

ABALONE

Proyecto Practico De Inteligencia Artificial

Mario German Baquero Cedeño

201667471-3743

[mario.baquero@correounivalle.edu.co](mailto:mario.baquero@correounivalle.edu.co)

Andrés Felipe González Rojas

201759599-3743

[andres.gonzalez.rojas@correounivalle.edu.co](mailto:andres.gonzalez.rojas@correounivalle.edu.co)

Andrés Felipe Medina Tascón

201667602-3743

[andres.medina@correounivalle.edu.co](mailto:andres.medina@correounivalle.edu.co)

Oscar Alexander Ruiz Palacio

201667600-3743

[ruiz.oscar@correounivalle.edu.co](mailto:ruiz.oscar@correounivalle.edu.co)

Facultad de Ingeniería

Escuela de Ingeniería de Sistemas y Computación

Programa Académico de Ingeniería de Sistemas

Tuluá, Junio 11 del 2018

***Resumen.*** Este documento presenta el análisis y diseño del segundo proyecto práctico del curso de Introducción a la Inteligencia Artificial, contiene la definición del problema, abordaje del mismo, donde se presenta como se implementó el árbol minimax, se analizan las heurísticas implementadas, y se finaliza con las conclusiones del proyecto.

***Palabras clave.*** Min, Max, Heurística, Inteligencia Artificial, Abalone.

1. **Introducción**

Abalone es un juego de mesa y de estrategia diseñado para 2 jugadores por Michel Lalet y Laurent Levi, en el que cada jugador controla canicas de colores opuestos a las del adversario sobre un tablero hexagonal, las canicas se pueden mover según las reglas del juego, el objetivo del juego es sacar del tablero seis canicas del oponente.

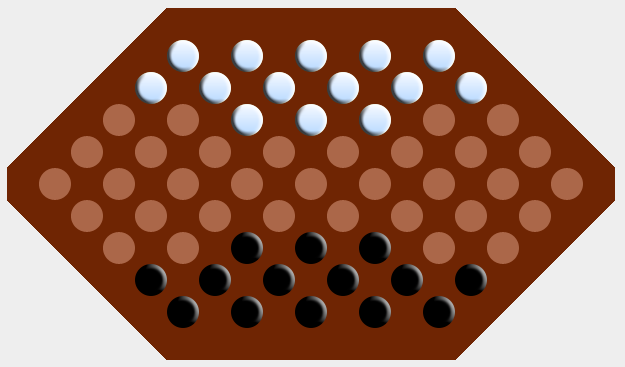
Máximo se pueden mover 3 canicas. Esas 3 canicas pueden empujar 2 de su color contrario, de la misma forma 2 canicas solo pueden empujar 1 canica del color contrario y no se puede empujar canicas en el sentido que estén intercaladas.

Representación de archivo de texto:

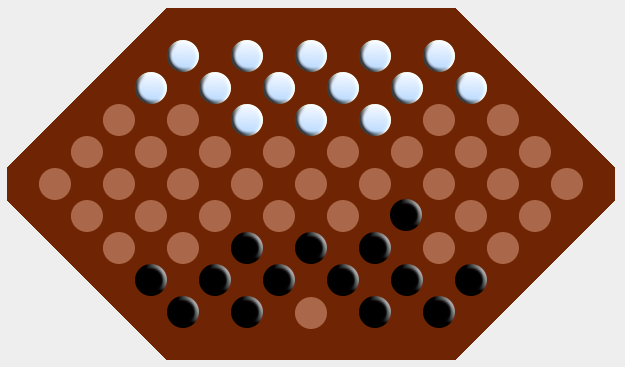
|  |  |  |
| --- | --- | --- |
| Elemento | Representación | Imagen |
| Espacio | **0** |  |
| Ficha IA | **1** | **C:\Users\Oscar\Desktop\Abalone\src\images\blanco.png** |
| Ficha Jugador | **2** | **C:\Users\Oscar\Desktop\Abalone\src\images\negro.png** |
| Espacios en el tablero | **3** |  |
| Espacios a no mostrar | **4** |  |
| Lados Izquierdo superior | **5** | **C:\Users\Oscar\Desktop\Abalone\src\images\borde.png** |
| Lados derecho superior | **6** | **C:\Users\Oscar\Desktop\Abalone\src\images\borde2.png** |
| Lados izquierdo inferior | **7** | **C:\Users\Oscar\Desktop\Abalone\src\images\borde3.png** |
| Lados derecho inferior | **8** | **C:\Users\Oscar\Desktop\Abalone\src\images\borde4.png** |

