Instituto Tecnológico
de Costa Rica

Algoritmos y estructuras de datos 1.

Connect dots

Fabricio Mena 2019042722

Joseph Murillo 2022437962

Nathalia Ocampo 2021008216

Prof. Leonardo Araya

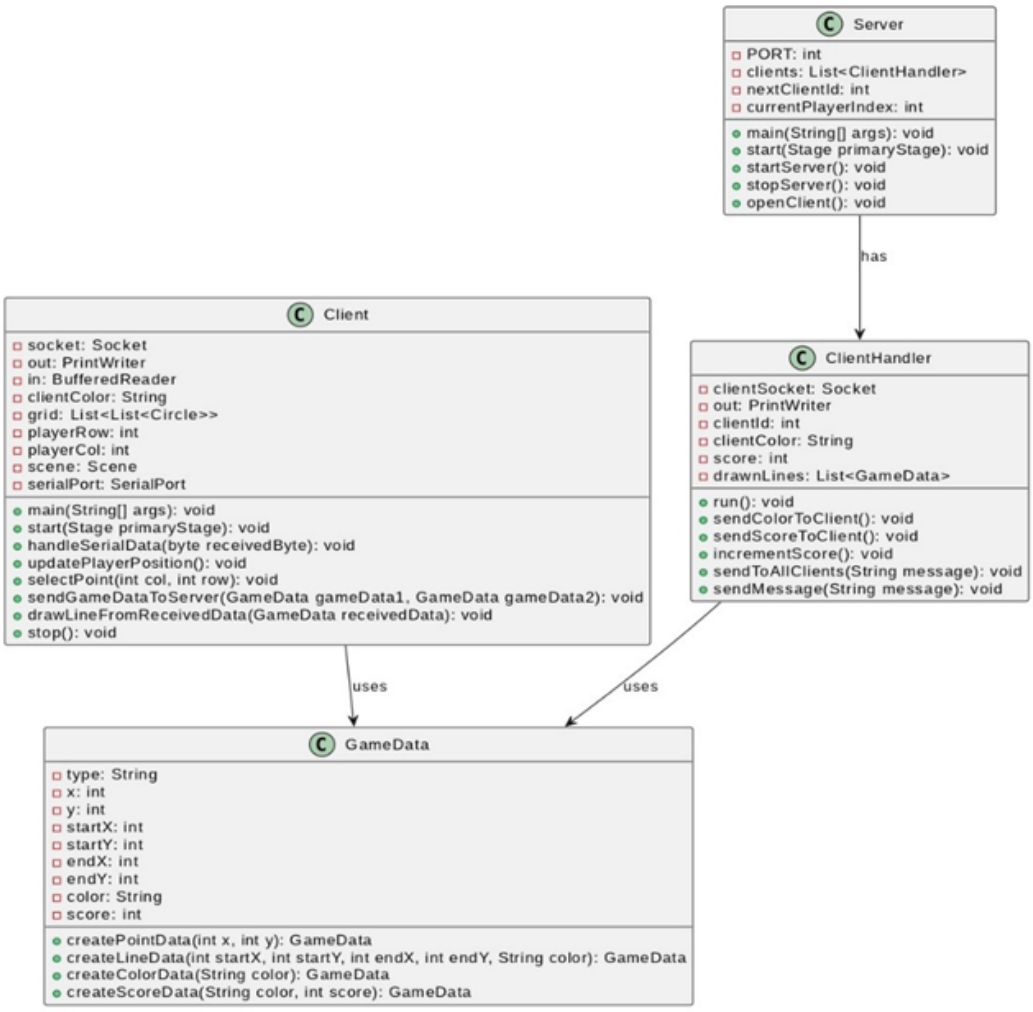
September 30, 2023

0.1 Descripcion de problema

Para este proyecto se pretende recrear el juego "Connect Dots", dicho juego presenta modalidad de multijugador, está diseñado para n cantidad de participantes.
Este juego consiste en una malla de puntos donde el jugador puede unir los puntos por medio de lineas por turno. Cuando algún jugador logra cerrar un cuadrado en su turno, puede seguir jugando hasta que se agregue una línea que no cierre en algún cuadrado.

