Instituto Tecnológico de Costa Rica

Escuela de Ingeniería En Computadores

Algoritmos y Estructuras de Datos I

Proyecto I

Video Juego: Tron Legacy

Estudiante: Brainer Chacón Orellana

Profesor: Leonardo Araya Martínez

Código: CE-1103

Tipo de curso: Teórico

II Semestre 2024

Introducción:

Este juego está inspirado en la película *Tron: Legacy*, donde las motos de luz son el núcleo central de la jugabilidad. En el juego, cada moto deja una estela de luz a su paso, la cual actúa como una barrera destructiva para cualquier otra moto que la toque. Las motos deben recorrer el mapa, esquivando tanto sus propias estelas como las de las demás motos para evitar ser destruidas.

El juego también introduce elementos estratégicos mediante el uso de poderes e ítems. Estos ítems, que proporcionan habilidades especiales o mejoras temporales, aparecen de manera aleatoria en el mapa, y las motos pueden recogerlos y almacenarlos. Al usar estos poderes, los jugadores pueden obtener ventajas adicionales.

Cuando una moto es destruida, los ítems y poderes no utilizados vuelven al mapa, generando nuevas oportunidades para los jugadores restantes. Esto añade un componente dinámico, donde los jugadores pueden cambiar el curso del juego no solo mediante su habilidad en el manejo de la moto, sino también en la gestión estratégica de los poderes.

Este sistema de combate, que combina velocidad, reflejos y tácticas, busca capturar la emoción y el ambiente visual de *Tron: Legacy*, mientras que los elementos aleatorios y las decisiones estratégicas elevan la complejidad y el desafío del juego.

Tabla de Contenidos:

Breve Descripción Del Problema:

Este problema plantea el desarrollo de un juego inspirado en "Tron: Legacy", en el cual los jugadores controlan motos de luz que se mueven por una malla o grid, dejando una estela destructiva a su paso. El objetivo es crear un sistema donde las motos puedan moverse de manera continua en cuatro direcciones y destruirse si colisionan con la estela de otra moto o si se quedan sin combustible. Además, las motos nunca se detienen; los jugadores solo pueden cambiar su dirección.

La solución requiere la implementación de varias estructuras de datos lineales como listas, pilas y colas para gestionar el comportamiento y los atributos de las motos. Las motos, por ejemplo, se representan mediante listas enlazadas simples, lo que facilita la manipulación de su estela, que inicialmente tiene tres posiciones y se expande con el tiempo. Las pilas se utilizan para manejar los poderes temporales que los jugadores pueden recolectar, permitiendo que el jugador elija cuándo aplicarlos y en qué orden. Los ítems, por su parte, se gestionan a través de colas, aplicándose en el orden en que se obtienen.

El juego también debe gestionar diversos atributos de las motos, como la velocidad, el tamaño de la estela, y el nivel de combustible, que varían durante el transcurso de la partida. Ítems como celdas de combustible y crecimiento de estela se generan aleatoriamente en el mapa y afectan estos atributos. Los poderes, como escudos temporales o hiper velocidad, ofrecen ventajas tácticas durante un tiempo limitado y también son recogidos en el mapa.

Otro aspecto importante del problema es el manejo de la destrucción de las motos. Cuando una moto se destruye, los ítems y poderes que no fueron utilizados deben volver al mapa en posiciones aleatorias, manteniendo el dinamismo del juego. También se debe implementar la aparición de bots, que actúan como jugadores controlados por la IA y siguen las mismas reglas que los jugadores humanos, aportando un desafío adicional.

Breve Descripción De La Solución:

1. Requerimiento 01: Implementación de la Moto y la Estela

Descripción del Problema:

Las motos de luz se deben implementar utilizando una lista enlazada simple. La moto debe dejar una estela destructiva a su paso, y su movimiento debe asemejarse al de una oruga, es decir, la cabeza de la moto avanza mientras que la cola sigue el rastro. Cuando la moto se crea, inicialmente tiene una estela de 3 posiciones.

