JUEGO SERPIENTES Y ESCALERAS CON TEMATICA DE DISNEY

COMPUTACIÓN Y ESTRUCTURAS DISCRETAS

JUANJO CONTRERAS

DAMY VILLEGAS

JUAN ESTEBAN ERAZO

21/10/2023

**METODO DEL DISEÑO DE INGENIERIA**

DESCRIPCIÓN DEL CONTEXTO PROBLEMÁTICO:

El contexto del problema parece centrarse en el desarrollo de un juego de mesa basado en grafos con elementos desafiantes y entretenidos. Se requiere crear un juego que utilice al menos dos algoritmos de grafos (como BFS, DFS, Dijkstra, Floyd-Warshall, Prim, Kruskal) en un tablero que represente relaciones entre jugadores, elementos del juego o relaciones dentro del mundo del juego.

IDENTIFICACIÓN Y DEFINICIÓN DEL PROBLEMA:

El problema principal es el desarrollo de un juego de mesa basado en un grafo con un mínimo de 50 vértices y 50 aristas. Este juego debe ser desafiante y entretenido para los jugadores, permitiendo movimientos estratégicos de piezas en un tablero donde las aristas y vértices del grafo representen relaciones entre jugadores, elementos del juego o características del mundo del juego.

ESPECIFICACIÓN DE REQUERIMIENTOS FUNCIONALES:

Desarrollo de Grafos (Dos Versiones):

Implementar dos versiones del grafo desde su TAD hasta las pruebas unitarias automáticas.

MÉTODO DE INGENIERÍA PARA LA SOLUCIÓN:

Documentar cada fase del método de ingeniería, desde el análisis hasta la implementación.

DISEÑO DEL JUEGO DE MESA:

El tablero debe representar un mundo desafiante con diferentes tipos de casillas y relaciones entre ellas.

Las aristas pueden representar relaciones entre jugadores, elementos del juego o características del mundo del juego.

DESARROLLO DE LAS IDEAS DEL JUEGO:

* UTILIZACIÓN DE LAS ARISTAS:

Representación de relaciones entre jugadores, elementos del juego o características del mundo del juego.

* POSIBLES EJEMPLOS DE USO DE ARISTAS:

1. Amistades, alianzas, enemistades entre jugadores.
2. Relaciones comerciales, románticas o de poder entre jugadores y elementos del juego.
3. Relaciones entre ciudades, personajes u objetos dentro del mundo del juego.

SÍNTESIS:

El desafío consiste en crear un juego de mesa innovador y desafiante basado en grafos, donde los jugadores puedan mover sus piezas por un tablero representado por un grafo. Las relaciones entre los nodos y aristas del grafo pueden representar varios elementos del mundo del juego, como relaciones entre jugadores, elementos o características especiales del mundo del juego.

DEFINICION DE NUESTRO JUEGO A IMPLEMENTAR

Nuestro proyecto por desarrollar siguiendo todas las reglas del enunciado, se relaciona con el juego de serpientes y escaleras, por nuestro gusto decidimos hacerlo basándonos en una temática en especial, la cual es Disney, nuestra idea a plantear es la siguiente

IDEA

El juego a través de una interfaz grafica se desarrolla en un tablero que representa un mundo de fantasía de Disney, donde los jugadores se mueven por el tablero usando dados para avanzar.

OBJETIVO

El objetivo del juego es llegar al meta primero. La meta está ubicada en la casilla 30.

MOVIMIENTOS

Los jugadores se mueven por el tablero usando un dado para avanzar. Cada jugador tira el dado en su turno y avanza la cantidad de espacios que indica el resultado. EL dado contiene los números del 1 al 6.

RELACION CON EL JUEGO SERPIENTES Y ESCALERAS

El tablero tiene serpientes y escaleras que pueden mover a los jugadores hacia atrás o hacia adelante. Las serpientes representan obstáculos que los jugadores deben superar, mientras que las escaleras representan oportunidades que los jugadores pueden aprovechar.

