José Gallardo Harillo

Memoria del Proyecto 47

Índice:

Aspiraciones e idea original del proyecto:

Este proyecto ha tenido la aspiración principal de poder ser una versión minimalista del juego “Hitman”, un juego en cual debes infiltrarte en zonas hostiles llenas de guardias/patrullas sin que te vean donde hay que eliminar a uno o más objetivos, para luego poder salir del recinto y poder cumplir el contrato. Entonces con esta idea planteé los siguientes tipos de agentes:

- 47: Es el jugador/agente que controlaremos para cumplir el contrato. (color negro)
- Patrullas: Son los agentes que estarán patrullando por el mapa y los cuales querrán matarte al verte. (colores variados)
- Objetivo: Es el agente/s que tendremos que matar para poder salir del mapa. (color rojo)

Los patches que se plantearón para entonces fueron estos:

- Vacío/Suelo: Patches por los que los agentes se podrán desplazar (gris claro)
- Pared: Patches que no podrán ser penetrados (gris oscuro)
- Llave: Patch que simulará el ítem llave, la cual puede haber más de una y podrá abrir una puerta. (amarillo)
- Puerta: Patch penetrable siempre y cuando tengamos una llave
- Puerta del despacho: Puerta especial que podrá ser abierta si se obtiene la llave del despacho. (rosa)
- Llave del despacho: Llave especial con la que podremos entrar al despacho del objetivo. (magenta)
- Salida: Patch por el que podremos salir y terminar la fase cuando el objetivo/s sea eliminado. (cyan)
- Estatua: Patch que simula un ítem opcional, el cual nos hará ganar puntos al final de la partida si ganamos. (lima)

Este último tipo de patch se me ocurrió más tarde para darle un poco más de dificultad al juego, basándome en una fase de Hitman codename 47, donde nos encontramos en china y debemos recuperar una estatua de jade que está en posesión del líder de la triada roja Lee Hong (Además de tener que matarlo tras ello claro). Por tanto, podremos acabar la fase cumpliendo el objetivo principal, pero si queremos aspirar a una bonificación pues tendremos que obtener ese ítem.

Los requisitos y funcionalidades a programar eran las siguientes:

- Rapidez a la hora de cumplir el contrato, osea, empezar, coger la llave del despacho, matar al objetivo, y salir.
- Sin llave no se puede abrir puertas.
- Sin llave del despacho no se puede entrar en el despacho
- Si el agente 47 tiene el mismo color que un patrulla no será detectado.
- Si un patrulla está de espaldas puede ser asesinado y con ello quedarte un traje sí o no dependiendo de su necesidad.

La primera idea a la que enfoqué este proyecto fue de forma que no fuésemos nosotros los que jugásemos, sino que la máquina lo hiciese tomando el camino ideal teniendo en cuenta a los patrullas, y además que cumpliese las funciones y requisitos anteriormente dichos. Más tarde, viendo lo complicado que podía ser plantear un A* dinámico y a fin de cuentas lo "aburrido" que sería un jugador automático sin más, pues me planteé hacerlo para un jugador humano, y enfocar la inteligencia artificial en dos de las partes que considero más interesantes de tratar en un videojuego, los NPC (Non Playable Character), y la puntuación/score, los cuales quise tratar con las técnicas de Montecarlo y A* respectivamente (Machine Learning lo dejé para más adelante).

También pensé a la hora de generar el mapa en generar mapas dinámicos donde los patrullas, ítems, despacho, y punto de inicio se generasen directamente, e incluso poder mediante un deslizador generar más de un despacho/objetivos. Esta idea no tardó mucho en ser descartada cuando empecé a programar el juego, puesto que generar mapas con sentido donde no hubiese caminos sin sentido o incluso patches sueltos ya sea de pared o de suelo iba a ser súper difícil, sumado a ello el tiempo limitado que disponía, y a fin de cuenta no siendo lo importante del proyecto. Con lo anterior claramente la generación automática de objetivos, despachos y demás fueron también descartadas, y entonces decidí a inclinarme más por hacer mapas estáticos, diseñados a mano por mí mediante una función para colorear patches.

Tenía pensado entonces disponer de varios niveles diseñados a mano para probar los distintos algoritmos, planteado esto ya tenía la base y la idea al completo de mi proyecto, el proyecto 47.

