# Memoria de la Practica 3

### Alumno : Leon Elliott Fuller

# Introduccion

En esta memoria iré explicando el **avance** de la práctica, es decir, los distintos **planetamientos** de la práctica junto con sus distintas **implementaciones** y métodos. A lo largo de la memoria se va a tener en cuenta lo siguiente: * **Diseño** de los métodos propuestos * **Implementación** de los métodos propuestos * **Alternativas** de los métodos propuestos * **Ventajas** y **Desventajas** de los métodos propuestos * Alternativas de los métodos propuestos * El por qué falla/triunfa un método/algoritmo

Aunque en resumen, lo realmente importante es dar con una **buena funcion heurística**, algo que a simple vista no se puede razonar. Es por ello, que empezaré por lo más básico (el tutorial) hasta avanzar a lo que sería la implementación final.

# Objetivo

El objetivo de esta memoria es intentar ir explicando y plasmando las ideas / correcciones / problemas que van surgiendo a lo largo de la práctica con el objetivo de plasmar cierto avance, además de poder disfrutar de la puntuación máxima asignada a esta tarea.

# Heuristica

De este primer avance podemos concluir que tendremos que tener en cuenta los distintos aspectos del juego para añadir en nuestra heurística: * Formar una **barrera** * La **distancia** requerida para llegar a la **meta** * La **distancia** requerida para llegar a una **casilla segura** * **Comer** cualquier ficha posible * Intentar **priorizar** cuando se pueda, **sacar** las fichas de **casa**, es decir, mientras más fichas tengamos en el tablero, más chances tenemos de coger objetos, comer fichas enemigas/aliadas, crear barreras, etc. Tener fichas en el tablero aumenta significativamente nuestros chances de ganar

# Heuristica v2

## Planteamiento

Al implementar la heuristica, voy a priorizar lo siguiente de mayor a menor: * Llegar a la **meta**: *esto implica **cualquier** moviemiento que permita a la ficha llegar a la meta. Ej –> comerse a una ficha, usar la bala, etc..* * Tambien implica el **sacrificio** de una ficha aliada, aqui es donde se distingue el uso de la

**cooperacion** entre colores, es decir, una toma el rol de **apoyo** y el otro el rol de **ataque**. Aunque esto lo implementaré un poco más adelante en la práctica, ya que no es sencillo de ver.

- Llegar al **pasillo** de la meta (las 8 casillas antes de la meta)
- **Comer** una ficha **enemiga**
- Formar una **barrera** *(de alguna manera, formar una barrera implica que ambas fichas estén en una casilla de posicionamiento seguro)*
- Llegar a una **casilla segura**
- **Sacar** una ficha de **casa**
- **Coger** un **objeto** del tablero
- Ya lo último sería simplemente **analizar** la **posicion** en la que esta la ficha, el valor corresponderia con la distancia necesaria para llegar a la meta.

### Inconvenientes

Como era de esperar, es evidente que esta heurística no es buena, pues trata a todos los objetos por igual, es decir, lo suyo es aplicarle valores a cada objeto dependiendo del estado del tablero, algo que tendré en cuenta al mejorar esta heurística. * Asignarle **valores dinámicos** a los **objetos** (dependiendo del estado del tablero los objetos son mejores/neutros). * Asignarle **prioridades** a los objetos, es decir, si hay una **ficha** enemiga **cerca** de la **meta** es muy bueno usar la **concha azul**, incluso si hay dos fichas enemigas formando una barrera, se usaría la concha para eliminar ambas fichas que forman la barrera y darle via libre a alguna ficha que quedó bloqueada por la barrera.

## Implementación Poda Alfa-Beta

No tiene mucha complicacion con respecto a la funcion minimax, simplemente voy a dejar el pseudocodigo que he utilizado para implementarla:

### Pseudocodigo

```
function minimax(position, depth, alpha, beta, maximizinPlayer)
    if depth == 0 or game over in position
        return static evaluation of position;

    if maximizingPlayer
        maxEval = -infinity
        for each child of position
            eval = minimax(child, depth, -1, alpha, beta,  false);
            maxEval = max(maxEval,eval);
            alpha = max(alpha,eval);
            if beta <= alpha break;
        return maxEval;
```

```
else
    minEval = +infinity
    for each child in position
        eval = minimax(child, depth, -1, beta, true);
        minEval = min(minEval,eval);
        beta = min(beta,eval);
        if beta <= alpha break;
    return minEval;
```

# Encuentro Final

## Observaciones

Con las **implementaciones anteriores**, he llegado al verdadero **fracaso**, es por ello, que voy a implementar una heuristica desde 0 para al menos intentar ganarle a los dos primeros ninjas, ya que me estoy quedando sin tiempo y tengo bastantes examenes por delante.