Implementación:

- **Clase LightCycle:**
  - **Lista Enlazada Simple:** La moto de luz se representa como una lista enlazada simple, donde cada nodo de la lista representa una posición en la malla del juego. La cabeza de la lista representa la posición actual de la moto, y los nodos subsiguientes representan las posiciones anteriores ocupadas por la moto, formando la estela.
  - **Estela Inicial:** Al crear una moto de luz, se inicializa una lista enlazada con tres nodos, representando la estela inicial de la moto.
  - **Movimiento:** Cuando la moto se mueve, se agrega un nuevo nodo al inicio de la lista para representar la nueva posición de la moto, y se elimina el último nodo para mantener el tamaño constante de la estela.

Alternativas Consideradas:

- **Array Dinámico:** Se consideró utilizar un array dinámico para manejar la estela, pero se descartó en favor de una lista enlazada simple debido a la naturaleza dinámica y continua del movimiento de la moto, que se adapta mejor a una lista enlazada.

Limitaciones:

- **Colisiones:** Actualmente, el código no implementa detección de colisiones complejas, lo que podría permitir que la moto se mueva a través de otros objetos en el grid si no se maneja correctamente.

Problemas Encontrados:

- **Manejo del Movimiento:** Hubo dificultades iniciales en asegurar que el movimiento de la moto se representara correctamente en la interfaz gráfica, específicamente en mantener la estela intacta mientras se actualizan las posiciones.

2. Requerimiento 02: Atributos de las Motos

Descripción del Problema:

Cada moto debe tener varios atributos:

- **Velocidad:** Un valor aleatorio entre 1 y 10 que determina qué tan rápido se mueve la moto.
- **Tamaño de la Estela:** El largo de la estela, que inicialmente debe ser 3.
- **Combustible:** Un valor entre 0 y 100 que determina cuánto combustible tiene la moto. El combustible se consume automáticamente dependiendo de la velocidad a una tasa de 1 celda de combustible por cada 5 elementos de la malla recorridos.
- **Items y Poderes:** La moto debe tener una cola de ítems que afecten permanentemente y una pila de poderes que afecten temporalmente.

Implementación:

- **Clase LightCycle:**
  - **Atributos de la Moto:** Se agregaron propiedades para la Velocidad, Tamaño Estela, y Combustible. La velocidad se genera aleatoriamente al crear la moto, y el tamaño de la estela se establece inicialmente en 3.
  - **Consumo de Combustible:** Se implementó un mecanismo en el método de movimiento de la moto para reducir el valor de combustible basado en la distancia recorrida y la velocidad.
  - **Items y Poderes:** Se implementaron estructuras de datos adicionales (cola y pila) para manejar ítems y poderes, respectivamente. La cola (Queue) maneja los ítems que afectan la

moto permanentemente, mientras que la pila (Stack) maneja los poderes temporales.

Alternativas Consideradas:

- **Algoritmo de Consumo de Combustible:** Se consideró un algoritmo más complejo para el consumo de combustible basado en la dirección del movimiento o la cantidad de estela, pero se optó por un sistema más sencillo basado en la distancia recorrida.

Limitaciones:

- **Interacción entre Ítems y Poderes:** La implementación inicial no considera todas las posibles interacciones entre ítems y poderes, lo que podría llevar a inconsistencias en el comportamiento del juego.

Problemas Encontrados:

- **Sincronización de Ítems y Poderes:** Hubo dificultades iniciales en coordinar la aplicación de ítems y poderes en el tiempo adecuado, especialmente cuando se combinan efectos.

3. Requerimiento 03: Manejo de Ítems y Poderes al Destruir una Moto

Descripción del Problema:

Cuando una moto se destruye, los ítems y poderes que tenía sin usar deben colocarse en el mapa en posiciones aleatorias.

Implementación:

- **Destrucción de la Moto:** Cuando una moto se destruye (por ejemplo, al chocar con otra moto o con una estela), se activa un evento que libera todos los ítems y poderes no usados en posiciones aleatorias en el grid.
- **Generación Aleatoria de Posiciones:** Se implementó una función para generar posiciones aleatorias dentro del grid, asegurando que los ítems y poderes no caigan en posiciones ocupadas por estelas u otras motos.

Alternativas Consideradas:

- **Respawn de Ítems:** Se consideró hacer que los ítems y poderes no usados se eliminen del juego en lugar de reaparecer, pero esto se descartó para mantener la dinámica del juego.

Limitaciones:

- **Posicionamiento Aleatorio:** El posicionamiento aleatorio puede hacer que algunos ítems o poderes aparezcan en zonas de difícil acceso o donde no sean recogidos, lo que podría afectar el equilibrio del juego.

