SIMULADOR DE MATES EN AJEDREZ

Resumen—Este simulador propone una solución complementaria al curso E-learning de aprendizaje de ajedrez en la UMNG.

Con el cual es estudiante resolverá problemas específicos preestablecidos con el objetivo de ganar la partida, aparte de eso debe realizarlo en cuatro (4) o menos movimientos, siendo siempre ejercicios propuestos para solucionar por máximo en esa cantidad de movidas. La aplicación contará con información sobre el tablero de ajedrez y sobre las piezas, basados en la notación algebraica de ajedrez, utilizada como estándar actual para escribir una partida. A su vez y por solicitud del tutor de la materia electiva de ajedrez en la UMNG, el simulador deberá permitir al usuario practicar sus conocimientos a nivel global, por ende el usuario deberá poder jugar una partida contra él.

I. INTRODUCCIÓN

Dadas las sugerencias por el docente evaluador de la propuesta del proyecto de grado, se consideró que era muy extenso realizar el simulador desde cero, ya que solo la realización del mismo podría implicar un proyecto aparte.

Por tal motivo se optó por buscar una solución ya existente e implementarla y adecuarla para el propósito del proyecto, logar que el estudiante aprenda a jugar ajedrez.

Sin embargo se tomó como soporte para la adecuación del simulador la siguiente información, basada en el respectivo proceso de investigación con el fin de dar solución a este aspecto del proyecto, como fuente fundamental la inteligencia artificial.

¿Si podemos crear una maquina capaz de jugar ajedrez, estaremos más cerca de entender el pensamiento humano?[1]. La respuesta fue no, ya que el humano es un ser inteligente, y la maquina

tiene inteligencia, aunque parezcan términos similares, significa que la maquina aunque es mejor que el humano para jugar ajedrez, no sabe hacer nada más, no sabe si su oponente está haciendo trampa[2], lo que nos quiere decir que basar la inteligencia artificial en el pensamiento humano no traería beneficios ya que la mente humana se puede distraer fácilmente, la maquina hará lo que tenga que hacer sin importarle su entorno, por eso son tan buenas en lo que hacen.

Recordando nuestro límite de cuatro movimientos, ahí yace una de las debilidades de las maquinas, estas se concentran en resultados rápidos, algunas incapaces de ver estrategias que desarrollan lentamente a lo que pasa el tiempo, sin embargo otras que se enfocan y dan prioridad a un análisis estratégico a través del tiempo, lo cual propone que un jugador capaz de ganar a una máquina, debe ser bastante diestro en la práctica del ajedrez, [3]es como caminar en un túnel oscuro hacia la luz, si nunca vez la luz te pierdes y no sabes para dónde ir.

De este modo podemos ver que no solo ahorramos la necesidad de gran capacidad de procesamiento al solo usar pocos movimientos, y también evitamos llegar a un campo donde la computadoras tienen problemas para empezar, debido a la cantidad de posibles movimientos que se pueden llevar a cabo al comienzo del juego y el nivel de análisis que deba realizarse en las variables posteriores para de tal modo lograr la victoria.

El ajedrez tiene su origen en la India, más concretamente en el Valle del Indo,[5] y data del siglo VI D.C. Originalmente conocido como Chaturanga, o juego del ejército, se difundió rápidamente por las rutas comerciales, llegó a Persia, y desde allí al Imperio bizantino, extendiéndose posteriormente por toda Asia. La mayoría de los historiadores coinciden en ubicar el origen del ajedrez en la India en el siglo VII. El mundo árabe, adoptó el ajedrez con un entusiasmo sin igual: estudiaron y analizaron en profundidad los mecanismos del juego, escribieron numerosos tratados sobre ajedrez y desarrollaron el sistema de notación algebraica.

El juego llegó a Europa entre los años 700 y 900, a través de la conquista de España por el Islam, aunque también lo practicaban los vikingos y los Cruzados que regresaban de Tierra Santa. En las excavaciones de una sepultura vikinga hallada en la costa sur de Bretaña, se encontró un juego de ajedrez, y en la región francesa de los Vosgos se descubrieron unas piezas del siglo X, de origen escandinavo, que respondían al modelo árabe tradicional. Durante la edad media España e Italia eran los países donde más se practicaba. Se jugaba de acuerdo con las normas árabes (descritas en diversos tratados de los que fue traductor y adaptador Alfonso X el Sabio), según las cuales la reina y el alfil son piezas relativamente débiles, que sólo pueden avanzar de casilla en casilla.

