Jugador inteligente

BattleTech

ICO

Carlos Jesús Fernández Basso 75927137-C
Juan Manuel Rodríguez Trillo 76439757-Q

ÍNDICE

1

Introducción

BattleTech *es un juego de combates entre enormes máquinas de aspecto humanoide llamados BattleMechs (o más brevemente llamados Mechs).*

BattleTech es un juego de tablero ambientado en un mundo ciencia ficción en la que grandes robots (denominados BattleMechs o Mechs) controlados por expertos pilotos humanos (denominados MechWarrior), se baten en cruentas batallas. El juego está por tanto dentro de la categoría de Juegos de Guerra que se caracterizan por tener una estructura por turnos con un desarrollo largo y con turnos en cuya resolución influye una componente de estrategia por parte del jugador y el azar de los dados. Fue inventando por la extinta FASA Corporation, actualmente propiedad de WizKids, en 1984.
El juego tiene numerosas versiones pero nosotros nos regiremos al Battletech Clásico.

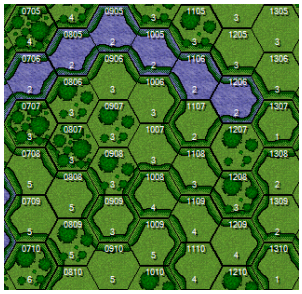
1.1 Descripción del Problema

En la práctica se ha de diseñar y desarrollar un prototipo de Jugador Inteligente. Nuestro objetivo es crear un jugador inteligente que sea capaz de dirigir a los BattleMechs. Para ello debe decidir la acción a realizar en cada momento por ellos. Nuestro problema se restringe únicamente a controlar a los Mechs, aunque existan muchos otros tipos de vehículos en los combates - BattleMech, vehículo, Pelotón de infantería o Punta o Escuadra de armaduras de combate -.

1.1.1 BattleMechs

Los BattleMechs –las más poderosas máquinas de guerra jamás construidas- dominan los campos de batalla del siglo XXXI. Estos inmensos vehículos de forma humanoide alcanzan doce metros de altura y llegan a pesar 100 toneladas. Son más rápidos, más maniobrables, están mejor blindados y más pesadamente armados que cualquier otra unidad de combate existente. Equipados con cañones proyectores de partículas, láseres, cañones automáticos de tiro rápido, misiles, ametralladoras y lanzallamas, estas bestias colosales disponen de suficiente potencia de fuego para aplastar cualquier cosa excepto a otro Mech.

1.1.2 Mapa



En Battletech las partidas se juegan en mapas divididos en áreas de seis lados llamadas hexágonos, las cuales regulan el movimiento y el combate entre varias unidades. Cada una de estas casillas tiene una serie de características:

- Elevación, define la altura del terreno del terreno.
- Objetos en casilla, como pueden ser escombros, edificios, bosques, etc.
- Tipo de terreno, que describe el material del suelo del mapa.
- Etc...

1.1.3 Secuencia de Juego

Una partida de Battletech consiste en una serie de turnos. Durante cada turno todas las unidades en el mapa tienen la oportunidad de moverse y disparar sus armas. Cada turno consiste en varios segmentos de tiempo más pequeños, llamados fases.
Durante cada fase, los jugadores pueden escoger un tipo de acción, cada turno incluye las correspondientes fases, que se desarrollan en el orden siguiente:

1. Fase de Movimiento
2. Fase de Reacción
3. Fase de Ataque con Armas
4. Fase de Ataque Físicos
5. Fase de Final de Turno

Fase de Movimiento

El jugador que ha perdido la iniciativa mueve primero. El jugador que ganó la iniciativa mueve entonces. El movimiento se alterna entre los bandos hasta que todas las unidades se hayan movido.

Las unidades de Battletech cambian de posición y ubicación en el mapa realizando uno de los distintos movimientos. Durante la Fase de Movimiento de cada turno los jugadores deben escoger un modo de movimiento, para los Mechs las opciones son andar, correr o saltar.

Fase de Reacción

En ella los jugadores podrán cambiar el encaramiento de su torso (no de las piernas) hacia izquierda o derecha, acción que luego afectara al resto de fases.

