**Universidad de La Habana**

**Facultad de Matemática y Computación (MATCOM)**

**Carrera: Ciencias de la Computación**

**Asignatura: Programación (Primer Año)**

**Informe del Proyecto de Programación**

**Autor:** Daniel Collazo Aldana  
**Grupo:** 122

**Informe del Proyecto: Lógica y Algoritmos en el Juego**

**Introducción**

Este informe describe el desarrollo de un juego como parte del primer proyecto de la asignatura de Programación en la carrera de Ciencias de la Computación. El juego está basado en un laberinto generado procedimentalmente mediante el algoritmo de **Prim**, con búsqueda de rutas optimizada mediante **A**\* para el movimiento eficiente de la inteligencia artificial (**IA**).

Además, el juego incorpora mecánicas estratégicas como trampas y habilidades especiales para los jugadores, lo que añade profundidad a la jugabilidad y permite la toma de decisiones tácticas.

**Instrucciones de Instalación**

1. **Requisitos previos:**
   * Tener instalado **.NET SDK 6.0** o superior.
   * Contar con un editor de código como **Visual Studio Code** o **Visual Studio 2022**.
2. **Clonar el repositorio:**
3. git clone <URL\_DEL\_REPOSITORIO>
4. cd <NOMBRE\_DEL\_PROYECTO>
5. **Compilar y ejecutar el juego:**
6. dotnet run

**Cómo Jugar**

El juego se desarrolla en una interfaz de consola. A continuación, se detallan las reglas y mecánicas principales:

1. **Modo de juego:** Se puede jugar en dos modalidades:
   * **Jugador vs. IA:** Un jugador humano contra la inteligencia artificial.
   * **Jugador vs. Jugador:** Dos jugadores humanos compiten entre sí.
2. **Objetivo:** Cada jugador debe atravesar el laberinto y llegar a la meta antes que su oponente.
3. **Turnos:** El juego es por turnos, alternando entre los jugadores.
4. **Controles:** Los jugadores pueden moverse utilizando las teclas:
   * **W** → Arriba
   * **S** → Abajo
   * **A** → Izquierda
   * **D** → Derecha
5. **Acciones disponibles:** En su turno, el jugador puede:
   * Moverse en una de las cuatro direcciones posibles.
   * Usar habilidades especiales (si tiene intentos disponibles).
   * Colocar trampas estratégicamente para obstaculizar al oponente.

**Detalles Técnicos e Implementación**

**Generación del Laberinto con el Algoritmo de Prim**

Se utiliza **Prim** para la generación procedural del laberinto. Su lógica se basa en:

1. Inicializar la cuadrícula con paredes.
2. Seleccionar una celda de inicio y marcarla como parte del camino.
3. Agregar sus paredes a una lista.
4. Seleccionar una pared al azar y verificar si su celda vecina solo tiene una conexión.
5. Convertir la pared en un camino y agregar nuevas paredes a la lista.
6. Repetir hasta que no queden paredes por procesar.

Esto garantiza que siempre exista un camino válido desde el inicio hasta la meta.

**Búsqueda de Ruta con el Algoritmo A\***

El movimiento de la IA está optimizado con el algoritmo **A**\*, que encuentra la ruta más corta desde su posición hasta la meta.

Pasos del algoritmo:

1. Inicializar una lista de nodos abiertos (por explorar) y otra de nodos cerrados (explorados).
2. Agregar la posición inicial de la IA a la lista de nodos abiertos.
3. Mientras la lista de nodos abiertos no esté vacía:
   * Seleccionar el nodo con menor costo estimado (g + h).
   * Moverlo a la lista de nodos cerrados.
   * Expandir sus vecinos y actualizar costos si se encuentra una mejor ruta.
4. Reconstruir la ruta desde la meta hasta el inicio.

Este método permite que la IA se mueva de manera óptima dentro del laberinto.

**Sistema de Clases y Mecánicas Implementadas**

El juego sigue un modelo basado en clases para organizar la lógica del sistema:

**Clase GameManager**

* Controla la lógica del juego, incluyendo la generación del laberinto, el flujo de turnos y la interacción entre los jugadores.

**Clase Player**

* Representa a los jugadores y gestiona el uso de habilidades especiales.
* Habilidades disponibles:
  + **Teletransporte:** Se mueve a una posición aleatoria dentro del laberinto.
  + **Teletransporte hacia la salida:** Se mueve un paso en dirección a la meta usando A\*.
  + **Intercambio de posiciones:** Cambia de lugar con el otro jugador.
  + **Aturdir al oponente:** Hace que el otro jugador pierda un turno.
  + **Colocar trampa:** Agrega una trampa en una celda aleatoria del laberinto.

**Clase Trap**

* Representa una trampa colocada en el laberinto.
* Tipos de trampas:
  + **Trampa normal:** Teletransporta al jugador a otra posición aleatoria.
  + **Trampa de intercambio:** Intercambia la posición del jugador con su oponente.
  + **Trampa de retroceso:** Mueve al jugador tres pasos hacia atrás.

**Sistema de Turnos**

* Los turnos alternan entre los jugadores.
* Cada turno permite al jugador moverse o usar una habilidad especial.
* La IA utiliza **A**\* para calcular la mejor ruta hacia la meta.

**Conclusión**

El desarrollo de este juego permitió aplicar algoritmos fundamentales como **A**\* y **Prim**, reforzando conceptos clave de estructuras de datos y programación orientada a objetos.

Se logró implementar una mecánica de juego estratégica, en la que los jugadores deben planificar cuidadosamente sus movimientos y el uso de habilidades.

Este proyecto representa un primer acercamiento al desarrollo de videojuegos dentro de la carrera de Ciencias de la Computación y sienta las bases para futuros trabajos más complejos.