EDA

Trabajo Práctico Especial

19 de Junio de 2018

Micaela Banfi 57293

Marcos Lund 57159

Clara Guzzetti 57100

Introducción

En este documento se expondrá el diseño de un juego, Timbiriche, donde se puso en práctica el algoritmo recursivo minimax para el desarrollo de una partida entre inteligencias artificiales. También se explicarán la estructura general del programa y otras decisiones ligadas al back-end así como también del front-end.

Estructura general

Para la arquitectura del programa se implementó el patrón MVC (Modelo - Vista - Controlador) con el fin de separar adecuadamente la lógica de la representación del juego. El modelo incluye todas las clases relacionadas al funcionamiento interno del proyecto; la vista agrupa aquellas clases que abarcan la interacción con el usuario e interfaces gráficas; y el controlador responde a eventos de ambos grupos y los comunica entre sí. De esta manera se logró facilitar el desarrollo del proyecto y su mantenimiento.

Cabe mencionar que la generación del .DOT se garantiza que funcione sólo para el modo depth con parámetro 2 o 1, ya que con mayores valores el árbol era muy grande para evaluar. Para acceder al mismo, al presionar el botón DOT, se creara esta archivo en la carpeta del repositorio.Para su correcta visualización, se lo deberá abrir con un visor de .dot.

Implementación del algoritmo *minimax*

- Detalle de la heurística

Con respecto a la heurística se decidió que la misma fuese lo más simple posible para que el la IA del algoritmo se sustente en la profundidad o el tiempo. Por ende, la misma se definió dentro de un nodo estado como *misPuntos - puntosOponente.*

- Decisiones de diseño para la implementación del algoritmo minimax.

El algoritmo minimax se construyó sobre la estrategia IDS (iterative deepening search) el cual combina BFS con DFS ejecutándose repetidamente con mayor profundidades límites hasta que se encuentre el objetivo, o en este caso, se corte por parámetro. IDS es equivalente a BFS, pero usa mucha menos memoria.
La principal ventaja es la capacidad de respuesta del algoritmo. Debido a que las iteraciones tempranas usan valores pequeños para la profundidad, se ejecutan extremadamente rápido. Esto permite que el algoritmo proporcione indicaciones tempranas del resultado casi de inmediato, seguido de refinamientos a medida que la condición de corte aumenta.

- Problemas encontrados durante la implementación del algoritmo.

Se encontraron problemas a la hora de la implementación relacionados con el corte por tiempo, ya que se tuvo que asignar una fracción del mismo a cada rama del primer nivel para que cada una efectúe el IDS dentro de ese marco de tiempo.

También resultó un obstáculo la funcionalidad recursiva de la exploración del árbol entre los MAX y MIN y la creación de nuevos estados para cada movimiento, aunque pudo ser superado.

Decisiones sobre back-end

El diseño del tablero fue una de las primeras decisiones que se tomó respecto al proyecto por su gran incidencia en todo lo demás. Una posible respuesta que surgió fue la de un grafo, pero las partes gráfica y de chequeo de puntos se complicaba de cierta manera. Finalmente se decidió una estructura de datos que consistía en dos matrices: una que contuviera todas las aristas verticales del tablero y otra que contuviera las aristas horizontales del mismo. También se incluyó una matriz cuyos elementos fueran los cuadrados que forman las aristas del tablero.

En cuanto a los jugadores se diseñaron clases para jugadores humanos y para inteligencias artificiales, ambas implementando una misma interfaz de jugador y con sus métodos de jugar respectivos. Aquí es donde se encuentra implementado el algoritmo minimax.

Finalmente, para cada jugada se diseñó una clase que consistía en el jugador que la llevó a cabo y la artista del tablero que rellenó. A medida que transcurre el juego se utiliza un stack que pushea las diferentes jugadas realizadas y que las popea cuando se selecciona la opción undo.

Decisiones sobre front-end

Para el desarrollo del TPE se decidió utilizar la librería Swing de Java para los aspectos relacionados al front-end: mostrar el tablero, puntajes y botones. Consideramos que el juego no era lo suficientemente complejo visualmente como para usar LibGDX o alguna otra alternativa, teniendo en cuenta las limitaciones que Swing posee.

La ventana principal (clase GameWindow) incluye dos paneles dentro de sí misma. Por un lado, la clase BoardPanel contiene el tablero y por el otro, la clase ScorePanel contiene JLabels indicando a quién le toca jugar y los respectivos puntajes, así como también botones de Undo (deshacer la última jugada) y Dot (generar el archivo dot).

Para el tablero se generaron nodos internos en un formato de tipo GridLayout y se agregaron Listeners a cada uno. BoardPanel luego “pinta” líneas por cada conexión válida entre nodos que se realiza. También se encarga de enviar al Model (mediante el Controller) el eje seleccionado y de recibir del mismo el eje que la Inteligencia Artificial ha elegido para mostrarlo visualmente.

Conclusión

Resultó muy interesante poder observar cómo el algoritmo se llevaba a cabo adecuadamente para poder implementar el mismo en futuras aplicaciones de inteligencia artificial. Hasta el momento se venía trabajando en TPEs con sistemas de decisión muy absurdos, por lo cual este trabajo cumplió nuestras expectativas.

Como futuras extensiones del trabajo figuran optimizar el algoritmo minimax, la implementación de juegos de a más de 2 jugadores, la opción para guardar el estado de una partida y el mejoramiento de la parte visual como puede ser un menú principal y mostrar los cuadrados pintados según quién obtuvo el punto. Otra posible implementación para una extensión sería que el juego termine antes de completar todas las aristas posibles si los puntos indican que uno de los jugadores va a ganar sin importar los siguientes movimientos.