Durante cada turno, una ficha se puede mover de la siguiente manera (aplica también para las fichas de la maquina):

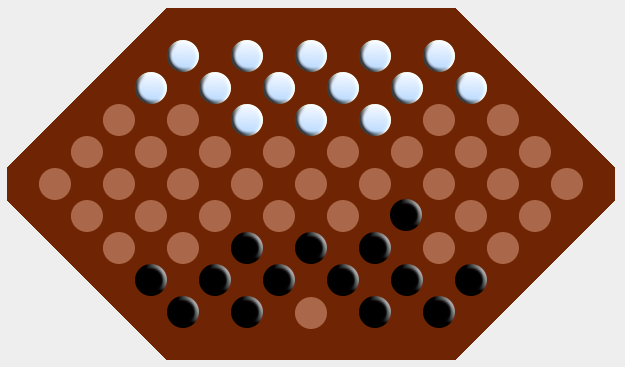
* Hacia los lados



* Línea diagonal 45 positivo o negativo.



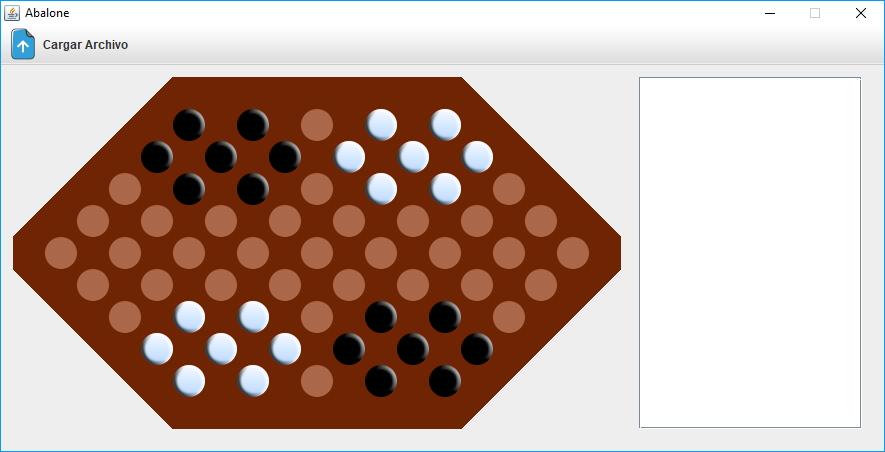
* Línea diagonal 135 positivo a negativo.



El objetivo del juego es sacar del tablero las fichas del adversario, el juego termina cuando alguno de los dos jugadores pierda 6 fichas.

La aplicación debe leer un archivo de entrada con el siguiente formato que representa el tablero.

Ejemplo:



Datos en el archivo:

**4 4 4 4 5 3 3 3 3 3 3 3 3 3 6 4 4 4 4**

**4 4 4 5 3** **2 3** **2 3** **0 3** **1** **3** **1 3 6 4 4 4**

**4 4 5 3** **2** **3** **2** **3** **2** **3** **1** **3** **1** **3** **1** **3 6 4 4**

**4 5 3 0** **3** **2** **3** **2** **3** **0** **3** **1** **3** **1** **3** **0** **3 6 4**

**5 3 0** **3** **0** **3** **0** **3** **0** **3** **0** **3** **0** **3** **0** **3** **0** **3 6**

**3** **0** **3** **0** **3** **0** **3 0** **3** **0** **3** **0** **3** **0** **3** **0** **3** **0** **3**

**7 3** **0** **3** **0** **3** **0** **3** **0** **3** **0** **3** **0** **3** **0** **3** **0** **3** **8**

**4 7 3** **0** **3** **1** **3** **1** **3** **0** **3 2** **3** **2** **3** **0** **3 8 4**

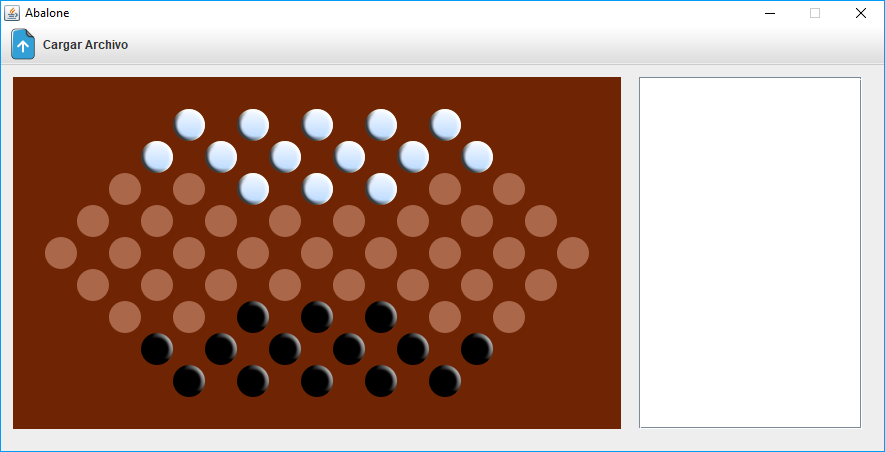
**4 4 7 3** **1** **3** **1** **3** **1** **3** **2** **3** **2** **3** **2** **3 8 4 4**

**4 4 4 7 3** **1** **3** **1** **3** **0** **3** **2** **3** **2** **3 8** **4 4 4**

**4 4 4 4 7 3 3 3 3 3 3 3 3 3 8 4 4 4 4**

Es recomendable no cambiar los valores en negro, para tener una manera más agradable de ver el tablero.

O para mayor comodidad, también se puede cargar un archivo de la siguiente manera:



Datos en el archivo:

3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

3 3 3 3 3 1 3 1 3 1 3 1 3 1 3 3 3 3 3

3 3 3 3 1 3 1 3 1 3 1 3 1 3 1 3 3 3 3

3 3 3 0 3 0 3 1 3 1 3 1 3 0 3 0 3 3 3

3 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 3

3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3

3 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 3

3 3 3 0 3 0 3 2 3 2 3 2 3 0 3 0 3 3 3

3 3 3 3 2 3 2 3 2 3 2 3 2 3 2 3 3 3 3

3 3 3 3 3 2 3 2 3 2 3 2 3 2 3 3 3 3 3

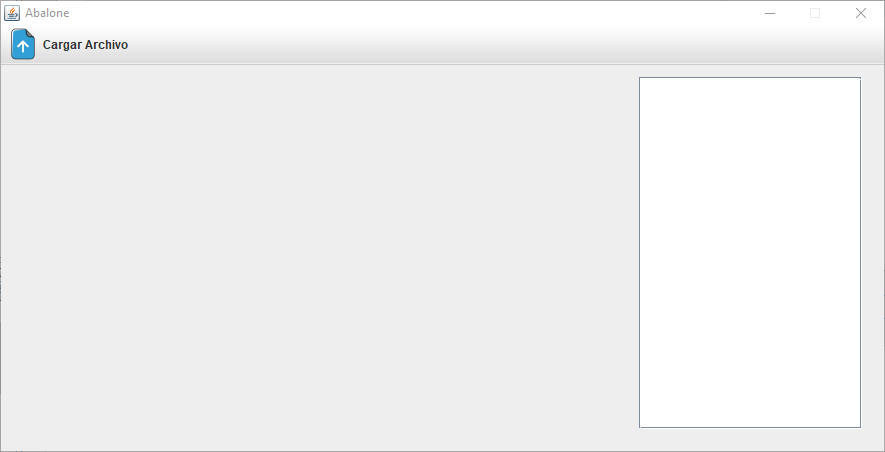
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

1. **Abordaje del problema**

Se abordara el problema utilizando un método para minimizar la perdida máxima esperada en juegos contra adversarios, minimax.

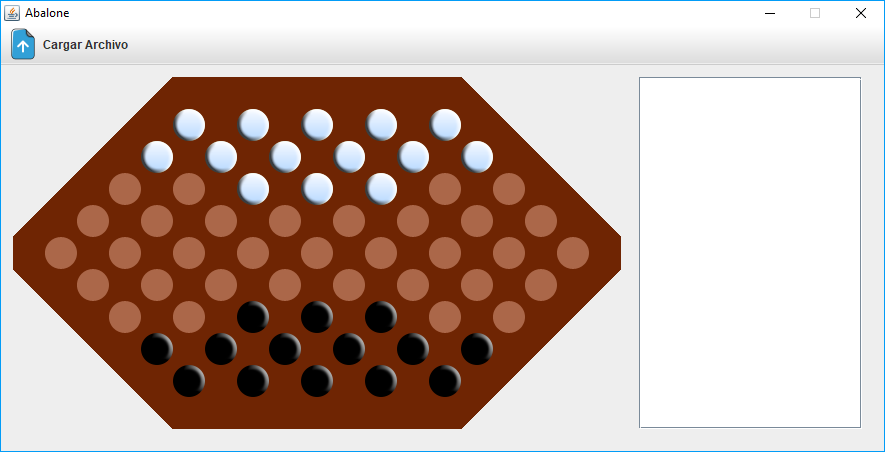
* 1. **Descripción de la Interfaz**

La implementación se inicia con una ventana inicial:



Esta tiene una barra de herramientas en donde se puede elegir el archivo que generara el tablero del juego.