SERPIENTES Y ESCALERAS:

El tablero tiene 14 serpientes y 14 escaleras. Las serpientes representan obstáculos que los jugadores deben superar, mientras que las escaleras representan oportunidades que los jugadores pueden aprovechar.

PERSONAJES DE DISNEY

Las casillas del tablero representan lugares y personajes de Disney. Los jugadores a medida de su avance en el juego pueden encontrarse con diferentes objetos de Disney que les dan poderes especiales.

ALGORITMOS

Con la implementación de nuestros algoritmos de grafos los jugadores los podrán usar para completar sus misiones y ganar el juego, por ejemplo:

* BFS: Algoritmo que ayudaría a explorar el tablero de manera eficiente y encontrar la siguiente casilla disponible.
* DFS: Algoritmo que ayudaría a encontrar un camino específico a la meta.
* Caminos de peso mínimo: Ayudaría a encontrar el camino más seguro o eficiente a la meta.

Para realizar este proyecto y al debido a la gran extensiva de temáticas que propone Disney, decidimos guiarnos con la película de Frozen, un ejemplo claro sería el siguiente:

El tablero representaría el reino de Arendelle de la película Frozen. Las casillas del tablero podrían representar lugares como el castillo de Arendelle, el bosque encantado, y el glaciar Northuldra. Las serpientes y escaleras podrían representar obstáculos como el océano congelado, el fuego del dragón Marshmallow, y la tormenta de nieve. Los personajes para escoger podrían ser Elsa, Anna, Olaf, Kristoff, y Sven.

DETALLER ADICIONALES

1. Numero de Jugadores: Nuestro juego puede tener de 2 a 4 jugadores.
2. Duración del Juego: El juego puede durar de 30 a 60 minutos, dependiendo de la cantidad de jugadores y de el tiempo que empleen estos mismos.
3. Edad Recomendada: El juego es recomendado para niños de 6 años en adelante.
4. Numero de Serpientes y Escaleras: El numero de serpientes y escaleras en nuestro juego va a ser de 5 y 5.
5. Objetos que otorgan poderes especiales a los jugadores: El numero de objetos en el juego son 5 y pueden variar para asi darles a los jugadores diferentes ventajas.

REGLAS DEL JUEGO:

* Los jugadores se turnan para tirar un dado y avanzar sus fichas por el tablero.
* Si una ficha cae en una casilla con una escalera, sube automáticamente a una casilla superior, lo que le da al jugador cierto tipo de ventaja.
* Si cae en una casilla con una serpiente, desciende a una casilla inferior, lo que le da al jugador cierto tipo de desventaja.
* Gana el primer jugador que llegue a la casilla final.
* Cada jugador con la implementación de aristas tiene relaciones diferentes con objetos que se encuentran a lo largo del tablero.

CASILLAS:

Las casillas del tablero están numeradas del 1 al 30.

Además, ten en cuenta los siguientes eventos especiales que ocurren en las casillas señaladas:

ESCALERAS:

**Casilla 3:** Elsa. La magia de Elsa eleva a los jugadores desde la casilla 22 hasta la 30, acercándolos a la meta. El jugador avanza a la casilla 13.

**Casilla 5:** Olaf. El jugador sube por el abrazo cálido de Olaf, avanzando a la casilla 10.

**Casilla 8:** Sven. Los jugadores avanzan desde la casilla 8 hasta la 17, ayudados por Sven.

**Casilla 14:** Anna. La determinación de Anna impulsa a los jugadores desde la casilla 14 hasta la 21.

SERPIENTES:

**Casilla 12:** Monstruos de nieve de Elsa. Los jugadores retroceden desde la casilla 12 hasta la 2 al ser perseguidos por los monstruos que rodean el castillo.