Problemas Encontrados:

- **Colisiones en el Mapa:** Asegurar que los ítems y poderes no se generen en posiciones ocupadas fue un reto inicial.

4. Requerimiento 04: Manejo de la Pila de Poderes

Descripción del Problema:

El jugador debe poder escoger cuándo ejecutar los poderes, que se ejecutan en un orden definido por el jugador. En pantalla, el jugador podrá ver la pila de poderes y reordenarla antes de usarlos.

Implementación:

- **Interfaz para Poderes:** Se implementó una sección en la interfaz gráfica para mostrar la pila de poderes. El jugador puede seleccionar y reordenar los poderes antes de ejecutarlos.
- **Ejecución de Poderes:** Cuando el jugador decide usar un poder, se extrae el elemento del tope de la pila y se aplica el efecto correspondiente a la moto.

Alternativas Consideradas:

- **Manejo Automático de Poderes:** Se consideró hacer que los poderes se apliquen automáticamente en lugar de darle control al jugador, pero esto fue descartado para darle más estrategia al juego.

Limitaciones:

- **Interacción del Jugador:** Dependiendo de la cantidad de poderes, la interfaz puede volverse sobrecargada, lo que podría dificultar la selección adecuada del poder.

Problemas Encontrados:

- **Sincronización de la Interfaz:** Asegurar que la interfaz gráfica y la lógica de juego estén sincronizadas correctamente para reflejar el estado actual de la pila de poderes fue un desafío.

5. Requerimiento 05: Aplicación Automática de Ítems

Descripción del Problema:

Los ítems se aplican automáticamente en el orden de llegada con un delay de 1 segundo entre cada uno, priorizando las celdas de combustible. Si el combustible está lleno, la celda se reinserta en la cola sin aplicarse.

Implementación:

- **Manejo de Ítems en la Cola:** Los ítems se almacenan en una cola (Queue), y un temporizador se encarga de aplicar cada ítem en intervalos de 1 segundo. Las celdas de combustible tienen prioridad y se reinserta en la cola si no se pueden aplicar.
- **Reinserción de Ítems:** Se implementó lógica para verificar si el ítem puede ser aplicado. Si no puede ser aplicado, se vuelve a insertar en la cola para intentar aplicarlo más tarde.

Alternativas Consideradas:

- **Aplicación Manual de Ítems:** Se consideró permitir al jugador aplicar los ítems manualmente, pero se descartó para mantener el enfoque en la estrategia de los poderes.

Limitaciones:

- **Sobrecarga de Ítems:** Si un jugador acumula demasiados ítems sin usarlos, podría sobrecargar la cola y ralentizar el juego.

Problemas Encontrados:

- **Timing de la Aplicación:** Asegurar que los ítems se aplicaran de manera uniforme y sin sobrecargar la lógica del juego fue un reto inicial.

6. Requerimiento 06: Destrucción de la Moto

Descripción del Problema:

Una moto se destruye al:

- Chocar con otro jugador (ambos mueren).
- Cruzar una estela, ya sea la propia o la de otro jugador.
- Quedarse sin combustible.

Las motos nunca se detienen; el jugador únicamente puede cambiar la dirección en la que se mueven.

Implementación:

- **Detección de Colisiones:**
  - **Colisión con Otra Moto:** Se implementó una verificación de colisiones al final de cada movimiento de la moto. Si una moto se mueve a una posición ocupada por otra moto, ambas motos se destruyen. Esto se maneja en el método de movimiento de la moto y en la actualización del estado del grid.
  - **Colisión con la Estela:** Al mover la moto, se verifica si la nueva posición coincide con alguna de las posiciones en la lista enlazada de la estela de cualquier moto. Si es así, la moto se destruye.
  - **Combustible:** El combustible se reduce a medida que la moto se mueve, y si llega a 0, la moto se destruye automáticamente.
- **Destrucción de la Moto:**
  - Cuando una moto se destruye, se elimina su referencia de la lista de motos activas en el juego, y sus ítems y poderes no usados se dispersan en el mapa (como se mencionó en el Requerimiento 003).

Alternativas Consideradas:

- **Moto Fantasma:** Se consideró una alternativa donde la moto podría atravesar otras estelas temporalmente utilizando un poder especial, pero esta característica se dejó como opcional para no complicar la mecánica básica del juego.