La ventana principal está dividida en 2 partes:

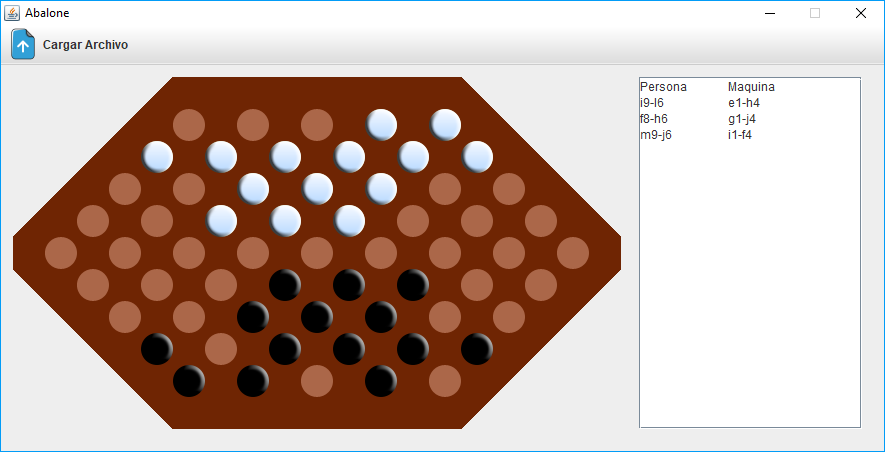


**2**

**1**

1. Zona del tablero: En esta zona se mostrara el tablero del juego que se genera según el archivo.
2. Zona Información: En esta zona se mostrara las jugadas realizadas, tanto de la persona como de la máquina.

Así se verá la interfaz cuando hayamos elegido un archivo y ya se hayan realizado jugadas de ambas partes.



* 1. **Estructura de la implementación**

El proyecto consta en total de 9 clases, las cuales se explican a continuación junto con sus métodos.

Además, cuenta con un paquete llamado Images, el cual contiene las imágenes de las fichas, y diferentes bordes para mostrar de una manera más agradable el tablero.

**Abalone**: Esta es la clase principal del programa, desde acá arranca la aplicación.

**Tablero**: Esta clase se encarga de crear la interfaz gráfica del programa y contiene los eventos de todas las fichas, estos eventos corresponden a los movimientos que hace cada ficha para calcular si es posible su movimiento y si es posible empujar rivales.

**CargarArchivo**: Esta clase se encarga de realizar la carga del archivo de texto plano para que sea visualizado en la interfaz.

**Angulo:** Se encarga de calcular el valor del Angulo 45 y Angulo 135 para una ficha seleccionada a mover.

**Imagen:** Se encarga de darle la forma a las imágenes para poder mostrarlas de forma circular.

**Posiciones:** Es la clase donde se crean las posiciones del tablero con las que se puede interactuar.

**Nodo:** Es la clase encargada de manejar los atributos de cada nodo, cada nodo contiene un identificador, un tablero, su heurística centro, heurística borde, heurística total, heurística diferencia de fichas, heurística tres en línea, booleano expandido, profundidad en el que se encuentra, jugada realizada y el identificador del padre que extendió.

**Movimiento:** Es la clase que se encarga de calcular todos los movimientos posibles que puede realizar una ficha a mover y verificar la cantidad de fichas rivales que puede empujar.

Para esto, a partir de su posición, se verifica todas las posiciones referentes al Angulo 45 positivo y negativo, Angulo 135 positivo y negativo y movimiento en el eje X positivo y negativo.

**Minimax:** En esta clase, se crea el árbol mirando las jugadas posibles de ambas partes con una profundidad de 3, generando la heurística para cada nodo, seguido, se recorre el árbol eligiendo un camino por donde irse con minimax y finalmente, elige una jugada con base a ello.

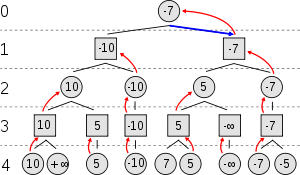
* 1. **Algoritmo Implementado**

El algoritmo que se implementó para la realización del proyecto fue el algoritmo Minimax.