Fase de Ataque con Armas

Fase en la que los jugadores, por orden de iniciativa, pueden abrir fuego con la artillería de sus mechs. El Mech elegir ‘a las armas que disparara, dentro de las que dispone en su arsenal, y sobre qué enemigo usarlas.

Fase de Ataque Físicos
Fase en la que se presenta una nueva opción la opción de realizar daño físico a otros Mech, y es la última opción de hacerles daño dentro del turno en el que nos encontramos. El ataque físico consistirá en realizar un impacto físico, con algún garrote/arma o extremidad del robot.

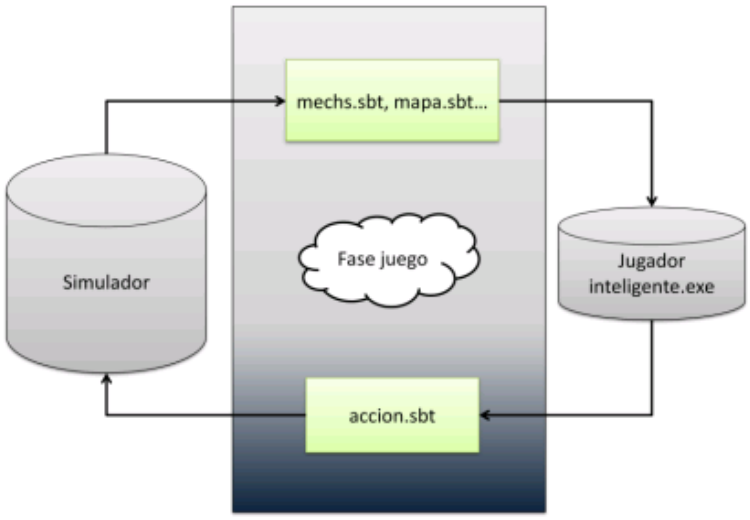
Fase de Final de Turno
Fase en la que los jugadores pueden determinar acciones especiales como tirar algún garrote o munición al suelo o apagar o encender radiadores.

1.1.4 Objetivo

El objetivo del juego es sencillo: acabar con el resto de battlemechs de la partida de forma que al final nuestro mechs sea el ultimo el pie.

1.2 Abstracción del Problema

Para resolver nuestro problema podemos ver al jugador inteligente como un *ente*, que percibe el universo a través de un conjunto entradas - que nos indican el estado actual del juego -. Este ente debe ser capaz de escoger una acción a realizar.



Pero, ¿Cómo escoger cual de entre todas las posibles acciones a realizar? La respuesta es bastante subjetiva, ya dependerá de la estrategia a seguir. Según la escogida, se obtendrá un resultado mejor o peor, pero también depende del caso concreto que estemos estudiando, ya que si cambiamos ciertos parámetros de nuestro universo, otra estrategia puede resultar más favorable.

A la hora de decidir sobre qué estrategia seguir, se debe de tener en cuenta que tanto la experiencia adquirida tras años de juego y el conocimiento de las reglas del mismo, son puntos críticos a la hora de maximizar nuestro rendimiento. Nuestro ente se mueve en un espacio temporal discreto, el juego se divide por turnos y estos a su vez, se ve dividido en segmentos más pequeños a los que llamaremos fases. Cada fase se utiliza para realizar una acción determinada, y según la acción que queramos realizar seguiremos una manera de razonar distinta según los objetivos que queramos conseguir. Veamos entonces cada una de las fases.

1.2.1 Fase de Movimiento

1.2.2 Fase de Reacción

1.2.3 Fase de Ataque con Armas

1.2.4 Fase de Ataque Físico

1.2.5 Fase de Final de turno

2

Modelo Teórico

"Los agentes constituyen el próximo avance más significativo en el desarrollo de sistemas y pueden ser considerados como la nueva revolución en el software." Dr. Nicholas Jennings

Agente Inteligente

Un agente inteligente, es una entidad capaz de percibir su entorno, procesar tales percepciones y responder o actuar en su entorno de manera racional, es decir, de manera correcta y tendiendo a maximizar un resultado esperado.
En este contexto la racionalidad es la característica que posee una elección de ser correcta, más especıficamente, de tender a maximizar un resultado esperado.
Este concepto de racionalidad es más general y por ello más adecuado que inteligencia (la cual sugiere entendimiento) para describir el comportamiento de los agentes inteligentes. Por este motivo es mayor el consenso en llamarlos agentes racionales.