**Casilla 19:** Hans. Los jugadores retroceden desde la casilla 19 hasta la 6 debido a las malas intenciones de Hans.

**Casilla 25:** Los lobos. El encuentro con los lobos lleva a los jugadores desde la casilla 25 hasta la 15.

**Casilla 29:** Invierno eterno. El invierno eterno de Elsa hace retroceder a los jugadores desde la casilla 29 hasta la 16.

EVENTOS ESPECIALES:

**Casilla 9:** Minijuego de construir un muñeco de nieve con Olaf. El jugador lanza el dado de nuevo (pero no avanza lo que obtenga en el lanzamiento). Si saca par, avanza 3 casillas.

**Casilla 18:** Elsa congela el camino. Pierdes un turno mientras esperas a que el hielo se derrita.

**Casilla 24:** Desafío de patinaje sobre hielo en el lago congelado. El sistema arroja un número al azar (entre 1 y 6) y el jugador lanza el dado de nuevo (pero no avanza lo que obtenga en el lanzamiento), si saca un número igual o mayor al número arrojado por el sistema, gana un lanzamiento de dado extra de inmediato. De lo contrario, pierde un turno.

**Casilla 27:** Encuentro con los trolls. El jugador lanza el dado (pero no avanza lo que obtenga en el lanzamiento), si saca un número impar retrocede 4 casillas.

ESPECIFICACION DE REQUERIMIENTOS

1. Tablero y Grafos

* Creación del Tablero: Diseño de un tablero gráfico que represente el mundo de Disney (Arendelle de Frozen) con al menos 50 vértices (casillas) y 50 aristas (conexiones entre casillas).
* Implementación del Grafo: Desarrollo de una estructura de grafo que represente las relaciones entre las casillas del tablero. Los vértices representarán ubicaciones o personajes, y las aristas las conexiones entre ellos.

2. Interfaz Gráfica y Movimientos

* Desarrollo de Interfaz Gráfica: Creación de una interfaz visual que muestre el tablero y permita a los jugadores interactuar con él.
* Movimiento de Jugadores: Implementación de la mecánica de movimiento utilizando dados, permitiendo que los jugadores avancen por el tablero según el resultado del lanzamiento de los dados.

3. Serpientes y Escaleras:

* Incorporación de Serpientes y Escaleras: Colocación de al menos 5 serpientes y 5 escaleras en el tablero, representando obstáculos y oportunidades respectivamente para los jugadores.

4. Personajes y Objetos de Disney:

* Representación de Personajes y Lugares: Asignación de personajes y ubicaciones de Disney a las casillas del tablero.
* Objetos Especiales: Incorporación de al menos 5 objetos especiales (poderes) en ciertas casillas, otorgando ventajas a los jugadores que las alcancen.

5. Algoritmos de Grafos:

* Implementación de Algoritmos: Desarrollo de al menos tres algoritmos de grafos (BFS, DFS, Caminos de Peso Mínimo) para que los jugadores los utilicen estratégicamente durante el juego.

6. Reglas y Mecánicas del Juego:

* Definición de Reglas: Establecimiento de las reglas del juego, incluyendo el turno de juego, efectos de las serpientes y escaleras, condiciones de victoria y uso de poderes especiales.

7. Jugabilidad y Testing:

* Pruebas y Depuración: Realización de pruebas exhaustivas para garantizar la jugabilidad, corrección y fluidez del juego.
* Iteraciones y Mejoras: Realización de ajustes y mejoras basadas en los resultados de las pruebas y el feedback de los jugadores beta.

RECOPILACION DE LA INFORMACION NECESARIA

Para el desarrollo de este proyecto se necesita de una recopilación de información:

1. Algoritmos de Grafos

Los algoritmos de grafos son un conjunto de instrucciones que recorren (visitan los nodos de) un grafo. Algunos algoritmos son usados para hallar un nodo específico o el camino entre dos nodos dados. Los grafos son estructuras de datos muy útiles que se usan para modelar varios problemas. Estos algoritmos tienen aplicaciones directas en sitios de redes sociales, modelado de máquinas de estado y más. Algunos de los algoritmos de grafos más comunes son:

* Búsqueda en Amplitud o Anchura (Breadth First Search, BFS):

Búsqueda en Amplitud o Anchura es uno de los algoritmos de grafos más sencillo. Recorre el grafo al primero comprobar el nodo actual y luego expandirlo al agregar sus sucesores al siguiente nivel. El proceso se repite para todos los nodos del nivel actual antes de moverse al siguiente. Si se encuentra la solución, la búsqueda termina.