El algoritmo Minimax se aplica para el caso de juegos de dos participantes, MAX y MIN. Estos algoritmos permiten encontrar la acción que debería realizar MAX para minimizar la perdida en el juego.

**Pasos:**

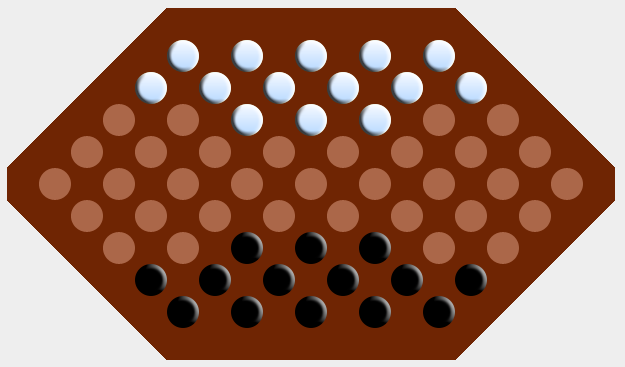
1. Se genera el árbol del juego y se generan los nodos hasta llegar a un estado terminal.
2. Calcula el valor de la utilidad de cada nodo, aplicando 4 tipos de heurísticas implementadas.
3. Calcular el valor de los nodos superiores a partir del valor de los inferiores. Según nivel si es MAX o MIN se elegirán los valores mínimos y máximos representando los movimientos del jugador y del oponente.
4. Elegir la jugada valorando los valores que han llegado al nivel superior.



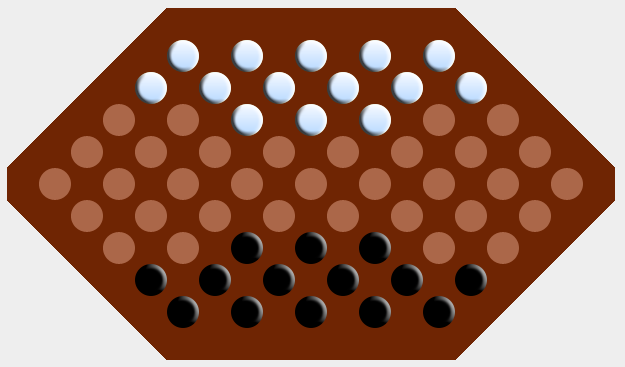
Ejemplo Árbol MinMax

* 1. **Heurísticas**

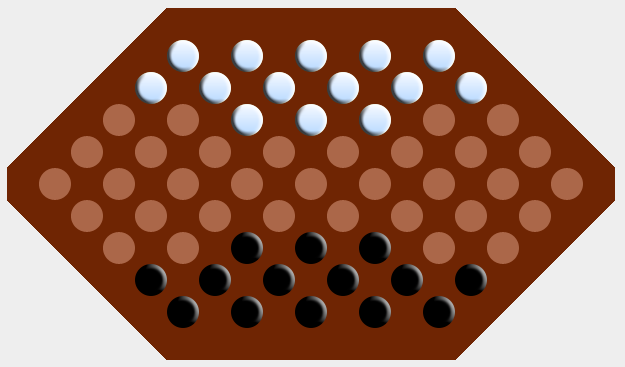
La primera heurística se basa en la diferencia de la cantidad de fichas que se encuentran en el centro del tablero. Esto es debido a que cuando las piezas están más cercas al centro del tablero, menos posibilidades tienen de ser expulsadas fuera, y también a que da más posibilidades de que se encuentren filas de 3, haciendo que sea imposible un empuje



La segunda heurística consiste en la diferencia de la cantidad de fichas que hay en los bordes. Esta heurística se implementó con la finalidad de tener en cuenta cuantas fichas están en riesgo de ser expulsadas del tablero.



La tercera heurística consiste en la diferencia de la cantidad de veces que se encuentran 3 fichas en línea, por ejemplo:



Esta heurística se implementó debido a que es más beneficioso tener fichas en 3 en línea porque da la posibilidad de empujar el máximo número de fichas posibles del rival.

La cuarta heurística consiste en la diferencia de fichas que hay, siendo la cantidad de fichas de la maquina menos la cantidad de fichas que posee el jugador.

1. **Conclusiones**
2. **Referencias**

**[1]** https://es.wikipedia.org/wiki/Minimax

**[2]** https://es.wikipedia.org/wiki/Abalone\_(juego\_de\_mesa)

**[3]** Diapositivas Curso Inteligencia Artificial, Universidad del Valle Sede Tuluá, Periodo Febrero-Junio 2018.