El objetivo del juego consiste en cerrar cuadrados para así obtener puntos. El jugador que logré cerrar más cuadrados en total al finalizar la partida será el ganador.
Todos los cuadrados poseen el mismo valor. Cuando un jugador cierra un cuadrado, a este se le asignará algún identificador que permita saber cuál fue el jugador que lo cerró.
"Dots" es un juego diseñado para múltiples jugadores. Un servidor central, implementado como una aplicación en Java, asume la función de estar atento a las conexiones entrantes mediante un Socket.Cada aplicación cliente se ejecuta en una computadora y se conecta por socket al servidor. Al dar inicio al juego, el servidor recibe la solicitud del cliente y lo incorpora en una cola de espera. Una vez que se han reunido dos o más jugadores, se desencadena el comienzo del juego. Este puede tener simultáneamente a varios jugadores que participan.
El servidor lleva el control completo del juego. Los clientes únicamente grafican lo que el servidor les envía. Los clientes a su vez envían las acciones que realicen al servidor, el cual se encarga de mantener el estado del juego completo. Toda la comunicación entre cliente y servidor es en formato JSON.
Adicionalmente, un cliente tendrá un control elaborado por los estudiantes que permitirá al jugador seleccionar dónde colocará las líneas.

0.2 Diagrama de clases



0.3 Descripcion de las estructuras de datos

GameData[ ] drawnLines, funciona para almacenar las líneas dibujadas y saber cuando se dibujaron todas, y así terminar el juego
GameData[ ] foundSquares, funciona para almacenar cuadrados y sumarselos al score del cliente
ClientHandler[ ] clients funciona para almacenar los clientes activos
String [ ] colors, mantiene una lista de los colores que se le asigna a los clientes
Circle [ ] [ ] grid, es una estructura de lista bidimensional utilizada en la clase Client para representar la cuadrícula de puntos (círculos) en la interfaz gráfica del juego. La lista contiene listas de objetos Circle (círculos) que representan los puntos en la cuadrícula.

Desafío: Sincronización y Concurrencia en la Arquitectura Cliente-Servidor Tradicional

a. Listado de requerimientos del sistema:
Los requerimientos del sistema para el juego "Connect Dots" pueden definirse de la siguiente manera:
1. Requerimientos Funcionales
• El sistema debe permitir a múltiples jugadores conectarse al servidor.
• Los jugadores deben poder seleccionar puntos en la cuadrícula para dibujar líneas.
• El sistema debe gestionar turnos entre jugadores.
• Los jugadores deben poder cerrar cuadrados para obtener puntos.
• El sistema debe llevar un registro de los puntos y cuadrados cerrados por cada jugador.

• El servidor debe mantener el control completo del juego y enviar actualizaciones a los clientes.
• Los clientes deben mostrar la interfaz gráfica del juego y enviar acciones al servidor.
• El sistema debe ser capaz de iniciar y finalizar partidas.

2. Requerimientos No Funcionales:
• La comunicación entre el servidor y los clientes debe utilizar el formato JSON.
• El sistema debe ser escalable para admitir un número variable de jugadores.
• La interfaz gráfica del juego debe ser intuitiva y fácil de usar.
• El sistema debe ser eficiente en términos de consumo de ancho de banda y recursos.
• Se debe garantizar la seguridad en la comunicación para prevenir trampas.

b. Elaboración de opciones de solución al problema:
A continuación, se presentan dos opciones de solución posibles para el juego "Connect Dots", junto con diagramas conceptuales simplificados que ilustran cada enfoque:

1. Arquitectura Cliente-Servidor Tradicional:
• **Descripción:** En esta opción, se implementa una arquitectura cliente-servidor tradicional donde el servidor centraliza la lógica del juego y la comunicación entre jugadores.
• **Diagrama Conceptual:** Se muestra un servidor central que gestiona múltiples clientes conectados. El servidor controla la lógica del juego y envía actualizaciones a los clientes.