Limitaciones:

- **Sin Detener el Juego:** Las motos nunca se detienen, lo que significa que el jugador siempre debe estar atento a cambiar de dirección, lo que puede aumentar la dificultad del juego.

Problemas Encontrados:

- **Sincronización del Combustible:** Hubo desafíos en sincronizar la reducción del combustible con el movimiento de la moto, asegurando que la destrucción de la moto ocurriera en el momento adecuado cuando el combustible llegara a 0.

7. Requerimiento 07: Implementación del Grid

Descripción del Problema:

El mapa del juego es un grid (malla) de tamaño fijo, implementado mediante una lista enlazada en la que cada nodo posee 4 referencias a otros nodos, formando así la red. Cuando el juego inicia, se carga el mapa con un tamaño previamente definido, y el jugador utiliza las flechas del teclado para mover la moto en el grid.

Implementación:

- **Estructura del Grid:**
  - **Lista Enlazada:** El grid se implementa como una lista enlazada donde cada celda tiene referencias a las celdas adyacentes (arriba, abajo, izquierda, derecha). Esta estructura permite un fácil acceso y manipulación de las celdas durante el juego.
  - **Carga del Mapa:** Al inicio del juego, se inicializa el grid con el tamaño especificado. Cada nodo se conecta con sus vecinos de acuerdo con su posición en la malla.
- **Movimiento en el Grid:**
  - **Teclas de Dirección:** El jugador puede mover la moto usando las flechas del teclado. Cada movimiento actualiza la posición de la moto en el grid, ajustando su referencia a los nodos correspondientes en la lista enlazada.

Alternativas Consideradas:

- **Matriz Bidimensional:** Se consideró usar una matriz bidimensional en lugar de una lista enlazada para representar el grid, pero la lista enlazada se seleccionó para permitir un manejo más flexible de las conexiones entre celdas.

Limitaciones:

- **Eficiencia:** El uso de una lista enlazada puede ser menos eficiente en términos de tiempo de acceso a celdas específicas en comparación con una matriz bidimensional.

Problemas Encontrados:

- **Conexiones Erróneas:** Durante la implementación inicial, hubo problemas al conectar correctamente las celdas adyacentes, lo que causaba movimientos incorrectos de las motos.

8. Requerimiento 08: Ítems y Poderes en el Mapa

Descripción del Problema:

Ítems y poderes aparecen aleatoriamente en el mapa, y los jugadores pueden recogerlos. Los ítems incluyen:

- **Celda de Combustible:** Incrementa el combustible de la moto. Cada celda tiene una capacidad aleatoria.
- **Crecimiento de Estela:** Incrementa el tamaño de la estela en un valor aleatorio de 1 a 10.
- **Bombas:** Cuando un jugador toma una bomba, explota.

Los poderes incluyen:

- **Escudo:** Permite que la moto se haga invencible por un tiempo variable.
- **Hiper Velocidad:** Aumenta la velocidad de la moto en un valor aleatorio y por un periodo aleatorio.

Implementación:

- **Generación Aleatoria de Ítems y Poderes:** Se implementó una función que genera ítems y poderes en posiciones aleatorias dentro del grid. Estas posiciones se eligen de manera que no coincidan con la estela o la posición actual de las motos.

- **Interacción con Ítems:**
    - **Celda de Combustible:** Al recoger una celda de combustible, se incrementa el combustible de la moto. La cantidad aumentada es aleatoria, y si el combustible está lleno, el ítem se reinserta en el mapa.
    - **Crecimiento de Estela:** Incrementa el tamaño de la estela de la moto de manera inmediata.
    - **Bombas:** Si una moto recoge una bomba, explota y se destruye.
- **Aplicación de Poderes:**
    - **Escudo:** Se activa un temporizador que hace a la moto invencible por un tiempo determinado. Visualmente, la moto cambia para reflejar su estado invencible.
    - **Hiper Velocidad:** Aumenta temporalmente la velocidad de la moto, también afectando su visualización en pantalla.

Alternativas Consideradas:

- **Ítems Fijos:** Se consideró hacer que algunos ítems aparezcan en posiciones fijas o predeterminadas para aumentar la predictibilidad del juego, pero se descartó en favor de la generación aleatoria para mantener la dinámica impredecible.