La era moderna del Ajedrez, sin embargo, puede ser ubicada en el siglo XV, donde las piezas obtuvieron la forma que tienen actualmente. El primer analista serio del juego fue el español Ruy López de Segura (Siglo XVI), quien en 1561 describió las reglas que aún se usan.

Durante los siglos XVI y XVII el ajedrez experimentó un importante cambio, y la reina se convirtió en la pieza más poderosa, en cuanto a su movimiento se refiere, del tablero. Fue entonces cuando se permitió a los peones avanzar dos casillas en su primer movimiento y se introdujeron la regla conocida como en passant ('al paso'), que permite capturar el peón que sigue su marcha y no come la ficha que se le ha ofrecido por una determinada estrategia, y el revolucionario concepto del enroque. Los jugadores italianos comenzaron a dominar el juego, arrebatándoles la supremacía a los españoles. Los italianos, a su vez, fueron desbancados por los franceses y los ingleses durante los siglos XVIII y XIX cuando el ajedrez, que había sido hasta entonces el juego predilecto de la nobleza y la aristocracia, pasó a los cafés y las universidades. El nivel del juego mejoró entonces de manera notable. Comenzaron a organizarse partidas y torneos con mayor frecuencia, y los jugadores más destacados crearon sus propias escuelas.

II. PROBLEMA

¿Cómo ayudar a los estudiantes del curso virtual de ajedrez en la práctica de los conocimientos adquiridos a lo largo del curso? Por medio de la implementación y adaptación de un simulador de ajedrez que utilice un método de búsqueda y solución que no requiera mayor esfuerzo de hardware. Para tal caso se utilizó el desarrollo realizado por el señor Don Cross, y citado como Chenard, un programa de ajedrez de código abierto disponible para Windows, que permite practicar jugando contra la máquina.

III. OBJETIVO PRINCIPAL

- Adaptar el software, para que sea capaz de cargar situaciones, que le permitan la practica al usuario, por medio de ejercicios o mates en ajedrez, en donde la solución se considera obligada o única. Con el fin de dar soporte al curso E-learning de ajedrez en la UMNG.

IV. OBJETIVOS ESPECIFICOS

- Adaptar la aplicación para que cargue ejercicios, con los cuales el estudiante del curso virtual de ajedrez, pueda practicar sus conocimientos.
- Comprender los métodos de inteligencia artificial integrados en el código ya desarrollado para validar si son útiles en la solución del problema.
- Adaptar la interfaz con una opción de carga de ejercicios en donde el software resolverá el problema, o en donde se dice al usuario quien juega y este deberá solucionar el problema.

V. DIAGRAMA DEL PROYECTO

A continuación se expone brevemente la interfaz que tendrá el proyecto.

La interfaz está compuesta de 4 partes generales.

Interfaz: Parte grafica que contiene 2 elementos; tablero, y botones de uso de la aplicación.

Tablero de ajedrez; Tablero grafico que muestra por medio de diagramas de cada pieza de ajedrez, los movimientos que son realizados o la posición a solucionar.

Espacio de Botones; En este espacio se puede encontrar los botones que permiten al usuario interactuar con el software, de los cuales se puede resaltar el botón de carga de ejercicios, parte de la adaptación realizada con el fin de lograr el objetivo del simulador.

VI. METODOLOGIA

ANTECEDENTES METODOLÓGICOS:

Claude Shannon fue el pionero que nos mostró el camino hacia la solución de partidas de ajedrez por medio de inteligencia artificial, casi cualquier derivación del ajedrez autómata viene de conceptos idealizados o planteados por el mismo.

El afirmaba que en una partida de ajedrez normal, duraban en un promedio de 40 jugadas hasta que un jugador pierda, así también por cada uno de esos turnos necesarios para llegar a esa situación cada jugador tenía la opción de al menos 30 movimientos posibles, lo que causaría un sinfín de posibles movimientos imposibles para calcular para el ser humano, pero incluso para una máquina, ya que pensar en tantas posibles jugadas toma tiempo en un procesador, lo que lleva años de posibles cálculos y no se garantizaría una respuesta, sin embargo hoy en día con los grandes avances tecnológicos, la posibilidad de cálculo y computo es mucho mejor en las maquinas, a su vez los algoritmos propuestos para la solución de determinados problemas, se encuentran mejor optimizados.

