**METODO DE DISEÑO DE LA INGENIERIA**

**FASE 1: PLANTEAMIENTO DEL PROBLEMA**

Descripción del Proyecto:

El objetivo de este proyecto es desarrollar un juego interactivo para uno o más jugadores que pueda ser modelado utilizando grafos. El juego debe tener un mínimo de 50 vértices y 50 aristas en su representación gráfica, permitiendo así la aplicación de al menos dos algoritmos de grafos específicos.

Definición del Juego:

El juego será diseñado y definido por el equipo, con un límite máximo de 3 personas, incluyendo al responsable del planteamiento. La mecánica del juego debe ser tal que permita la aplicación de algoritmos de grafos, y debe ser lo suficientemente compleja para justificar la representación mediante grafos.

**FASE 2: RECOLECCIÓN DE INFORMACIÓN**

La fase de recolección de información es esencial para comprender completamente el problema y establecer la base para el diseño del juego basado en grafos. En esta fase, se recopilará información relevante para definir los requisitos del juego y los elementos que serán modelados mediante grafos.

1. Definición del Juego:

Descripción General del Juego:

Identificar el género y la temática del juego.

Definir el objetivo principal del juego y las metas que los jugadores deben alcanzar.

Mecánica del Juego:

Describir cómo los jugadores interactuarán con el juego.

Identificar las acciones y decisiones clave que los jugadores tomarán durante el juego.

Requisitos Funcionales y No Funcionales:

Enumerar las funciones esenciales que debe cumplir el juego.

Identificar restricciones y requerimientos no funcionales, como rendimiento y escalabilidad.

2. Elementos del Juego Modelados con Grafos:

Identificación de Nodos y Aristas:

Determinar los elementos específicos del juego que se representarán como nodos en el grafo.

Identificar las conexiones entre estos elementos y cómo se traducirán en aristas del grafo.

Número de Vértices y Aristas:

Confirmar que el juego tiene al menos 50 vértices y 50 aristas, según lo requerido.

3. Algoritmos de Grafos Aplicables:

Selección de Algoritmos:

Evaluar cómo los algoritmos de grafos (BFS, DFS, Dijkstra, Floyd-Warshall, Prim, Kruskal) pueden aplicarse al contexto del juego.

Identificar situaciones específicas del juego en las que cada algoritmo podría ser útil.

4. Documentación y Estándares:

Normas y Convenciones:

Establecer las normas y convenciones para la documentación del código y diseño.

Definir estándares para la implementación de los algoritmos de grafos.

5. Tecnologías y Plataformas:

Plataforma de Desarrollo:

Seleccionar la plataforma o entorno de desarrollo más adecuado para implementar la interfaz gráfica y los componentes del juego.

Herramientas Gráficas:

Identificar las herramientas que se utilizarán para crear la interfaz gráfica de usuario.

6. Grupos de Trabajo y Roles:

Estructura del Equipo:

Definir los roles y responsabilidades de cada miembro del equipo.

Establecer una estructura de colaboración eficiente.

7. Evaluación de Riesgos:

Identificación de Riesgos:

Enumerar posibles desafíos y obstáculos que puedan surgir durante el desarrollo.

Evaluar cómo mitigar o abordar estos riesgos.

**FASE 3: BUSQUEDA DE SOLUCIONES CREATIVAS**

La fase de búsqueda de soluciones creativas es crucial para diseñar un juego atractivo basado en grafos. Aquí se explorarán ideas innovadoras y creativas que permitan la implementación de los requisitos establecidos en la fase de recolección de información. Esta fase busca maximizar la jugabilidad y la integración efectiva de los algoritmos de grafos.

1. Diseño del Juego:

Exploración de Conceptos:

Realizar sesiones de lluvia de ideas para conceptualizar diferentes enfoques y mecánicas del juego.

Considerar cómo los elementos del juego pueden ser representados y manipulados mediante grafos de manera interesante.

Interacción de Jugadores:

Diseñar interacciones innovadoras que involucren a los jugadores en la manipulación directa del grafo.

Explorar posibilidades de estrategias competitivas o cooperativas basadas en los algoritmos de grafos.

2. Implementación de Grafos:

Versatilidad de Implementación:

Asegurar que la implementación del grafo sea modular y adaptable.

Explorar diferentes estructuras de datos y algoritmos para la representación y manipulación eficiente del grafo.

Integración con la Mecánica del Juego:

Enlazar la representación del grafo con los elementos y la mecánica del juego de manera coherente.

