herramientas deben dar acceso al correo del profesor.
5. Es obligatorio utilizar un Git y GitHUB para el control de versiones del código fuente y evidenciar el uso de Commits frecuentes.
6. Es obligatorio integrar toda la solución.
7. El código tendrá un valor total de 70%, la documentación externa 20% y la defensa un 10%.
8. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
9. 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.
10. La nota de la documentación externa es proporcional a la completitud del proyecto.
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 primera cita de revisión oficial.
13. Aún cuando el código y la documentación externa tienen sus notas por separado, se aplican las siguientes restricciones
    a. Si no se entrega documentación externa, automáticamente se obtiene una nota de cero en la nota final del proyecto.
    b. Si no se utiliza un manejador de código se obtiene una nota de cero en la nota final del proyecto.
    c. Si la documentación externa no se entrega en la fecha indicada se obtiene una nota de cero en la nota final del proyecto.
    d. Sí el código no compila se obtendrá una nota de cero en la nota final del proyecto, por lo cual se recomienda realizar la defensa con un código funcional.
    e. El código debe ser desarrollado en C++ (Linux), en caso contrario se obtendrá una nota de cero en la nota final del proyecto.
14. La revisión de la documentación será realizada por parte del profesor. Podría ser revisada, por parte del profesor, antes o después de la cita de revisión del proyecto. Durante la defensa del proyecto sí se revisará el diagrama de clases, la 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 al profesor para coordinar el préstamo de estos.

18. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.