Cuando se empezó a crear los primeros software que eran capaces de jugar al ajedrez, tenían problemas para ganar las partidas, ya que muy a menudo era necesario calcular muchas jugadas para poder llevar a un jaque mate, es ahí donde Ken Thompson se le ocurrió una nueva idea para poder calcular estos finales y era la búsqueda retrospectiva, que era básicamente iniciar en el final del juego y ver si se llegó a ese resultado, este tipo de análisis dio resultado, ya que esta máquina al ser probada contra jugadores reales, no podían vencer a la máquina, lo máximo que podían lograr era un empate, este sería solo el inicio para los programas de ajedrez para ganar las partidas, uno de sus mayores obstáculos para lograr calcular un jaque mate, era el número de piezas disponibles, ya que incrementa exponencialmente las jugadas a analizarse.

Después llegarían Shredder y Fritz, utilizando las bases de datos Edward y Nalimov, lograron expandir poco a poco el número de piezas que podían ser calculadas para llegar al final del juego, este número creció hasta poder calcular con un total de 6 y 7 piezas por bando para ganar.

Y así fue como la capacidad de generar juegos completos de ajedrez que la maquina pudiera aumentar, no solo los procesadores de las maquinas eran cada vez más grandes sino que también, se implementaron diferentes y mejores métodos de búsquedas, como la llamada: "árbol de juego" en donde se analiza las jugadas de estas, se analiza las posibles respuestas para llegar a este resultado y así sucesivamente, pero este tipo de cálculo como lo imaginaran lleva un tiempo excesivamente largo de procesar por lo que se crearon criterios para llegar a estos resultados de manera más rápida y optima, como la jerarquización de las piezas, hacer que unas sean más importantes que otras, hace que se descarten varias jugadas de antemano y poder llegar al resultado final sin procesar tanta información, y para el jaque mate, suele asignarse un valor exorbitante al rey para que sin importar que, el objetivo sea atacar al rey.

[12] Las partes más importantes de un programa de ajedrez, son los generadores de movimiento, una función para evaluar estos movimientos, y otro para controlar la búsqueda, lo que hace que a partir de la posición actual del tablero el motor realiza una búsqueda en profundidad iterativa, y de estas se realizan las siguientes (árbol de juego), y con la función de evaluación se elige las mejores jugadas acatando las ordenes de prioridad que se hayan puesto en el programa.

Hoy en día uno de los mejores programas de código abierto para el ajedrez es del autor Robert Hyatt, el llamo a su código Crafty, pero también existen otros como Fruit, este último logro el segundo en el torneo mundial de 2005, perdió contra otro software llamado Zappa, donde este se llevó el primer lugar

del torneo, todos estos software están disponibles en internet ya que son de código abierto y cualquiera puede verlos.

DESCRIPCIÓN GENERAL DEL PROYECTO

Tras realizar un análisis y uso del código fuente se determinó que este cumple y facilita las siguientes acciones presentadas como sub-sistemas del mismo.

Los siguientes sub-sistemas, identifican y desglosan el proceso a seguir en pro del objetivo del simulador. Así mismo en lograr victorias rápidas en ajedrez o puntualmente buscar una solución óptima a un problema que puede ser en cuatro (4) o menos movimientos, en situaciones hipotéticas pre programadas, iniciando en:

- Subsistema de Entrada de datos: Realizada por la aplicación para ver todo el tablero y reconocer todo lo que hay en él, para que en los datos de salida de esta acción, el sistema haya podido reconocer las piezas que se encuentran en el tablero y en la posición en que están cada una de ellas, con estos datos la aplicación debe cumplir sus objetivo de ganar la partida. Para tal caso existirán varios problemas previamente guardados, los cuales contendrán las posiciones completas de las piezas del mismo en formato .PGN (Portable Game Notation), para tener una mejor idea del paso a paso de cómo se realiza la entrada de datos la dividiremos en la siguientes etapas:

    - Identificar piezas en el tablero: Este paso es el proceso para analizar la pieza a mover, donde se reconoce cada elemento que se encuentra en el tablero, las casillas solo pueden tener un elemento a la vez dentro de estas, entonces luego de analizarse el tablero el software debe saber cuáles y cuantas piezas se encuentran en el tablero en ese momento determinado.