2. Arquitectura Juego Distribuido:
Descripción: En esta opción, se implementa una arquitectura de juego distribuido donde cada cliente ejecuta su propia lógica de juego y se comunica con otros clientes para sincronizar el estado del juego. Diagrama Conceptual: Se muestran varios clientes

conectados que interactúan directamente entre sí para sincronizar el juego. El servidor actúa como un coordinador inicial.

c. Valoración de opciones de solución:

La valoración de las opciones de solución se realizaría en función de los siguientes criterios:

• **Escalabilidad:** El enfoque de arquitectura de juego distribuido es más escalable, ya que permite un mayor número de jugadores sin sobrecargar el servidor central.

• **Eficiencia:** La arquitectura cliente-servidor tradicional puede ser más eficiente en términos de ancho de banda y recursos del servidor, ya que centraliza la lógica del juego. Sin embargo, la arquitectura distribuida ofrece una distribución de carga entre los clientes.

• **Seguridad:** Ambas opciones deben implementar medidas de seguridad para prevenir trampas. Esto debe ser evaluado en detalle.

• **Facilidad de desarrollo:** La arquitectura cliente-servidor tradicional puede ser más fácil de desarrollar y mantener, ya que el servidor controla la lógica del juego. La arquitectura distribuida es más compleja en términos de sincronización entre clientes.

• **Experiencia del usuario:** Ambas opciones deben proporcionar una experiencia de usuario satisfactoria, por lo que la interfaz gráfica y la capacidad de juego son criterios importantes.

• **Mantenibilidad:** La facilidad de mantenimiento a largo plazo debe considerarse, incluyendo actualizaciones y modificaciones futuras.

Selección de la propuesta final:

La opción seleccionada como propuesta final es la Arquitectura Cliente-Servidor Tradicional. A continuación, se argumenta la elección:

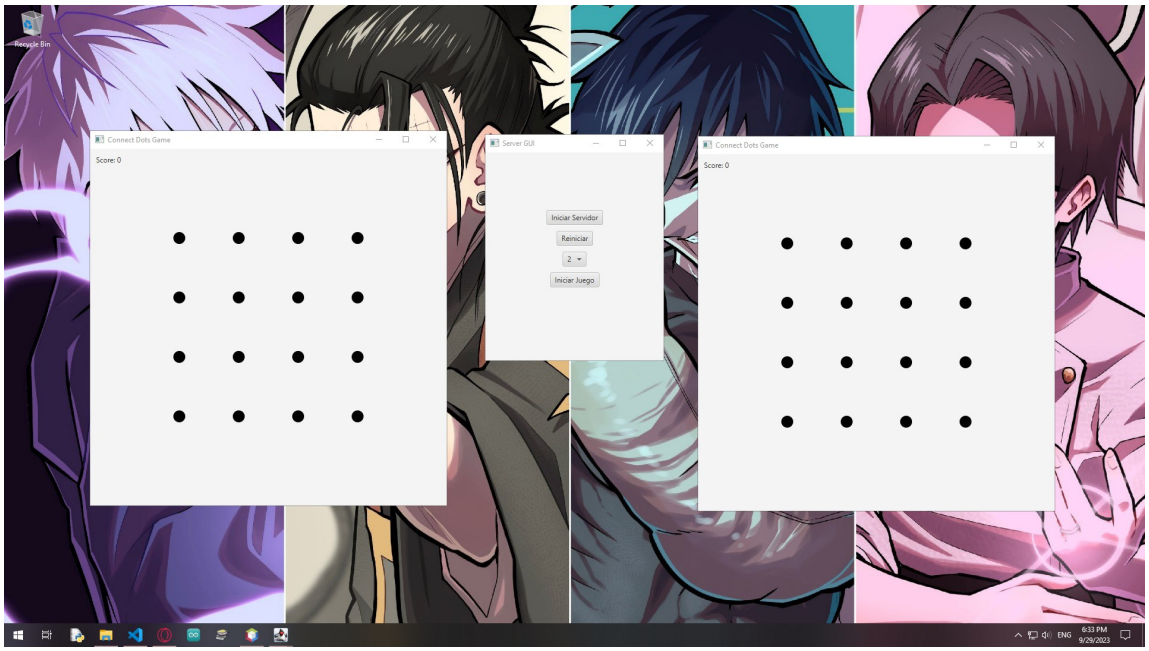
Razones para la Selección:
1. Centralización de la Lógica del Juego: La arquitectura cliente-servidor tradicional centraliza la lógica del juego en el servidor, lo que facilita el control y la administración del juego. Esto es especialmente útil en un juego como "Connect Dots", donde se requiere un control estricto de los turnos, la puntuación y la detección de cuadrados cerrados.
2. Seguridad: La centralización de la lógica del juego en el servidor permite implementar medidas de seguridad más efectivas para prevenir trampas y asegurar la integridad del juego. El servidor puede validar y autorizar las acciones de los jugadores de manera más confiable.
3. Escalabilidad Controlada: Aunque la arquitectura distribuida puede ser más escalable, la arquitectura cliente-servidor tradicional es más adecuada para un número razonable de jugadores, lo que se ajusta a los objetivos del proyecto.
4. Facilidad de Desarrollo y Mantenimiento: La opción cliente-servidor tradicional es más fácil de desarrollar y mantener a largo plazo. Facilita la identificación y corrección de errores, así como la implementación de futuras actualizaciones.
5. Experiencia del Usuario: La centralización de la lógica del juego simplifica la experiencia del usuario, ya que los clientes no necesitan gestionar tanta lógica compleja y sincronización entre sí.