* Búsqueda en Profundidad (Depth First Search, DFS):

La Búsqueda en Profundidad es uno de los algoritmos de grafos más sencillos. Recorre el grafo revisando primero el nodo actual y moviéndose después a uno de sus sucesores para repetir el proceso. Si el nodo actual no tiene sucesor a revisar, regresamos a su predecesor y el proceso continúa (moviéndose a otro sucesor). Si la solución es encontrada, la búsqueda termina.

* Dijkstra

El Algoritmo de Dijkstra es un algoritmo de grafo presentado por E. W. Dijkstra. Encuentra el camino más corto de origen único en un grafo con aristas no negativas.

Creamos dos arreglos: visitado y distancia, que registran si un vértice es visitado y cuál es la mínima distancia desde el vértice origen, respectivamente. Inicialmente, el arreglo visitado se asigna como falso y distancia como infinito.

Partimos del vértice origen. Dejemos que el vértice actual sea u y sus vértices adyacentes sean v. Ahora, por cada v que es adyacente a u, la distancia se actualiza si no ha sido visitado antes y la distancia desde u es menor que su distancia actual. Luego seleccionamos el siguiente vértice con la menor distancia y que no haya sido visitado.

La Cola de Prioridad se usa a menudo para cumplir este último requerimiento en el menor tiempo posible. Abajo se ve una implementación de la misma idea usando la cola de prioridad en Java.

* Algoritmo de Floyd-Warshall:

El algoritmo de Floyd Warshall es un gran algoritmo para encontrar la distancia más corta entre todos los vértices del grafo. Tiene un algoritmo muy conciso y complejidad del tiempo O(V^3) (donde V es el número de vértices). Puede ser usado con pesos negativos, aunque los ciclos de pesos negativos no deben estar presentes en el grafo.

Fuente: <https://www.freecodecamp.org/espanol/news/explicacion-de-algoritmos-y-estructuras-de-datos-de-grafos-con-ejemplos-en-java-y-c/#:~:text=%C2%BFQu%C3%A9%20es%20un%20Algoritmo%20de,camino%20entre%20dos%20nodos%20dados>.

1. Grafos, Vértices y Aristas

GRAFOS

Un grafo consiste en un conjunto finito de puntos llamados vértices y un conjunto finito de aristas, cada una de las cuales conecta dos vértices.

Se dice que dos vértices son adyacentes, si están conectados por una arista.

Matriz de adyacencia: Dado un grafo G que tiene n vértices, su matriz de adyacencia es la matriz A(G) de tamaño n × n definida por:

Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja

La anterior definición es apropiada solamente para los grafos simples. Un grafo es simple si no existen múltiples aristas que conectan a una pareja de vértices.

Fuente: <https://ciencias.medellin.unal.edu.co/cursos/algebra-lineal/clases/8-clases/40-clase-2-parte3.html#:~:text=Un%20grafo%20consiste%20de%20un,est%C3%A1n%20conectados%20por%20una%20arista>.

VERTICES

El grado de un vértice en un grafo es el número de aristas incidentes a él. Un vértice aislado es un vértice con grado cero; esto es, un vértice que no es punto final de ninguna arista. Un vértice hoja es un vértice con grado uno. En un grafo dirigido, se puede distinguir entre grado de salida ("outdegree", número de aristas que salen del vértice) y grado de entrada ("indegree", número de aristas que llegan al vértice); un vértice fuente es un vértice con grado de entrada cero, mientras que un sumidero es un vértice con grado de salida cero.