Nuestro jugador inteligente se corresponde con un agente racional, y por lo tanto debe poseer las características:

Reactivo. El jugador podrá responder a cambios en el entorno en el que se encuentra situado. Estos cambios se le indican al jugador mediante los ficheros, que en cada fase de la partida el jugador debe leer para obtener toda la información necesaria.

Pro-activo. El jugador debe ser capaz de intentar cumplir sus planes u objetivos. Su objetivo final será destruir al resto de los jugadores.

Autonomía. Nuestro agente actuará sin intervención de ningún usuario externo a nuestro programa. Por tanto tiene control total sobre sus acciones y estados internos.

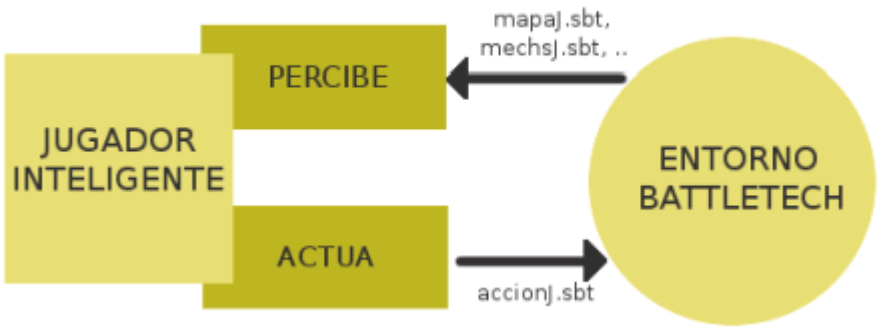


Figura 2.1: Esquema de funcionamiento.

Hemos de diferenciar nuestro sistema de los sistemas Multi-Agente (SMA).Estos son grupos de agentes que interaccionan entre sí para conseguir objetivos comunes, pero en nuestro entorno los distintos jugadores no se comunican entre sí - no tienen conducta social-. Si tuviéramos esto en cuenta, aumentaría la complejidad en el desarrollo. Nuestro agente es basado en metas. Estas ayudan a decidir las acciones correctas en cada momento. En nuestro juego, el agente o jugador escoge un objetivo, en base al cual toma sus decisiones. Por tanto, no basta con conocer el entorno, sino además es necesario determinar las acciones a seguir que permitan alcanzar la meta. Elegir las acciones correctas varia en complejidad.

Búsqueda

Planificación

Existen diferentes tipos de ambientes:

Accesibles y no accesibles. En nuestro caso se trata de un ambiente accesible, ya que el agente tiene acceso al estado total del ambiente.

Deterministas y no deterministas. Depende de si el estado siguiente se determina a partir del estado y las acciones elegidas por el agente.

Episódicos y no episódicos. En ambientes episódicos, como es nuestro caso, la experiencia del agente se divide en "episodios"- o fases-. Cada episodio consta de un agente que percibe y actúa.

Estáticos y dinámicos. Si el ambiente cambia mientras un agente toma una acción a seguir, entonces se dice que el ambiente es "dinámico". Pero en nuestro caso tratamos con ambientes estáticos, puesto que no se tiene que observar y pensar al mismo tiempo.

Discretos y continuos. Si existe una cantidad limitada de percepciones y acciones distintas y discernibles, se dice que el ambiente es discreto. Si no es posible enumerarlos, entonces es un ambiente continuo. Nuestro problema abarca ambientes discretos, lo cual nos facilita el trabajo.

Programa de Ambientes.

Un simulador toma como entrada uno o más agentes y dispone de lo necesario para proporcionar las percepciones correctas una y otra vez a cada agente y así recibir como respuesta una acción.
El simulador procede a actualizar al ambiente tomando como base las acciones, y posiblemente otros procesos dinámicos del ambiente que no se consideran como agentes.