e. Diseño de la Alternativa Seleccionada:
El diseño final seleccionado se basa en la arquitectura cliente-servidor tradicional. A continuación, se describe el diseño en términos de:
• **Protocolo de Comunicación:** Se utilizará el protocolo TCP/IP para la comunicación entre clientes y servidor, asegurando una comunicación confiable y ordenada.
• **Diagrama de Bloques del Sistema:** Se elaborará un diagrama de bloques del sistema que incluye componentes clave como

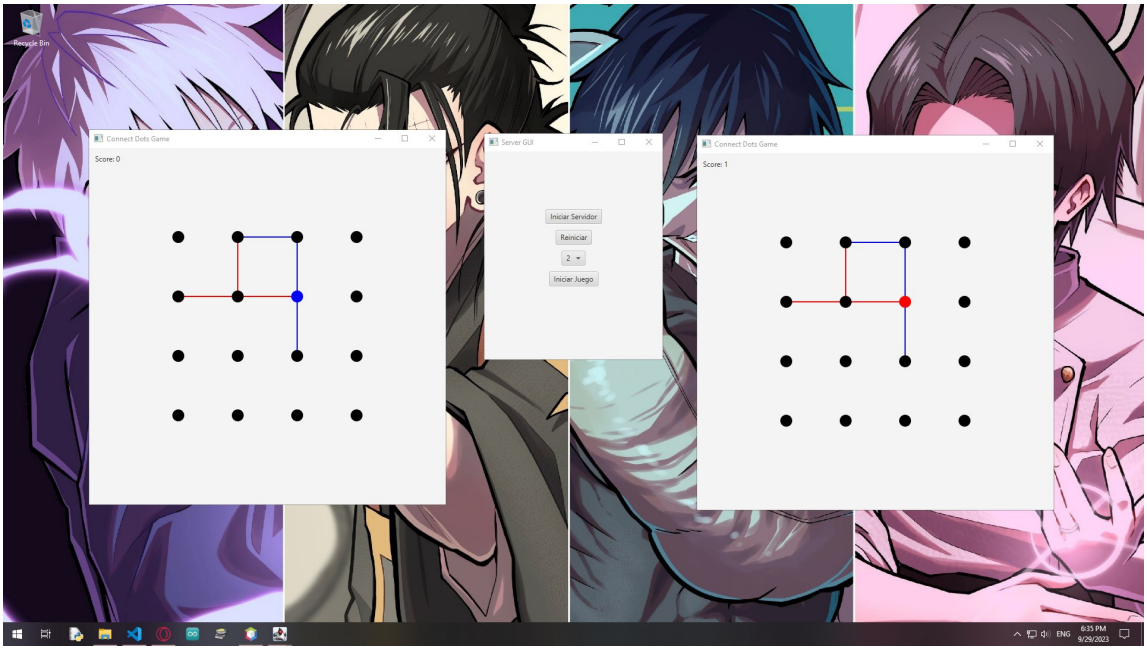
el servidor central, múltiples clientes, la interfaz gráfica del juego y la lógica del juego.
• **Diagrama UML:** Se crearán diagramas UML, incluyendo diagramas de clases y secuencia para representar la estructura de clases del juego y la interacción entre componentes durante el juego.