Score:

Lo primero que empecé a programar fue el score, de forma que el PSR del problema lo pude sacar en poco tiempo, siendo el siguiente:

- X = {corx, cory} $\in$ p (Siendo p un patch o un agente del mapa)
- D = dir
  Siendo dir la dirección a la que puede tirar en ese momento el algoritmo, pudiendo ser 90 (derecha), 0 (arriba), 270 (izquierda), 180 (abajo).
- C = Si la coordenada seleccionada no existe o es una pared no es válida.

Un problema que me surgió para entonces fue que el orden en el que podías tomar los distintos ítems o podías ir a lugares de interés, pues eran relevantes para un camino más o menos largo. Estuve pensando bastantes formas para tratar ese problema, pensando primero en un conjunto de caminos mínimos y compararlos para ordenarlos, o incluso mediante un IC3 que tuviese como atributos los momentos donde podías decidir si tomar uno u otro. Pero claro, caí en que esto es el famoso problema del viajante realmente, donde los vértices son los patches destacados y las aristas las distancias, por lo que antes de meterme en algo que no había planeado como podía ser el algoritmo de hormigas pues preferí avanzar antes por el Montecarlo de los NPC y tras ello volver al problema del score extendido.

NPCs:

Antes de nada, había que saber las distintas partes que tenía en netlogo, siendo una de ellas el contenido, osea, un “tablero” que pueda ir modificando la máquina, por lo el proyecto dispone de un método que te saca una lista con los “valores” (atributo que puse a los patches) para luego pasar la lista a una lista de listas, osea, una matriz.

Las reglas y las aplicaciones tuvieron una serie de etapas de modificación por problemas luego a la hora de arrancar el algoritmo. Empecé probando el algoritmo con un solo patrulla, de forma que las reglas serían las

direcciones que podía tomar en ese momento (al igual que en mi turno cuando simula mis movimientos), y entonces claro, el patrulla se tiraba 3 minutos perfectamente para moverse, y para ejecutar un movimiento que incluso no tenía sentido, por lo que empecé a ver el problema de MonteCarlo. Le di una oportunidad al algoritmo añadiendo 3 patrullas más al mapa para limitar las posibilidades, e incluso cambiar el objetivo de los patrullas y en vez de querer que te maten pues que te hagan un rey ahogado (osea, llevar mi situación a el momento del ajedrez donde tienes solo al rey y el otro jugador debe hacerte jaque mate, solo que el ahogado aquí es el jaque mate). Ahora me encontré con otro problema, y es que sacar todas las combinaciones de posibles fichas y movimientos para mover cada patrulla a otro patch era considerablemente enorme, solo imagina escoger 1 movimiento para cada patrulla en un turno y disponer como máximo de 4 alternativas para cada uno. Por tanto, pensé en dos alternativas:

1. Hacer que la máquina decida sólo la ficha que quiere mover y a donde.
2. Simular una partida con cuatro jugadores con un patrulla a disposición de cada uno con solo 4 alternativas como máximo cada uno.

La segunda opción pensé que podía llevar a descoordinación a la hora de que los patrullas planteasen una forma de pillarme, de forma que uno por ejemplo tuviese una estrategia y luego otro tuviese otra pero sin tener en cuenta las ideas de los otros, por tanto opté al final mejor por la primera opción.

Terminado ya el MonteCarlo, a la hora de los resultados me empecé a confundir, puesto que realmente se sabe cuándo me pueden ganar, pero no cuándo puedo ganar yo, y no le vi sentido al hecho de que tuviese que reflejar mis jugadas, a fin de cuentas, solo se puede entrar a los resultados si no hay más movimientos, y en ese caso es un rey ahogado a mí y por ende la victoria de la máquina. Deje el código de forma que los patrullas hiciesen lo posible para pillarme y yo en mis turnos como puedo tomar una decisión propia pues cumplir mi misión personal que es matar al objetivo y salir. Aparte de ello la toma de decisiones era lenta todavía, por lo que al final consideré dejar Montecarlo para tirar por algo más intuitivo

tratándose de un “corre que te pillo”, osea, un A* constante con cada uno por cada turno, de forma que la primera alternativa del camino mínimo actual fuese el movimiento a tomar (Además, el A* aplicado aquí es el mismo que el del caso del score, por lo que ya es cuestión de parámetros). De todas formas, he dejado el código de MonteCarlo por si lo quiere ver e incluso probarlo.