3

Descripción de la Solución

3.1 Módulo de Movimiento

Uno de los mayores desafíos en el diseño de Inteligencia Artificial realista en juegos de ordenador es el movimiento del agente. Las estrategias de búsqueda de caminos o pathfinding son empleados como centro de los sistemas de movimiento.

Las estrategias de pathfinding deben encontrar un camino desde cualquier coordenada del mundo hasta otra. Dados los puntos origen y destino, encuentran intermedios que formen un camino a nuestro destino. Para esto debemos tomar algún tipo de estructura de datos para guiarnos en el movimiento. Esto nos lleva inevitablemente a utilizar recursos de CPU, especialmente cuando buscamos un camino que no existe.

De entre todos los algoritmos que se usan actualmente, el más conocido y extensamente usado es el algoritmo A estrella A*.

Veamos una comparación de tres tipos distintos de algoritmos.

Comparación de algoritmos de búsqueda de caminos.

1. **Djkstra**. Este algoritmo empieza visitando los vértices del grafo en el punto de partida. Luego va reiteradamente examinando los vértices más cercanos que aún no hayan sido examinados. Se expando desde el nodo inicial hasta alcanzar el destino. Pero aunque esté garantizado encontrar una solución óptima, comprueba demasiadas casillas, por lo que hace un gasto de recursos enorme.
Para una implementación simple, tenemos un tiempo de ejecución O(n 2)

2. **Best First Search.** Es un algoritmo Greedy que trabaja de una forma similar al algoritmo de Dijkstra, aunque este presenta una "heurística" de cómo de lejos está nuestro objetivo. Aunque no nos garantice encontrar la solución óptima, si puede encontrar una solución aproximada en un tiempo mucho menor. El mayor inconveniente con este algoritmo es que intenta moverse hacia el objetivo aunque no sea el camino correcto -tal y como muestra la figura 3.1.1. Esto es debido a que sólo tiene en cuenta el coste para llegar al objetivo, e ignora el coste del camino que lleva hasta ese momento. Entonces intentara seguir aunque el camino sea muy largo.
El tiempo de ejecución es O(n).

3. **A*.** Este algoritmo fue desarrollado en 1968 para combinar enfoques heurísticos como en Best First Searh y enfoques formales como ocurre en Dijkstra. Aunque A* este enfocado construido sobre la heurística -y aunque esta no proporciona ninguna garantía-, A* puede garantizar el camino más corto.

Nota: podremos ver una prueba de ejecución entre A* y Dijkstra en la presentación

El Algoritmo A*

El algoritmo de búsqueda A* es un tipo de algoritmo de búsqueda en grafos. Se basa en encontrar, siempre y cuando se cumplan ciertas condiciones, el camino de menor coste entre un nodo origen y uno objetivo. Nuestro objetivo es encontrar el camino más corto entre dos puntos superando obstáculos (ya que en caso de que no hubiera obstáculos, es la línea recta).

Esta técnica muy usada en videojuegos de estrategia y, en general en todos los videojuegos donde se trata la inteligencia artificial. Por ello decidimos incorporarla a nuestra práctica.

La mayor ventaja de este algoritmo con respecto a otros es que tiene en cuenta tanto el valor heurístico de los nodos como el coste real del recorrido.

3.2 Módulo de Reacción

El módulo de reacción es inmediatamente posterior al módulo de movimiento. En esta fase se presenta la posibilidad de corregir el encaramiento del torso para mejorar la estrategia de ataque - mejorando el ángulo de disparo, tener a mas enemigos en línea de visión, proteger partes débiles de nuestro mech, etc. -.
Las posibilidades que se nos presentan ahora son tres:

change = {0: ' I g u a l ' , 1 : ' Derecha ' , 2: ' I z q u i e r d a '}

Las cuales nos indican los tres posibles cambios que podemos realizar:

1. Igual. Indica que no vamos a realizar ningún cambio. Dejamos nuestro encaramiento intacto para las siguientes fases.
2. Derecha. Giramos el mech para posicionar su torso una cara a Derecha.
3. Izquierda. Giramos el mech para posicionar su torso una cara a Izquierda.