Un vértice de corte es un vértice que al removerlo desconecta al grafo restante. Un conjunto independiente es un conjunto de vértices tal que ninguno es adyacente a otro, y una cobertura de vértices es un conjunto de vértices que incluye los puntos finales de cada arista en un grafo.

La vecindad de un vértice x, denotado como N(x) está dado por todos los vértices adyacentes a x.

Fuente: <https://es.wikipedia.org/wiki/V%C3%A9rtice_(teor%C3%ADa_de_grafos)#:~:text=En%20teor%C3%ADa%20de%20grafos%2C%20un,que%20est%C3%A1n%20formados%20los%20grafos>.

ARISTAS

Son las líneas que unen los vértices de un grafo. Hay diferentes tipos de aristas:

* Aristas adyacentes: Dos aristas son adyacentes si convergen en el mismo vértice.
* Aristas paralelas: Dos aristas son paralelas si los vértices iniciales y finales son el mismo vértice
* Aristas cíclicas: Aristas que parten de un vértice para entrar en el mismo.

Términos importantes

* Cruce: Punto donde dos aristas se cruzan.
* Vértices: Los vértices son los elementos que forman un grafo. Cada uno lleva asociada una valencia característica según la situación, que se corresponde con la cantidad de aristas que confluyen en dicho vértice.
* Camino: Se denomina camino a un conjunto de vértices interconectados por aristas. Dos vértices están conectados si hay un camino entre ellos.

Fuente: <https://es.wikipedia.org/wiki/Teor%C3%ADa_de_grafos#:~:text=Aristas%3A%20Son%20las%20l%C3%ADneas%20que,convergen%20en%20el%20mismo%20v%C3%A9rtice.&text=Aristas%20c%C3%ADclicas%3A%20Aristas%20que%20parten,para%20entrar%20en%20el%20mismo>.

1. Implementación de una Interfaz Grafica

Interfaz Gráfica de Usuario, interfaz de usuario o GUI (Graphical User Interface): es el entorno de objetos gráficos disponibles para un usuario en el marco de una aplicación o sistema operativo. El sistema operativo MS-Dos se basaba en intérpretes de comando (escritura de instrucciones por consola) pero Windows se basa en una interfaz gráfica de usuario (su entorno de escritorio), Linux en otra y Macintosh en otra.

Herramientas para crear Interfaces gráficas de usuario en Java. Hacemos referencia principalmente a los paquetes (packages) del API de Java swing y awt (Abstract Windowing Toolkit). Las clases de estos paquetes permiten crear interfaces gráficas de usuario basadas en ventanas estilo “Windows” para nuestras aplicaciones.

Fuente: <https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=644:concepto-o-definicion-de-interface-java-ique-es-una-interface-tipos-de-interfaz-ejemplo-practico-cu00677b&catid=68&Itemid=188#:~:text=c)%20Interfaz%20Gr%C3%A1fica%20de%20Usuario,una%20aplicaci%C3%B3n%20o%20sistema%20operativo>.

1. Listas de Adyacencia

Una lista de adyacencia es una estructura de datos bidimensional que permite búsquedas e inserciones eficientes y, a su vez, es óptima en el uso de memoria. Uno de sus principales usos es en la representación de relaciones binarias (o grafos) y la representación de listas con prioridad.

En particular, la estructura de datos lista de adyacencia ofrece las siguientes operaciones:

* Init (): crea una lista de adyacencia vacía (complejidad temporal O (1)).
* Eq (other): determina si la lista de adyacencia es igual a other (complejidad temporal O(n), en donde n es la cantidad de elementos en la lista de adyacencia).
* str (): representación en cadena de la lista de adyacencia (complejidad temporal O(n), en donde n es la cantidad de elementos en la lista de adyacencia).
* Insert (x, y): inserta el elemento y con prioridad x (complejidad temporal O (log2 n), en donde n es la cantidad de elementos en la lista de adyacencia).
* Contains (x, y): retorna True si el elemento y con prioridad x está en la lista de adyacencia; False de lo contrario (complejidad temporal O (log2 n), en donde n es la cantidad de elementos en la lista de adyacencia).
* Remove (x, y): elimina el elemento y con prioridad x de la lista de adyacencia, si este existe (complejidad temporal O (log2 n), en donde n es la cantidad de elementos en la lista de adyacencia).