- Analizar posición en el tablero: A continuación de saber que piezas se encuentran en el tablero, sigue el paso de reconocer en qué posición del tablero se hallan exactamente, por lo que en este paso se requiere de un tablero bien claro con coordenadas ya establecidas para por medio de este lenguaje se explique donde se encuentra tal pieza buscada.

- Finalmente se revisa la base de datos en busca de un archivo .pgn que fue guardado allí con anterioridad, esta imagen contiene información del tablero y las piezas que se encuentran dentro de este, a partir de estos datos el software puede empezar a buscar una solución para el problema presentado, pero solo si el usuario se lo ordena así.

Pero estos datos no son los suficientes para llegar a nuestro objetivo, se necesitan de varios sub sistemas capaces de procesar toda la demás información, empezando por el sistema principal de:

- Subsistema de movimiento o acción: Aquí se recibirán los datos de entrada sobre el tipo de pieza que se va a mover, ya que no todas las piezas se mueven del mismo modo, hay que saber diferenciarlas. Una vez el sistema haya sido anunciado del tipo de pieza que se va a mover, el subsistema tendrá una salida de datos con las posibilidades de movimiento de la pieza seleccionada, cabe recordar que según la situación actual del tablero u otras condiciones estos movimientos pueden variar. Sin embargo los movimientos para nuestro problema específico irán enfocados a dar mate al rey oponente, por lo cual todos los movimientos que realice la máquina, se enfocarán en la búsqueda del medio más rápido para ganar, para esto se requiere que todos nuestros criterios estén bien ordenados para dar así prioridades objetivas a la maquina como:

  - Determinar el valor de la pieza: Esto es lo que le da al software el objetivo principal de

tomar la pieza rey del oponente, ya que al ser la pieza de más valor, el software decidirá ir tras ella, causando que se analicen las mejores jugadas posibles para hacerlo, de no ser posible se recurren a segundas opciones y así sucesivamente para generar la mejor jugada posible.

- Proponer objetivo: No siempre será posible ir tras el rey o al menos no de forma inmediata, lo que significa que el software deberá idear maneras de cómo llegar a ella, con ayuda de ente criterio se puede hacer que el software a la vez que vaya por el rey también pueda ir por otras piezas de valor, situación que puede considerarse como beneficioso para este, ya que el jugador que tenga mayor "presencia" en el tablero tiene una ventaja considerable.

- Determinar qué bando juega: Este paso se realiza al mismo tiempo que se revisa la imagen en la base de datos, allí se determina por medio de una línea de código implícita en la imagen pgn, que el software puede leer y entender, para definir el color del bando que juega primero.

- Aplicación de una jugada especial: De vez en cuando puede ocurrir que en una partida de ajedrez se quiera usar uno de los movimientos especiales con el cual cuenta el juego, esto requiere de una posición casi exacta en la mayoría de los casos de ciertas piezas, este movimiento tiene un resultado no logrado por métodos normales para el juego regular de ajedrez, también este consume el turno del jugador que lo implemente ya que se considera un turno normal, la lista de movimientos de esta categoría son como: "enroque" que requiere de la torre y el rey para poder ser efectuado, y otros como el "peón al paso" que requiere de acciones entre peones de ambos bandos para poder realizarse.

- Subsistema de reconocimiento terreno: En el ajedrez no puede haber dos piezas en una misma casilla, por eso este sub sistema tendrá datos de entrada sobre donde va a ser movida una pieza, cada casilla será reconocida por un sistema de coordenadas, esta será revisada y enviará de vuelta información sobre el estado actual de la casilla, no porque la casilla este ocupada significa que la pieza no pueda moverse allí. Ya que la pieza que quiere llegar a determinada casilla, podrá capturar a la que allí se encuentre. Quedando ubicada posteriormente en la posición de la anterior pieza en dicha posición del tablero.

    - Reconocimiento de color de casilla: Aunque no muy importante para el usuario que está jugando en ese momento esto es muy útil para el software que analiza las jugadas posibles, ya que gracias a esta diferencia de colores entre casillas, puede saber más fácil como mover las piezas, por ejemplo, el rey solo se puede mover una casilla a la vez, lo que significa que no importa en qué color de casilla se encuentre el rey antes de moverse siempre terminara en una casilla del otro color, exceptuando la situación donde se mueva de forma diagonal, donde mantendrá en color inicial, así mismo se puede aplicar para los alfiles que no importa cuántas casillas recorran siempre se quedaran en su color original.