f. Validación del Diseño:
La validación del diseño se llevará a cabo mediante pruebas exhaustivas del sistema. Se verificará que el diseño cumpla con los requerimientos establecidos, garantice la salud y seguridad pública, minimice el costo total de la vida, mantenga el carbono neto cero y considere aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.
Las pruebas incluirán:
• Pruebas de funcionalidad para asegurarse de que todas las funciones del juego funcionen correctamente, incluyendo la selección de puntos, cierre de cuadrados, registro de puntajes y control de turnos.
• Pruebas de seguridad para validar la integridad del juego y prevenir trampas.
• Pruebas de rendimiento para evaluar la eficiencia del sistema en términos de uso de recursos y consumo de ancho de banda.
• Pruebas de usabilidad para garantizar una experiencia de usuario satisfactoria.
• Pruebas de escalabilidad para evaluar el comportamiento del sistema con múltiples jugadores.
• Pruebas de mantenibilidad para asegurar que el sistema pueda actualizarse y modificarse de manera efectiva en el futuro.

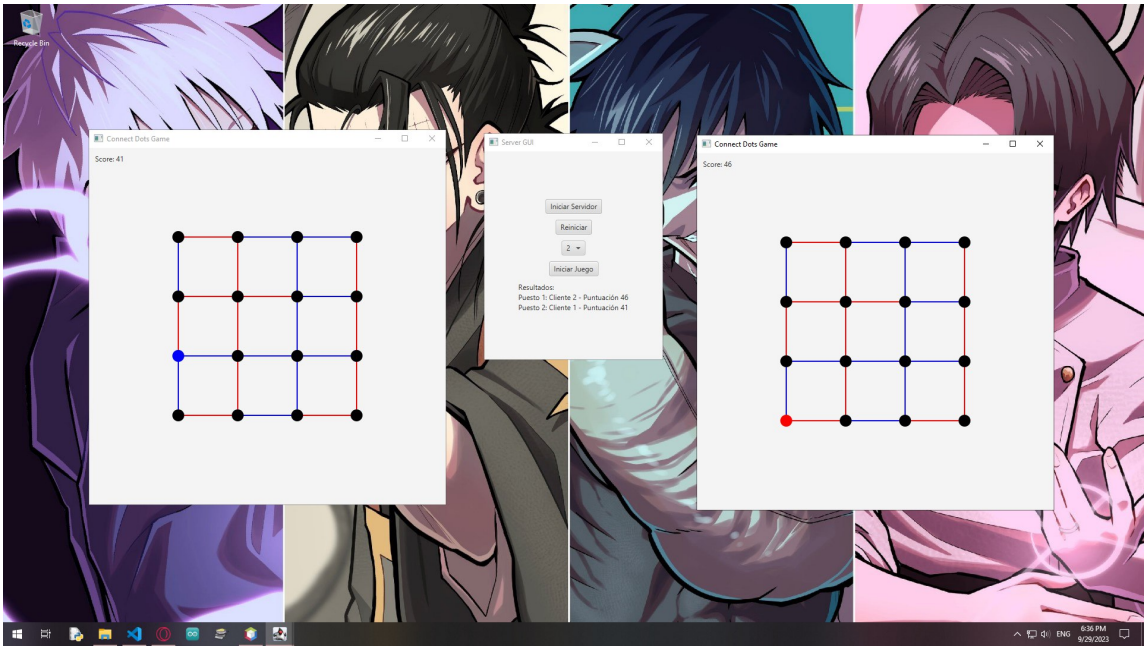
0.4 Capturas de pantalla



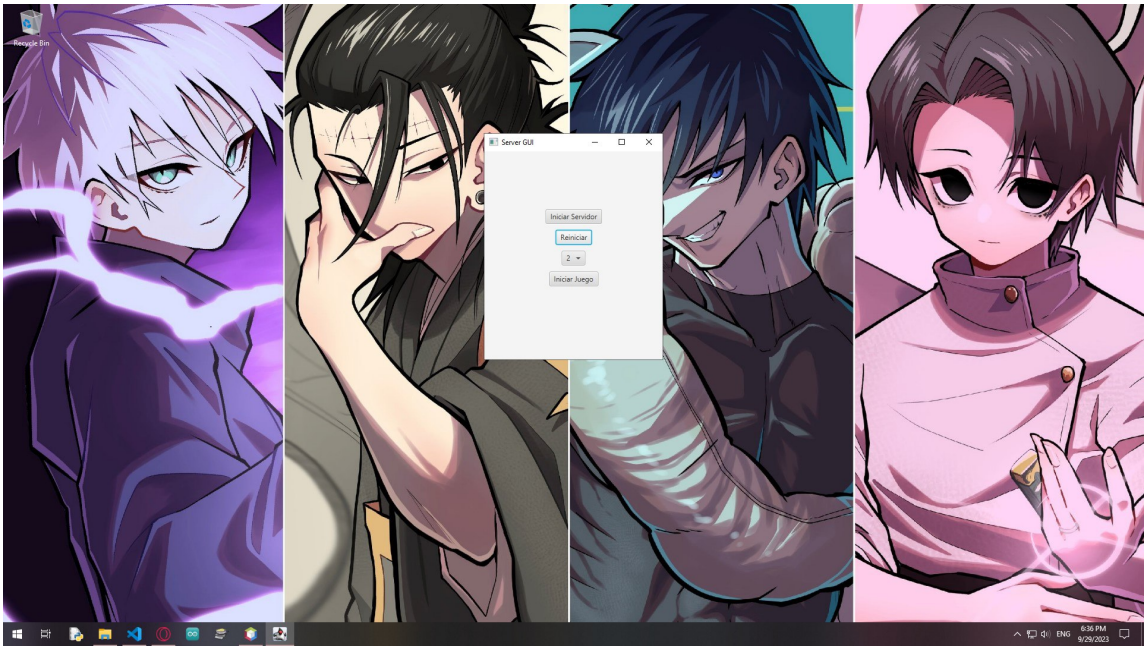
En esta imagen se puede observar el menú del juego con sus dos pantallas para cada usuario, así mismo se observa la ventana del servidor donde se evidencian los botones de “inicio”, siendo este botón quien da por iniciado el juego, también se puede observar el botón de “reiniciar”, seguidamente el list box permite seleccionar la cantidad de jugadores, por último en el servidor del juego se observa el botón de “inicio juego”, este es quien da por iniciado el juego de “Connect dots”



En esta imagen se evidencia de como el jugador de la derecha (cliente 2) encierra su cuadro y suma sus puntos, en el "score" se evidencia que hasta este momento el cliente 2 se mantiene ganando la partida por 1 puntos, por el otro lado el jugador de la izquierda (cliente 1), se mantiene con 0 puntos lo de por el momento lo mantiene perdiendo.



En esta imagen se logra observar el avance del juego siendo así el cliente 2 quien se mantiene ganando la partida por 46 puntos, se observa el puntaje del cliente 1 con 41 puntos es decir, quien gana la prtida en esta ocación es el cliente 2.



Finalizando se evidencia el botón de "reiniciar", esto se oprime para que el score reinicie y puedan jugar más partidas y seguir disfrutando del juego.