Fuente: <https://www.camilorocha.info/teaching/pimo/2013-1/laboratorios/laboratorio8listasdeadyacencia>

1. Matriz de Adyacencia

Es una matriz booleana que representa las conexiones entre pares de vértices. La matriz de adyacencia de un grafo es simétrica. Si un vértice es aislado entonces la correspondiente fila (columna) está compuesta sólo por ceros. Si el grafo es simple entonces la matriz de adyacencia contiene solo ceros y unos (matriz binaria) y la diagonal está compuesta sólo por ceros.

La matriz de adyacencia de un dígrafo no es simétrica. Es una matriz binaria. El número de unos que aparecen en una fila es igual al grado de salida del correspondiente vértice y el número de unos que aparecen en una determinada columna es igual al grado de entrada del correspondiente vértice.

Algoritmo para construir la matriz de adyacencia de un grafo.

* Paso 1: Se asignan un orden arbitrario a los vértices.
* Paso 2: Se construye una matriz de dimensión n\*n, cardinalidad (# vértices) por cardinalidad (# vértices).
* Paso 3: En la posición I j-ésima se coloca 1 si el vértice i es adyacente al vértice j; 0 en el otro caso

Ejemplo:

Imagen que contiene Diagrama

Descripción generada automáticamente

Fuente: <https://www.utm.mx/~mgarcia/ED4(Grafos).pdf>

1. Pruebas Unitarias Automáticas

Las pruebas automatizadas consisten en la aplicación de herramientas de software para automatizar el proceso manual de revisión y validación de un producto de software que lleva a cabo una persona. Ahora, la mayoría de los proyectos de software ágiles y de DevOps modernos incluyen pruebas automatizadas desde el principio; sin embargo, para apreciar plenamente el valor de dichas pruebas, hay que saber cómo era la vida antes de que se adoptaran de forma generalizada.

¿Por qué son importantes?

La entrega continua (CD) consiste en publicar versiones de código nuevas lo más rápido posible para los clientes. Las pruebas automatizadas resultan fundamentales para alcanzar ese objetivo. No hay forma de automatizar dicha publicación si hay un paso manual que requiere mucho tiempo en el proceso de publicación.

La CD forma parte de un proceso de implementación mayor. Es sucesora de la integración continua (CI) y depende de ella. La CI es totalmente responsable de ejecutar pruebas automatizadas ante cualquier cambio de código nuevo y de verificar que dichos cambios no afectan a la integridad de las funciones establecidas ni introducen errores nuevos. La CD se activa una vez que el paso de integración continua supera el plan de pruebas automatizado.

Esta relación entre las pruebas automatizadas, la CI y la CD aporta numerosas ventajas a los equipos de software que trabajan a gran velocidad. Las pruebas automatizadas garantizan la calidad en todas las fases del desarrollo, ya que aseguran que las confirmaciones nuevas no introducen ningún error, por lo que el software sigue estando listo para implementarse en todo momento.

Fuente: <https://www.atlassian.com/es/continuous-delivery/software-testing/automated-testing#:~:text=Las%20pruebas%20automatizadas%20consisten%20en,lleva%20a%20cabo%20una%20persona>.