    - Casillas ocupadas: Las casillas ocupadas son barreras naturales para todos los tipos de piezas, excepto el caballo que puede “saltar” entre ellas, en especial para piezas del mismo color, que al ser del mismo bando, estas no pueden tomarse unas a otras, pero en el caso de que sean piezas enemigas una captura de piezas se puede llevar a cabo, lo que la pieza atacante debe hacer es moverse dentro de sus limitaciones de clase sobre la pieza que quiere tomar, y allí ser eliminada la pieza atacada, pero esto también causa que la pieza no pueda seguir moviendo y termina el turno del jugador.

- Lenguaje para la ubicación de las piezas: Este es un “lenguaje” usado en las partidas de ajedrez para explicar más fácilmente de donde y para donde se mueven las piezas, usando coordenadas verticales y horizontales con letras del abecedario y números del 1 al 8 respectivamente, esto se usa para avisar de movimientos y las piezas también tienen un nombre especial para este tipo de escritura, para diferenciarlos de otras piezas.

- Subsistema de reconocimiento de pieza atacante: Si una pieza se va a mover donde se encuentra una pieza enemiga, este sub sistema revisa los datos almacenados para procesar la acción, para realizar la toma de la pieza enemiga y reemplazándola con la pieza invasora, pero aun así no porque una pieza pueda invadir a otra significa que pueda hacerlo, hay ciertos casos en los cuales según la pieza, situación actual o consecuencia de este ataque no dejaran que esta acción sea permitida. Como en aquellas posiciones donde no se puede mover una pieza para atacar por que esta se encuentra en una posición defensora de su rey, o porque el movimiento que intenta realizar el usuario no es un movimiento valido para la pieza.

  - Pieza que captura y capturada: No siempre hay una única opción a la hora de capturar una pieza, en muchos casos hay más de una opción de capturar piezas, y no siempre una pieza puede ser capturada con una única opción sino que puede haber diferentes piezas disponibles que pueden hacer el mismo trabajo, por lo que en este proceso llega la apreciación por valor de puntos de las piezas, como se expone, las piezas tienen diferentes habilidades y características que hacen que unas sean más valiosas que otras. Por ejemplo si el usuario tiene la opción de capturar a una reina y a un peón, por sus valores, se decide fácilmente por la reina ya que es la de mayor valor entre las dos, pero ahora si esta jugada puede poner en riesgo la pieza que va a tomar la reina, se envía la de menor valor si es posible así se minimizan perdidas por si la situación no sale como esperada.

- Subsistema de reconocimiento de jaque: Luego de realizarse una acción de movimiento o captura de pieza existe la posibilidad de crearse un jaque, en el cual el rey puede ser objetivo de ataques del oponente, este sub sistema recibirá los datos de la posición del rey y quien lo tenga como objetivo al ocurrir el jaque, este sub sistema tendrá una salida de datos sobre las posibles acciones a realizar para sacar al rey del jaque, pero si no hay movimientos posibles se entra a otro sistema, se encuentran varias opciones de cómo responder a una situación como estas:

    - Escape del rey: En ciertos jaques puede darse la opción de que el rey pueda salir de esto sin ayuda de otras piezas, simplemente el rey se mueve a otra casilla donde no esté en riesgo, normalmente no muy usada ya que en este tipo de movimientos es donde se están sacrificando otras piezas que pueden ayudarnos a ganar.

    - Defender con otra pieza: No porque una opción sea viable significa que sea la mejor, lo que significa que a veces moverse lejos del peligro podría no ser una solución optima, lo que nos lleva a la opción en la cual el jugador que tiene un jaque en su contra puede usar de sus piezas y moverla entre el rey y su atacante, de este modo el jaque se termina y la partida continua su rumbo normal, muy útil ya que en muchos casos ni el rey si la pieza defensora serán blanco de ataques al estar siendo cubiertas entre si y posiblemente por otras piezas.

- Subsistema de jaque mate: Puede ocurrir luego de realizar un movimiento que haya concluido en jaque, si el rey u otras piezas del mismo equipo no pueden retirar al rey, evitar, o cubrir el jaque con ayuda de sus piezas, se considerada un jaque mate en el cual el jugador sufriendo de jaque perderá la partida y el juego terminará.

Todo el sistema de movimiento y sus subsistemas funcionan de manera lineal, lo que significa que para llegar a cierto subsistema se debe haber pasado por algunos o varios de los anteriores, pero ahora ya vimos los posibles casos de cada movimiento y sus consecuencias, complementario a esto, el código Chenard expone los parámetros bajo los cuales se realizará algunas de las acciones ya mencionadas.