Garantizar que los algoritmos de grafos se integren de manera fluida con la progresión del juego.

3. Interfaz Gráfica de Usuario (GUI):

Diseño Atractivo:

Desarrollar una interfaz gráfica atractiva que refleje visualmente el grafo y sea fácil de entender.

Implementar animaciones o efectos visuales que mejoren la experiencia del usuario.

Facilidad de Navegación:

Diseñar controles intuitivos que permitan a los jugadores interactuar de manera eficiente con el grafo.

Asegurar que la interfaz sea accesible para jugadores de diferentes niveles de experiencia.

4. Algoritmos de Grafos:

Contextualización en el Juego:

Identificar situaciones específicas del juego donde los algoritmos de grafos se pueden aplicar de manera estratégica.

Integrar los algoritmos de manera natural en la narrativa o dinámica del juego.

Adaptabilidad:

Diseñar los algoritmos de manera que sean adaptables a diferentes configuraciones y tamaños de grafos.

Considerar la complejidad y eficiencia de cada algoritmo en el contexto del juego.

5. Pruebas y Ajustes:

Prototipado Iterativo:

Desarrollar prototipos del juego para probar las ideas y realizar ajustes rápidos.

Recopilar retroalimentación del equipo y realizar mejoras continuas.

Pruebas de Jugabilidad:

Realizar pruebas de jugabilidad para evaluar la experiencia general de los jugadores.

Ajustar la dificultad y el equilibrio del juego según la retroalimentación recibida.

6. Documentación Creativa:

Documentación Visual:

Utilizar diagramas, esquemas y gráficos para documentar visualmente la relación entre los elementos del juego y el grafo.

Crear documentos visuales que ayuden a comunicar de manera efectiva el diseño del juego.

7. Evaluación de Viabilidad:

Revisión de Recursos:

Evaluar la viabilidad técnica y de recursos de las soluciones propuestas.

Hay que asegurar que la implementación sea factible dentro de los plazos y recursos disponibles.

Esta fase de búsqueda de soluciones creativas busca inspirar y explorar enfoques únicos que no solo cumplan con los requisitos técnicos, sino que también proporcionen una experiencia de juego emocionante y atractiva para los jugadores. La creatividad y la innovación son clave para diferenciar el juego y garantizar su éxito.

**FASE 4: TRANSICIÓN DE LA LLUVIA DE IDEAS A LOS DISEÑOS PRELIMINARES**

La transición de la lluvia de ideas a los diseños preliminares es una etapa crucial en la fase de diseño del juego basado en grafos. Aquí, se organizarán y refinan las ideas generadas durante la lluvia de ideas para dar forma a conceptos más sólidos y tangibles. Esta fase busca establecer los fundamentos del juego y preparar el terreno para el desarrollo posterior.

1. Seleccionar Conceptos Clave:

Identificación de Conceptos Prometedores:

Revisar las ideas generadas durante la lluvia de ideas.

Seleccionar conceptos que se alineen mejor con los objetivos del juego y la aplicación de algoritmos de grafos.

Combinación de Ideas:

Explorar la posibilidad de combinar elementos de varias ideas para crear un concepto integral y único.

2. Esbozar Mecánicas y Reglas:

Definir Mecánicas de Juego:

Detallar las acciones y operaciones que los jugadores realizarán en el juego.

Establecer las reglas básicas que guiarán la interacción del jugador con el grafo.

Relacionar Mecánicas con Grafos:

Asegurarse de que las mecánicas estén directamente vinculadas a la manipulación del grafo.

Garantizar que la implementación de algoritmos de grafos tenga un impacto significativo en el desarrollo del juego.

3. Diseño de la Interfaz Gráfica de Usuario (GUI):

Bocetos de la Interfaz:

Crear bocetos de la interfaz gráfica que representen visualmente cómo se mostrará el grafo y las interacciones del jugador.

Considerar la disposición de elementos, colores y animaciones.

Iteración de la Interfaz:

Realizar iteraciones en los bocetos en función de la retroalimentación del equipo.

Asegurar que la interfaz sea intuitiva y atractiva para los jugadores.

4. Implementación de Grafos:

Detalles de Implementación:

Refinar la elección de la implementación de grafos, teniendo en cuenta la versatilidad y eficiencia.

Definir la estructura de datos que mejor se adapte a los requisitos del juego.

Prototipado de Grafos:

Desarrollar prototipos del grafo para evaluar su funcionalidad y cómo se integra con las mecánicas del juego.