Al ver la heurística jugar varias partidas, me estoy dando cuenta de algo **fundamental**, y es que, si tengo una ficha de color azul a 20 de **distancia** para ganar, pero tengo 3 fichas de color verde a 7 de distancia para ganar, la heurística se lo toma de la misma manera que si hay una ficha de color azul a 20 casillas de ganar la partida y 3 fichas de color verde en casa, y no estoy muy seguro de por qué, no obstante, es algo que tengo que modificar para poder ganarle al ninja 3.

La heurística anterior (heuristica 3) le asigna valores subjetivos a los objetos, algo que no está del todo mal, pero voy a intentar configurarlo de manera que se puedan 'equilibrar' las ponderaciones de los objetos, me refiero a que por ejemplo, si el rival tiene un caparazón azul, mi 'counterplay' sería tener la bocina, para evitar el ataque, además de evitar que el rival avance tantas casillas como se le asigne por comerme a una o dos fichas (dependiendo de si mis fichas están formando una barrera, pueden ser golpeados ambos, algo MUY malo para mi).

No obstante, por las partidas que estoy jugando sin tener en cuenta los objetos, tampoco va tan mal, sino que creo que el jugo de la practica está en lo que he comentado anteriormente sobre las **distancias**. Tambien, he de tener en cuenta y admitir que he buscado informacion acerca de estos juegos sobre distancias, y aunque hayan objetos, caminos distintos, variedades, etc lo imprescindible para analizar la situacion del tablero de la partida, o del estado del juego es lo cerca que estás de ganar, al fin y al cabo, **si mi rival es un deportista de atleta y yo un simple campesino, si a mi me ponen a 1cm de la meta y a él a 1metro, voy a ganar debido a las distancias, independientemente si mi rival me podría superar en las mismas condiciones de distancia**, entonces creo que la clave del éxito está ahí. Además mi teoría se refuerza con el hecho de que los ****objetos** solamente pueden ser escogidos y usados una unica vez, por tanto, **son** una **herramienta** para ganar, pero realmente no hay

que hacerle mucho incapié.

Por tanto, creo que la clave está en ir asignando valores subjetivos para ponderar los datos e ir probando las partidas para ver cual es el que tiene mas exito, entonces la idea es la **clave** pero lo que va a hacer ganar la mayoría de veces son las **constantes** que voy modificando para adaptarlos al mejor juego posible.

## Heuristica

Con lo descrito anteriormente, he hecho varias modificaciones que voy a resaltar a continuacion: * Enfocar mas el problema a las distancias que a los objetos, pienso que los objetos son una **herramienta** para ganar pero no condicion necesaria como viene a ser las distancias a la meta * Sinopsis entre objetos, por ejemplo: * El **rival** posee **caparazon** azul o rojo, por tanto un buen tablero debe de contener a la **bocina** para bloquear el ataque e impedir que el enemigo avance junto con que elimine la/s ficha/s con la que interactue el caparazon * Solamente he considerado los objetos que proporcionan distancias, es decir, la bala, la seta, los caparazones (porque eliminan fichas enemigas), y les he asignado valores cerca de lo que proporcionan en distancia, es decir, como la bala suma 45 casillas, pues le he asignado un valor de 40, para la seta igual, suma minimo 8, por tanto le asigno un valor de 6. Esto ha incrementado mucho la mejora del bot.

- Añadir la condicion de si hay fichas en casa, es peor, algo sencillo de entender pero que no habia tenido en cuenta anteriormente.
- Siempre intentar valorar que mis fichas esten en el tablero, esto es algo que va enlazado con las distancias.

Yo creo que la clave ha sido lo siguiente: * Al considerar las distancias, el mejor caso es cuando yo tengo una distancia pequeña y mi enemigo una distancia grande, pero al realizar la Poda Alfa Beta, se van a escoger tableros donde el valor de la heurística sea lo más grande posible, esto se traduce en que tanto mi rival como yo tengamos distancias grandes, algo que evidentemente no se quiere, por tanto, despues de echar un buen rato pensando, se me ha ocurrido hacer: * Para cada jugador * Para cada color, recorrer todas las fichas del mismo y sumar la distancia de cada ficha hacia el objetivo, como queremos que nuestras fichas esten lo mas cerca posible de la meta, lo guardo en una variable que vaya sumando el conjunto de distancias de las fichas de cada color. * Compruebo si las fichas están en casillas seguras, si estan en el tablero, etc.. Esto lo he tomado de la heuristica ValoracionTest

Con todo esto, devuelvo la distancia de mis fichas en negativo, teniendo en cuenta la distancia minima (que es lo mismo que maxDistancia - miDistancia) y le sumo algo de las distancias maximas (max*0.35 para que se tenga algo en cuenta, la constante la he calculado a partir de desarrollo empirico), para que tambien se tengan en cuenta. Devuelvo mi puntuacion respecto a objetos, fichas en casillas seguras, etc. Va bastante bien, de hecho le vence a los 3 ninjas, aunque se podria mejorar un poco, estoy satisfecho.