Estando preestablecido como primer objetivo, ganar la partida, pero para llegar a ello se requiere analizar cada movimiento posible que se pueda realizar y elegir el más óptimo, sabiendo que las acciones del oponente deben intentar evitar llegar a este objetivo, esto casi siempre será por medio de la captura de piezas del oponente porque quien tenga más piezas tendrá la ventaja para la victoria, pero no todas las piezas tienen la misma utilidad, habrá piezas más importantes que otras para realizar este objetivo, de este modo el simulador incluye un sistema de "valor de piezas", el cual permita facilitar la toma de decisiones para cumplir el objetivo.

Podemos considerar la siguiente tabla de puntaciones:

- Peón: 1
- Caballo: 3
- Alfil: 3
- Torre: 5
- Reina: 9
- Rey: Considerado como un valor que tiene a infinito y determinado en el simulador como el máximo puntaje.

Se observa que en este método la aplicación siempre deberá buscar atacar o dar mate al rey, no necesariamente en un solo movimiento, así también cuando vaya a ocurrir un ataque a las piezas

enemigas lo haga con un margen mínimo sino nulo de perdida, así no va a arriesgar la Dama o la Torre si hay Peón disponible que pueda realizar la misma acción, de este modo tomar las piezas más importantes del oponente y proteger sus propias piezas se vuelven un sub-objetivo que ayudan a alcanzar la victoria, este criterio no significa que vaya a anular ciertas posibles jugadas ganadoras, por ejemplo si al perder la Dama significa ganar en el siguiente turno, la aplicación lo realizara, ya que por encima de todo está tomar el rey enemigo.

- Subsistema de Notación Algebraica: Este subsistema se encarga de documentar las acciones realizadas por las piezas sobre el tablero de ajedrez, haciendo comprender los sucesos realizados por medio de una notación algebraica conocida a nivel mundial, sin embargo con leves variaciones de la notación de las piezas, en donde varia la inicial de cada uno de un idioma a otro, sin embargo fáciles de comprender.

Es importante aclarar que la toma de decisiones tendrá un objetivos principal y único, realizar la serie o secuencia de pasos adecuados con el fin de buscar dar mate en menos movimientos, ya sea en la práctica de juego contra el simulador o en la solución de ejercicios, para ello la maquina deberá analizar cada una de las posibles variables de cada pieza y de cada combinación de piezas, seleccionando solo aquella que sea posible según el objetivo principal, dar mate.

DIAGRAMAS UML

DEFINICIÓN BREVE DEL PROBLEMA

Adaptar un código abierto para el propósito del curso E-learning para la enseñanza de ajedrez, esto por medio de ejercicios de mate o la posibilidad de practicar contra el simulador en un partica completa.

REQUISITOS

Funcionales

Que la aplicación pueda brindar la inteligencia adecuada para seleccionar la mejor ofensiva en búsqueda de dar mate, para ello cada pieza contara con un valor, y el objetivo de todos los procesos será el mismo, dar mate al rey oponente.

NO FUNCIONALES

El programa cuenta con un algoritmo basado en valores de las piezas, el cual buscara suplir su objetivo principal, dar mate al rey, por ende buscar el valor de vitoria, llamado agregado con un valor de 100/100 en la tabla de mejores resultados buscados.

SISTEMA

El agente será un tipo de evaluador, el cual tras observar e identificar cada una de las piezas existentes en el tablero, buscara la secuencia más óptima con el fin de cumplir el objetivo, sin

embargo este agente será un tipo de conjunto que comprenda y entienda los movimientos de las piezas que existen en el tablero, y pueda realizar una actividad común e integra entre ella con el fin de dar mate y llevar a cabo el objetivo general.

DOMINIO

El dominio se basa en cumplir un objetivo específico; dar mate en menos de cuatro (4) movimientos, para tal caso la aplicación cuenta con tiempos o movimientos que realizará secuencialmente analizando las diferentes posibilidades de juego en el tablero por parte de ambos bandos, y determinando la base de cada problema, quien da mate, si blanco o negro, para lo cual analizará al ganador con el primero movimiento y la serié que obligará al rival a realizar.