Realizar ajustes según sea necesario.

5. Detalles de Algoritmos de Grafos:

Integración Coherente:

Definir cómo se aplicarán los algoritmos de grafos en situaciones específicas del juego.

Asegurarse de que los algoritmos se integren de manera coherente con la narrativa y la dinámica del juego.

Consideraciones de Rendimiento:

Evaluar la complejidad y eficiencia de los algoritmos, y ajustar según las limitaciones de recursos.

6. Documentación de Diseño Preliminar:

Esquemas y Diagramas:

Crear diagramas de flujo y esquemas que ilustren la relación entre los elementos del juego y el grafo.

Documentar las mecánicas clave y cómo se implementarán.

Revisión del Diseño Preliminar:

Someter el diseño preliminar a revisiones y comentarios del equipo.

Realizar ajustes según la retroalimentación recibida.

7. Prototipado:

Desarrollo de Prototipos:

Construir prototipos del juego que incorporen los diseños preliminares.

Probar la funcionalidad básica y las interacciones con el grafo.

Retroalimentación Continua:

Recopilar retroalimentación del equipo y realizar ajustes iterativos en el prototipo.

Asegurarse de que la visión del juego esté alineada con los objetivos del proyecto.

La transición de la lluvia de ideas a los diseños preliminares implica un refinamiento cuidadoso de las ideas iniciales para crear una base sólida para el desarrollo del juego. Este proceso establece la dirección y los detalles clave que guiarán el desarrollo posterior del proyecto.

**FASE 5: IMPLEMENTACIÓN DE DISEÑO**

**Implementación en lenguaje de programación (Java)**

Para la implementación del juego basado en grafos, que se describe en el enunciado, se llevará a cabo en el lenguaje de programación Java, siguiendo un enfoque estructurado que abarque las fases de análisis, diseño e implementación. El objetivo es desarrollar un juego interactivo de laberinto para uno o más jugadores, modelado utilizando grafos, con un mínimo de 50 vértices y 50 aristas. Se buscará aplicar al menos dos algoritmos de grafos, seleccionando entre Recorridos sobre Grafos (BFS, DFS), Caminos de Peso Mínimo (Dijkstra, Floyd-Warshall) y Árbol de Recubrimiento Mínimo (Prim, Kruskal).

En la fase de análisis, se define el juego como una experiencia de laberinto donde los jugadores navegan a través de habitaciones conectadas por puertas, modeladas como vértices y aristas de un grafo. Se establece la necesidad de desarrollar dos versiones del grafo para garantizar flexibilidad en la implementación. La representación del laberinto incluirá al menos 50 vértices y 50 aristas para proporcionar un desafío significativo.

En la fase de diseño, se crea un Tipo Abstracto de Datos (TAD) llamado Laberinto que encapsula las operaciones relacionadas con la construcción y manipulación del grafo del laberinto. Se definen diagramas de clase y objetos que representan las entidades clave del sistema, incluyendo el grafo, el jugador, la interfaz gráfica y los algoritmos de grafos seleccionados. Se diseñan casos de prueba para realizar pruebas unitarias automáticas que cubran escenarios críticos del juego, desde la creación del laberinto hasta la aplicación de los algoritmos.

En la fase de implementación, se desarrollan las dos versiones del grafo: una utilizando una Matriz de Adyacencia y la otra utilizando una Lista de Adyacencia. Se implementan los algoritmos de grafos seleccionados (por ejemplo, BFS y Dijkstra) dentro del TAD Laberinto. Además, se crea una interfaz gráfica de usuario (GUI) que permitirá a los jugadores interactuar con el juego visualmente.

Para cumplir con los requerimientos, se establece un enfoque modular y adaptable, permitiendo que el programa admita el cambio entre las dos implementaciones del grafo en cualquier momento. La documentación apropiada, desde la especificación de requerimientos hasta los diagramas de clase y objetos, garantiza una comprensión clara del diseño del sistema.

Este proceso sigue las fases del método de ingeniería de software, asegurando una implementación robusta y alineada con los objetivos del juego. Las pruebas unitarias automáticas se llevarán a cabo para verificar la funcionalidad del sistema, y cualquier ajuste necesario se realizará para garantizar un rendimiento óptimo y una experiencia de juego satisfactoria.

**Complejidad espacial:**

Para analizar la complejidad espacial de la implementación propuesta del juego basado en grafos, consideremos los principales elementos y estructuras de datos involucrados en el diseño.