3.3 Módulo de Ataque con Armas

3.3.1 Lógica del ataque con armas

El ataque con armas es una de las fases más importantes del juego. El objetivo principal del Mech en la fase de ataque con armas será elegir a que enemigos y con que armas desea atacar. Para poder realizar esta fase inteligentemente, dotaremos al Mech de la capacidad para analizar los distintos condicionantes que intervienen en el ataque. Antes de adentrarnos en que tipos de condicionantes, es necesario definir la estrategia general que realizarán nuestros Mechs. La estrategia será maximizar el daño sobre el atacado minimizando los daños en el atacante, es decir, atacaremos al Mech objetivo con todas las armas posibles mientras no se

supere un valor umbral de daño en forma de calor en el atacante. Por tanto queda claro que los condicionantes que debe determinar el Mech serán:

1. Se realiza ataque.
   a) Localización del Mech objetivo.
   b) Elección de armas a usar en el ataque.
   c) Determinación del valor umbral de temperatura.
2. No se realiza ataque.

3.3.2 Se realiza un ataque

En primer lugar es necesario saber si el Mech desea atacar o prefiero no realizar ningún tipo de ataque. Para discriminar entre las dos actuaciones nos basamos en calcular que enemigos están dentro de nuestra línea de visión. Si existe al menos un Mech dentro de nuestra línea de visión el ataque se realizará.
Localización del Mech objetivo Una vez seleccionados los Mechs que están dentro de nuestra línea de visión, elegimos cual es el que está más próximo a nosotros, calculando la distancia entre nuestra casilla en el tablero y las casillas de los enemigos dentro de nuestra línea de visión. Determinamos así el Mech objetivo sobre el que recaerá todo el poder de nuestras armas.
Elección de armas a usar en el ataque

Una vez determinado el objetivo del ataque procedemos a elegir las armas que podemos usar en el ataque. Cada arma tiene un valor que corresponde a la distancia que es capaz de alcanzar en un disparo, por tanto, el siguiente paso será comprobar de entre todas las armas de nuestro Mech cuales son capaces de alcanzar la distancia a la que está de nosotros el Mech objetivo.
Determinación del valor umbral de temperatura

Debe quedar claro que lanzar todas las armas posibles a nuestro enemigo no es una buena estrategia, ya que a cada disparo, la temperatura de nuestro Mech puede ascender de tal forma que incluso quede desactivado. Por tanto debemos calcular cuanta temperatura puede soportar el Mech en la realización del ataque.
Sabemos que si un Mech pasa de 15 o al acabar el turno, recibirá un punto de daño y si sobrepasa 26 o recibirá dos. Por tanto podemos determinar el umbral de temperatura soportable de la siguiente forma:

- Si la temperatura del Mech es menor que 10º o el umbral será 12º.
- Si la temperatura del Mech está comprendida entre 10º o y 15º o el umbral será 9º.
- Si la temperatura del Mech está comprendida entre 16º y 20º el umbral será 6º.
- Si la temperatura del Mech está comprendida entre 20º y 26º el umbral
- Será 4º.

- Si la temperatura del Mech es mayor que 26º el umbral será 2º

3.3.3 No se realiza ataque

Si no hay ningún enemigo dentro de nuestra línea de visión no realizaremos Ningún ataque, quedando como ´única posibilidad recoger un garrote, si está alojado en nuestra casilla, para usarlo más tarde en un posible ataque físico.

3.4 Ataque Físico

3.5 Final de Turno

Esta fase nos brinda la posibilidad de apagar o encender radiadores, soltar garrote o expulsar municiones.

Bibliografía

1. **González Duque, R., 2003. "PYTHON para todos." 1 st ed.**
2. **Gutschmidt, Tom, 2003. "Game Programming with Python, Lua, and Ruby." Premier Press**
3. **Weixiong Zhang, 1999. "State-Space Search. Algorithms, Complexity, Extensions and Applications." Springer-Verlag: NY**
4. **Pilgrim, Mark, 2004. "Dive into Python." Apress.**