BUSQUEDA DE SOLUCIONES CREATIVAS

1. Tablero y Grafos

Creación del Tablero:

* Tablero Tridimensional: Si el tablero se crea en tres dimensiones con ciertas áreas interactivas para los jugadores podría representar mejor el mundo de Frozen, permitiendo una experiencia más inmersiva, este podría ser implementado utilizando matrices tridimensionales, por ejemplo, se puede usar una matriz tridimensional para representar un cubo o cualquier otra forma tridimensional.
* Tablero Modular: Crear un tablero que pueda modificarse durante el juego, con secciones intercambiables que representen diferentes áreas de Arendelle y así ofrecer una experiencia variable en cada partida, esto requiere la utilización de una cuadricula o matriz que puede tener tamaño y variables, permitiendo la colocación de elementos en diferentes posiciones.

Implementación del Grafo:

* Grafo Dinámico: Diseñar un grafo que cambie dinámicamente durante el juego según las acciones de los jugadores, por ejemplo, si un jugador activa un poder especial que modifica las conexiones entre ciertas casillas, implica usar una combinación de clases y estructuras de datos adecuadas para representar los vértices, las aristas y las conexiones entre ellos.
* Grafo Interactivo: Permitir a los jugadores agregar temporalmente nuevas aristas o vértices durante el juego al alcanzar ciertas condiciones, brindando una experiencia más participativa en la construcción del mundo del juego, para implementar esto se pueden utilizar bibliotecas graficas como JavaFX, para la interfaz gráfica y representar el grafo mediante componentes gráficos como nodos y líneas.

1. Interfaz Gráfica y Movimientos

Desarrollo de la Interfaz Gráfica:

* Realidad Aumentada: Crear una interfaz de realidad aumentada que integre el tablero físico con elementos digitales, permitiendo a los jugadores ver animaciones y efectos especiales al interactuar con el tablero, esto puede realizarse utilizando bibliotecas específicas para manipulas la cámara y los gráficos en tiempo real, como Java3D.
* Personalización de Interfaz: Ofrecer opciones de personalización en la interfaz para los jugadores, como temas visuales basados en otras películas de Disney, permitiéndoles cambiar la apariencia del tablero, esto puede realizarse utilizando la biblioteca Swing, que es una biblioteca GUI estándar en Java.

Movimiento de los Jugadores:

* Movimientos Especiales: Implementar movimientos especiales adicionales que no sean solo avanzar, como la capacidad de retroceder ciertos pasos estratégicamente, ofreciendo un toque más estratégico al juego, esto se puede realizar utilizando la implementación de un método y actualizando la posición en la que se encuentra el jugador.
* Control de Movimiento: Permitir a los jugadores elegir entre diferentes opciones de movimiento después de tirar los dados, como tomar un camino más largo, pero más seguro o uno corto pero arriesgado, esto puede crearse al utilizar bibliotecas como Swing o JavaFX las cuales son utilizadas para crear interfaces graficas.

DESCARTE DE IDEAS NO VIABLES

DESCARTE DE IDEAS NO VIABLES

1. DESCARTE:

Realidad Aumentada (RA) con Java3D.

* Justificación:

Aunque la RA podría ser atractiva, puede resultar costosa y compleja de implementar para un juego de mesa. Requiere hardware específico y el coste adicional puede alejar a los usuarios.

IDEAS SELECCIONADAS

1. Tablero Modular:

* Explicación: Un tablero modificable añade versatilidad al juego y permite variaciones en la experiencia de juego. Una cuadrícula o matriz ajustable podría representar diferentes áreas de Arendelle.
* Diseño Preliminar: Utilizar una cuadrícula de tamaño variable con secciones intercambiables que representen lugares clave de Arendelle. Las secciones se conectarían fácilmente mediante un sistema de enganche para permitir la modificación del tablero.

1. Grafo Interactivo:

* Explicación: Permite a los jugadores agregar nuevas conexiones entre áreas o elementos del juego, lo que fomenta la participación y la construcción del mundo del juego.
* Diseño Preliminar: Implementar un sistema donde los jugadores puedan agregar temporalmente nuevas conexiones entre áreas del tablero al cumplir ciertas condiciones, utilizando componentes gráficos como nodos y líneas para representar estas conexiones.