Matrices de Adyacencia y Listas de Adyacencia:

Ambas implementaciones del grafo (Matriz de Adyacencia y Lista de Adyacencia) tendrán una complejidad espacial de O(|V|^2) y O(|V| + |E|), respectivamente, donde |V| es el número de vértices y |E| es el número de aristas.

En el caso de un mínimo de 50 vértices y 50 aristas, la complejidad espacial será constante para la Lista de Adyacencia (O(100)) y cuadrática para la Matriz de Adyacencia (O(2500)).

Algoritmos de Grafos:

La complejidad espacial de los algoritmos de grafos aplicados (por ejemplo, BFS y Dijkstra) dependerá de la implementación específica, pero en general, será influenciada por la cantidad de nodos y aristas explorados durante la ejecución.

Para BFS, la complejidad espacial será O(|V|) debido al uso de una cola para almacenar nodos.

Para Dijkstra, la complejidad espacial será O(|V|) para el almacenamiento de distancias mínimas y O(|V| + |E|) para la cola de prioridad.

Interfaz Gráfica de Usuario (GUI):

La complejidad espacial de la GUI dependerá del número de elementos gráficos y componentes utilizados. En general, será proporcional a la cantidad de información visualizada en la interfaz.

Si la GUI utiliza imágenes u otros recursos gráficos, la complejidad espacial puede aumentar en función del tamaño y la cantidad de estos recursos.

Estructuras de Datos Adicionales:

La complejidad espacial asociada con otras estructuras de datos utilizadas para mantener información adicional, como la pila de acciones de deshacer o cualquier otra estructura de datos auxiliar, dependerá de su implementación específica.

En resumen, la complejidad espacial general del juego basado en grafos estará dominada por la representación del grafo, con una diferencia significativa entre la complejidad cuadrática de la Matriz de Adyacencia y la complejidad lineal de la Lista de Adyacencia. Los algoritmos de grafos y la GUI también contribuirán a la complejidad espacial total, pero en menor medida en comparación con la representación del grafo. Es fundamental tener en cuenta estos aspectos al evaluar los recursos necesarios para ejecutar el juego en función de la cantidad de vértices y aristas especificados en los requisitos mínimos.

**Complejidad temporal:**

Vamos a analizar la complejidad temporal de las principales operaciones en el contexto del juego basado en grafos implementado en Java.

Construcción del Grafo (Matriz de Adyacencia y Lista de Adyacencia):

Matriz de Adyacencia:

La construcción de una matriz de adyacencia involucra llenar una matriz cuadrada con ceros o unos para representar la presencia o ausencia de aristas entre los vértices. Esto tiene una complejidad temporal de O(|V|^2), donde |V| es el número de vértices.

Lista de Adyacencia:

La construcción de una lista de adyacencia implica inicializar listas enlazadas para cada vértice y agregar los vecinos. La complejidad temporal es O(|V| + |E|), donde |E| es el número de aristas.

Recorridos sobre Grafos (BFS, DFS):

Tanto BFS como DFS visitan cada vértice y cada arista exactamente una vez. La complejidad temporal para ambos es O(|V| + |E|), donde |V| es el número de vértices y |E| es el número de aristas.

Caminos de Peso Mínimo (Dijkstra):

La complejidad temporal de Dijkstra depende de la implementación específica de la cola de prioridad. Con una implementación eficiente de la cola de prioridad (por ejemplo, utilizando un montículo binario o de Fibonacci), la complejidad temporal es O((|V| + |E|) \* log|V|).

Interfaz Gráfica de Usuario (GUI):

La complejidad temporal de la GUI dependerá de la cantidad de elementos gráficos y la complejidad de las interacciones del usuario. En general, la actualización de la interfaz gráfica puede ser O(1) por operación, pero puede depender de la frecuencia de actualización y la complejidad de renderizado.

Estructuras de Datos Adicionales:

La complejidad temporal de otras estructuras de datos utilizadas, como pilas o listas, dependerá de las operaciones específicas que se realicen con ellas.

En resumen, la complejidad temporal general del juego basado en grafos estará dominada por las operaciones de construcción del grafo y la aplicación de algoritmos de grafos, especialmente Dijkstra. La GUI y otras estructuras de datos adicionales tendrán un impacto menor en la complejidad temporal total. Es importante considerar estas complejidades al evaluar el rendimiento del juego, especialmente en situaciones donde el número de vértices y aristas puede aumentar significativamente.