ACTORES

Los actores son:

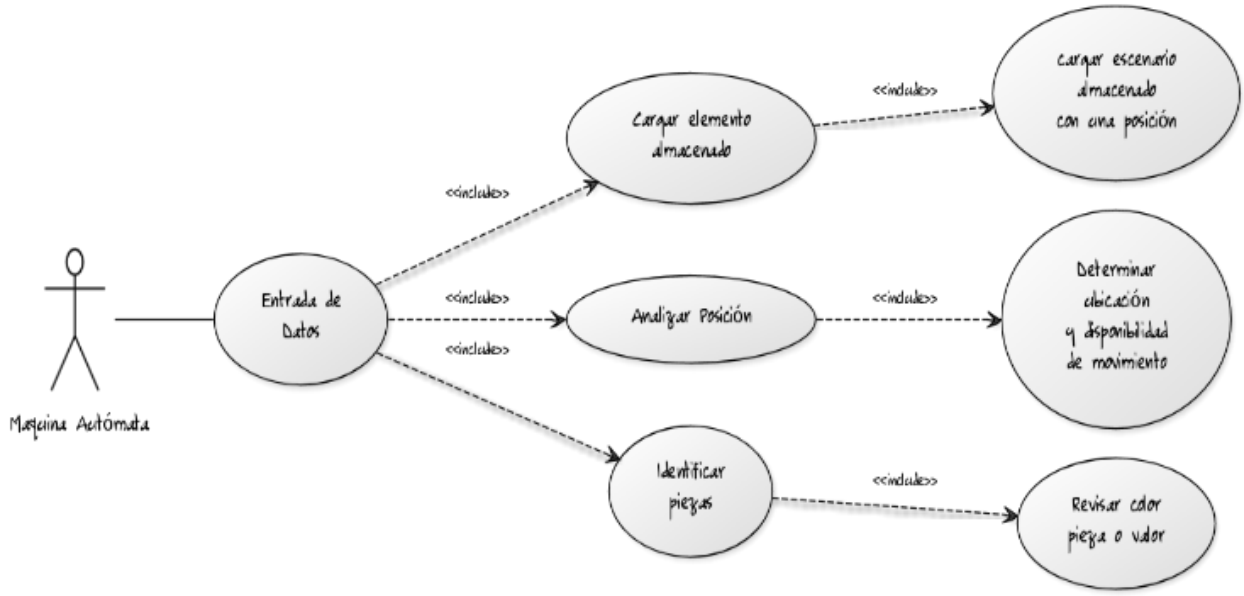
Usuario; quien podrá determinar si desea que la acción actué por sí sola, o si desea realizar actividades en modo de práctica, donde será el quien realice los movimiento, o si desea practicar jugando contra la máquina.

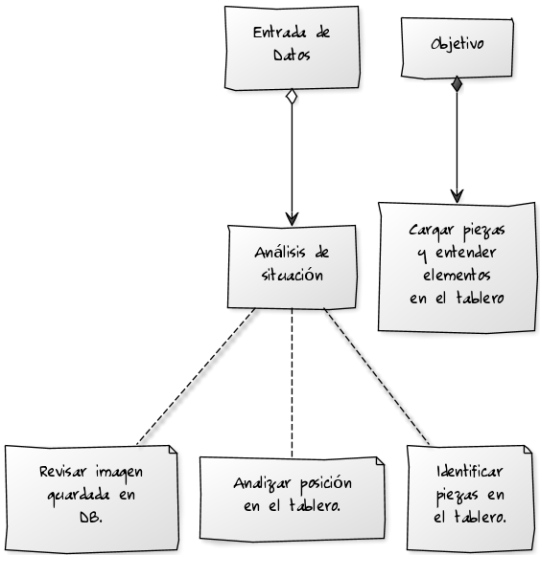
Maquina autómata: Quien determina las acciones correctas para cumplir el objetivo y ganar el juego, la cual puede ir determinada y regida por la opción de ganar y dar mate. Siguiendo los parámetros de valor de las piezas y la combinación adecuada para ganar.

Piezas: Cada pieza es considerada como un agente, ya que goza de la libertad de varios movimientos y actúan entre sí de forma diferente, sin embargo pueden ser consideradas como parte del actor máquina, ya que es este quien guiará a las piezas por una solución adecuada y pertinente.