Limitaciones:

- **Generación Aleatoria:** La generación aleatoria puede provocar que algunos ítems aparezcan en posiciones inaccesibles o fuera del alcance inmediato de los jugadores.

Problemas Encontrados:

- **Colocación de Ítems:** Asegurar que los ítems no aparecieran en posiciones ocupadas por estelas u otras motos fue un reto, requiriendo múltiples validaciones.

9. Requerimiento 009: Bots que Simulan Jugadores

Descripción del Problema:

El juego debe incluir bots que simulen otros jugadores. Todas las reglas anteriores aplican para los bots. Su comportamiento es aleatorio y debe haber al menos 4 bots simultáneos en el juego.

Implementación:

- **Clase BotLightCycle:**
  - **Movimiento Aleatorio:** Cada bot se mueve de manera aleatoria en el grid. Se implementó un algoritmo que permite a los bots cambiar de dirección al azar cada cierto intervalo, o al encontrarse con el borde del grid.
  - **Cumplimiento de Reglas:** Los bots siguen las mismas reglas que las motos controladas por el jugador, incluyendo la generación de estelas, el consumo de combustible, y la posibilidad de recoger ítems y poderes.
  - **Velocidad y Direcciones:** Se asignó a cada bot una velocidad y dirección inicial al azar. Los bots pueden cambiar de dirección automáticamente o seguir su curso hasta que decidan cambiar.

Alternativas Consideradas:

- **Bots Predeterminados:** Se consideró predefinir las rutas y movimientos de los bots para hacerlos más predecibles, pero esto se descartó para mantener la impredecibilidad y desafío del juego.

Limitaciones:

- **Comportamiento Aleatorio:** El comportamiento completamente aleatorio de los bots puede hacer que a veces se muevan de manera ineficaz o caótica, afectando la fluidez del juego.

Problemas Encontrados:

- **Colisiones entre Bots:** Se enfrentaron problemas iniciales con colisiones frecuentes entre bots debido a sus movimientos aleatorios, lo que requería ajustes en la lógica de movimiento.

10. Requerimiento 010: Plataforma de Desarrollo

Descripción del Problema:

El juego se programará en C# con una interfaz gráfica en Windows Forms, MAUI, o Unity.

Implementación:

- **Elección de Plataforma:** Se seleccionó **Windows Forms** para la interfaz gráfica del juego debido a su simplicidad y rápida integración con el lenguaje C#.
- **Diseño de la Interfaz:** Se implementó una interfaz básica que incluye un área para el grid, indicadores de estado como combustible, y controles para manejar los poderes.

Alternativas Consideradas:

- **Unity:** Se consideró utilizar Unity para aprovechar sus capacidades gráficas avanzadas, pero se decidió que era innecesario para la complejidad del juego actual.

Limitaciones:

- **Gráficos Básicos:** Windows Forms tiene limitaciones gráficas en comparación con plataformas más avanzadas como Unity, lo que afecta la visualización del juego.

Problemas Encontrados:

- **Integración de Gráficos:** Hubo problemas al integrar gráficos personalizados dentro de Windows Forms, lo que requirió trabajo adicional para lograr la visualización adecuada del juego.

Diseño General: (Atributos y Métodos en la imagen)

Relaciones entre las clases:

1. **MotoDeLuz** tiene una relación de **composición** con **Estela**, que es una lista enlazada de nodos que representan las posiciones de la estela.

2. **MotoDeLuz** tiene una relación de **agregación** con **Item** (usando una cola para los ítems).

3. **MotoDeLuz** tiene una relación de **agregación** con **Poder** (usando una pila para los poderes).