1. Movimientos Especiales:

* Explicación: Agregar la capacidad de movimientos estratégicos adicionales más allá de avanzar, como retroceder pasos para añadir una capa estratégica al juego.
* Diseño Preliminar: Incorporar un conjunto de acciones especiales, como retroceder ciertos pasos en el tablero, que los jugadores puedan usar estratégicamente una vez por turno.

DISEÑOS PRELIMINARES

1. Tablero Modular:

* Secciones intercambiables: Módulos de diferentes áreas (castillo, bosque, montaña) que se conectan mediante un sistema de ranuras para crear un mapa variable.
* Materiales: Piezas de plástico moldeable para crear los módulos y una base con ranuras para su intercambio.

1. Grafo Interactivo:

* Representación visual: Utilizar nodos y líneas gráficas que los jugadores puedan colocar entre áreas del tablero.
* Mecánica: Al alcanzar ciertos objetivos, los jugadores reciben piezas que pueden usar para crear nuevas conexiones entre áreas.

1. Movimientos Especiales

* Tarjetas de Movimiento: Incorporar tarjetas de acciones especiales que los jugadores puedan usar estratégicamente una vez por turno.
* Mecánica: Los jugadores pueden elegir retroceder un número específico de espacios en el tablero usando estas tarjetas.

EVALUACION Y SELECCIÓN DE LA MEJOR SOLUCION

1. Versatilidad del Juego:

* Escala: Baja (poca variabilidad) a Alta (mucha variabilidad).
* Tablero Modular: Alta - Permite múltiples configuraciones y variaciones en la disposición del juego.
* Grafo Interactivo: Media - Ofrece variabilidad mediante la creación de nuevas conexiones, pero menos que un tablero modular.
* Movimientos Especiales: Baja - Aporta estrategia, pero no cambia la disposición física del juego.

1. Involucramiento de los jugadores:

* Escala: Bajo (poco involucramiento) a Alto (máximo involucramiento).
* Tablero Modular: Media - Involucra a los jugadores en la creación del mapa, pero no constantemente.
* Grafo Interactivo: Alto - Permite a los jugadores cambiar activamente el juego durante su turno.
* Movimientos Especiales: Media - Añade opciones estratégicas, pero no implica cambios directos en el tablero.

1. Complejidad de Implementación:

* Escala: Baja (menos complejidad) a Alta (más complejidad).
* Tablero Modular: Media - Requiere diseño de módulos intercambiables y sistema de conexión.
* Grafo Interactivo: Alta - La representación gráfica y la interacción con nodos y líneas pueden ser complejas.
* Movimientos Especiales: Baja - Implementación de tarjetas de acción y reglas claras para su uso.

1. Experiencia Inmersiva

* Escala: Baja (poca inmersión) a Alta (mucha inmersión).
* Tablero Modular: Media - Ofrece cierta inmersión al cambiar la disposición física del juego.
* Grafo Interactivo: Alta - Permite a los jugadores influir directamente en la estructura del mundo del juego.
* Movimientos Especiales: Baja - Aporta estrategia, pero no afecta directamente la inmersión.

EVALUACION

* Tablero Modular: Versatilidad (Alta), Involucramiento (Media), Complejidad (Media), Inmersión (Media).
* Grafo Interactivo: Versatilidad (Media), Involucramiento (Alto), Complejidad (Alta), Inmersión (Alta).
* Movimientos Especiales: Versatilidad (Baja), Involucramiento (Media), Complejidad (Baja), Inmersión (Baja).

TOTAL, DE LA EVALUACION:

* Tablero Modular: 3 + 2 + 2 + 2 = 9
* Grafo Interactivo: 2 + 3 + 3 + 3 = 11
* Movimientos Especiales: 1 + 2 + 1 + 1 = 5

Según esta evaluación, podemos decir que el Grafo interactivo tiene la puntuación más alta y será la mejor idea a implementar que brinde una experiencia mas variada, participativa y sumergida en el juego de serpientes y escaleras con temática de Disney-Frozen.