Decisión de contratos:

El Machine Learning ha sido una de las cosas que más me ha preocupado puesto que no le veía aplicación en mi proyecto.

Aprendizaje supervisado no le veía razón alguna para usarlo, clustering podría haberlo usado en todo caso si dispusiese de muchos mapas grandes y quisiese clasificar los rangos de los patrullas, según la zona, pero para un mapa tan chico y solo 4 patrullas no vale la pena. Y redes neuronales pensé en un panel para dibujar algo y que se cumpliese, como una opción del juego o algo así, pero descarté la idea también. Al final pensé en lo que eran los árboles de decisión, y realmente quitando lo del problema del viajante mencionado antes no hay ninguna otra aplicación que necesite esto tampoco o por lo menos que vea yo, pero sí que he visto interesante hacer un “minijuego” que lo aplique. La motivación del minijuego es poder planear contratos para aumentar las probabilidades de éxito, entonces utilizando el algoritmo ID3, y habiendo creado yo a mano un dataset con unos cuantos casos de contratos fallidos y exitosos pues vas a poder en este minijuego escribir/proponer un plan para un contrato por medio de los atributos indicados, de forma que la máquina te indicará si planeado así tendrá éxito o no.

Mejora aproximada del score:

Con los NPCs tratados ya y el ID3 bien colocado me quedaba el problema del viajante del score por lo que probé implementar hormigas, de forma que creé un tipo de agentes llamados “invisibles”, de forma que fuesen los nodos unidos por aristas que tomasen el papel del grafo que recorren las hormigas. El problema que me encontré con esto fue en sí a la hora de sacar los resultados, puesto que los caminos podían no estar claros o ir variando, además de que no se me ocurrió cómo sacar la información de ahí, entonces estuve entre PSO usando como coordenadas la asignación del orden de los vértices o AG usando un cromosoma con tantos genes como nodos/patches de interés hay, y siendo un cromosoma de permutación, para decidir el orden de cada uno. Decidí por estar más familiarizado en usar AG, y a la hora del fitness la parte de las prioridades como “si coges algo que hay antes una puerta sin llave” las he podido definir, el problema que me he encontrado ha sido que no podía usar el A* para penalizar distancias por ser un contexto de tortugas, entonces por falta de tiempo lo he dejado sin describir en ese sentido, por lo menos la decisión se puede tomar, aunque no sea a lo mejor el mínimo camino. El crossover también fue un problema al no poder cortar y unir dos cromosomas de forma que perdiesen la coherencia, por lo que directamente he hecho que no se puedan mezclar y en todo caso que se muten cambiándose un cromosoma él mismo el valor de un gen determinado por el de otro concreto de él.

Seguramente no sea lo más óptimo, pero no se me ha ocurrido más con el tiempo que me ha quedado.

Resultado final:

Dicho todos los contratiempos que he ido teniendo, y las ideas que tenía al principio habrás notado que mi proyecto se ha reducido a algo más distinto incluso “simple” de lo que propuse a principio.

Los requisitos de los disfraces y lo de matar patrullas no lo he podido al final añadir por la dificultad de matar a un patrulla como lo tengo programado, además de que los tipos de patrullas es una pérdida de tiempo por la misma razón que el clustering.

Y como podrás ver en el código o incluso en las anteriores definiciones de los códigos, muchas cosas las he puesto estáticas para que la dificultad no se me elevase más de la cuenta, por lo que el tener más de un mapa no valdría la pena al no ser del todo “multiplataformas” el proyecto.

Pero aun todo esto, que es lo que a mí por lo menos me ha hecho sentirme satisfecho con mi proyecto, he podido mantener la esencia del juego, un juego donde debes cumplir el contrato lo antes posible, medido con el score con clasificación final que tiene de antemano el AG para luego usar A* en cada patch en el orden indicado, para al final comparar con un contador que tendrás y que incrementa por cada decisión que tomes el total de alternativas que hay desde que sales de tu punto de partida hasta que llegas a la salida (estatua obtenida incluida), sin que te detecten, ya que como te detecten los patrullas serán capaces de detectar las mejores alternativas para pillarte lo antes posible, y respetándose entre ellos. Además, con el complemento adicional del ID3 para planear contratos.

El juego funciona, y mantiene la esencia de lo que es el juego al que me he inspirado, osea, Hitman.