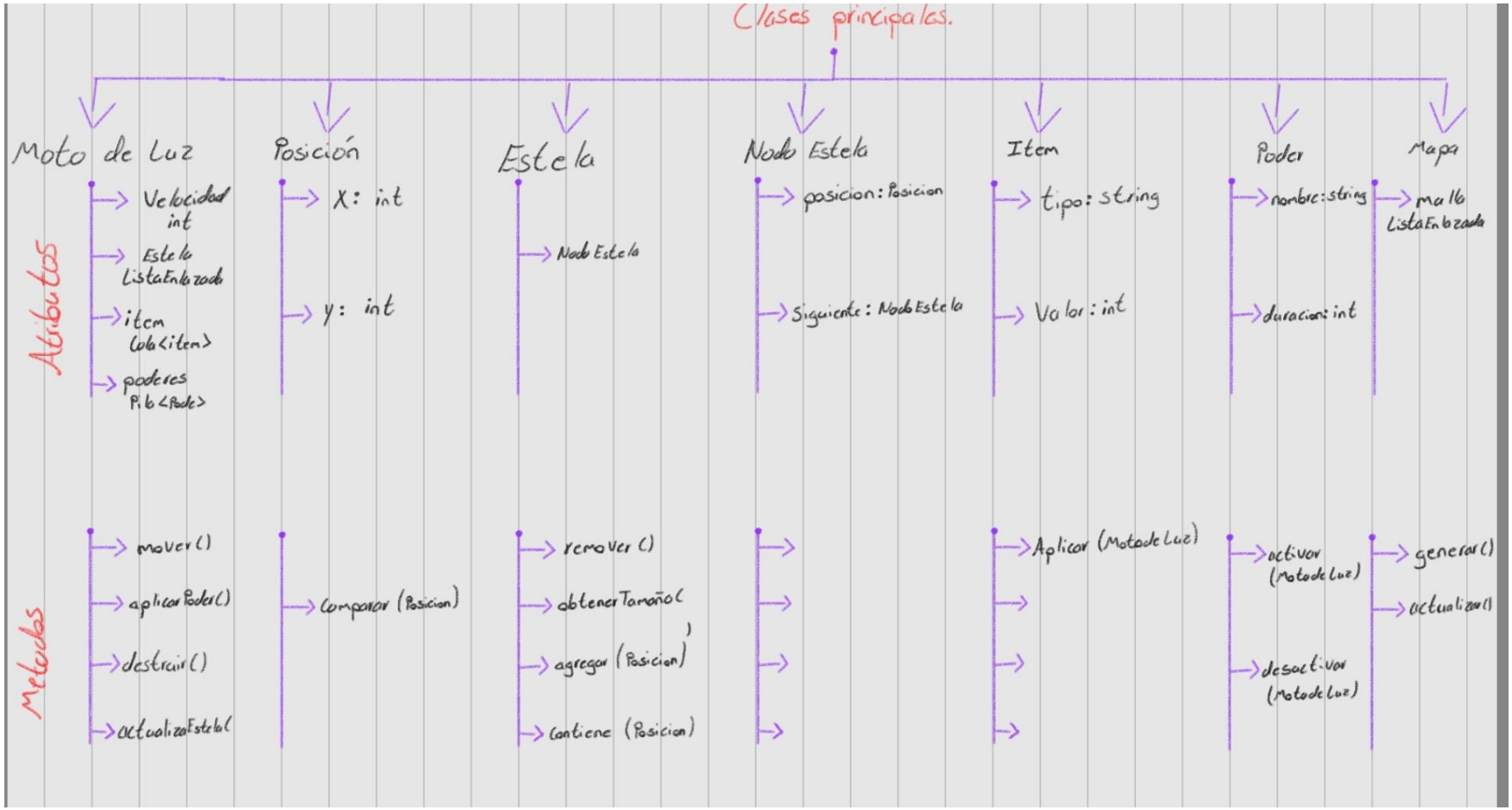
4. **Jugador** es una clase que **agrega** una instancia de **MotoDeLuz** para controlarla.

5. **Mapa** contiene una lista enlazada de nodos que representan las celdas del grid.

6. **Bot** hereda de **Jugador** y tiene comportamiento propio.

7. **ItemCombustible** y **ItemCrecimientoEstela** heredan de la clase base **Item**.

8. **PoderEscudo** y **PoderHiperVelocidad** heredan de la clase base **Poder**.


Clases principales.
Moto de Luz
Posición
Estela
Nodo Estela
Item
Poder
Mapa
Atributos
Velocidad int
Estela ListaEnlazada
item Lib<item>
poderes Pib<Poder>
x: int
y: int
NodoEstela
posicion: Posicion
Siguiente: NodoEstela
tipo: string
Valor: int
nombre: string
duracion: int
malla ListaEnlazada
Metodos
mover()
aplicarPoder()
destruir()
actualizarEstela()
Comparar (Posicion)
remover()
obtenerTamaño()
agregar (Posicion)
contiene (Posicion)
Aplicar (MotodeLuz)
activar (MotodeLuz)
desactivar (MotodeLuz)
generar()
actualizar()