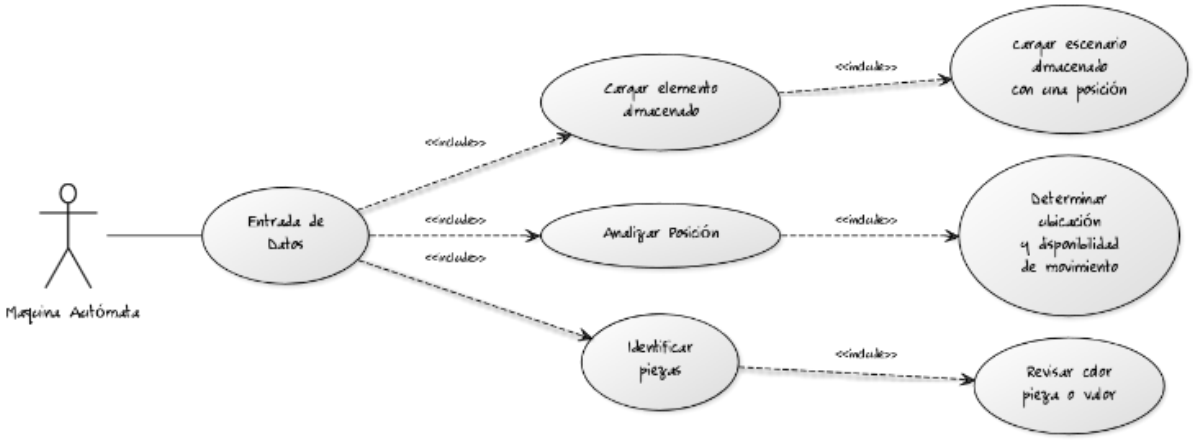
Diagramas UML de los Subsistemas

Subsistema entrada de Datos

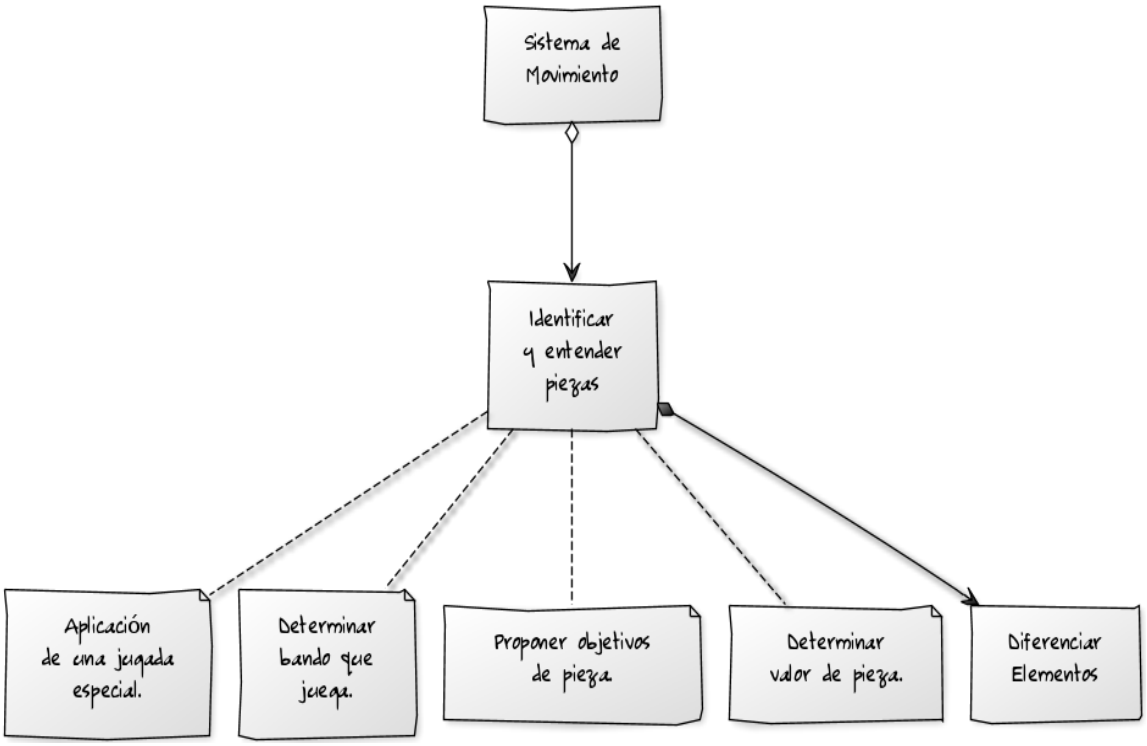




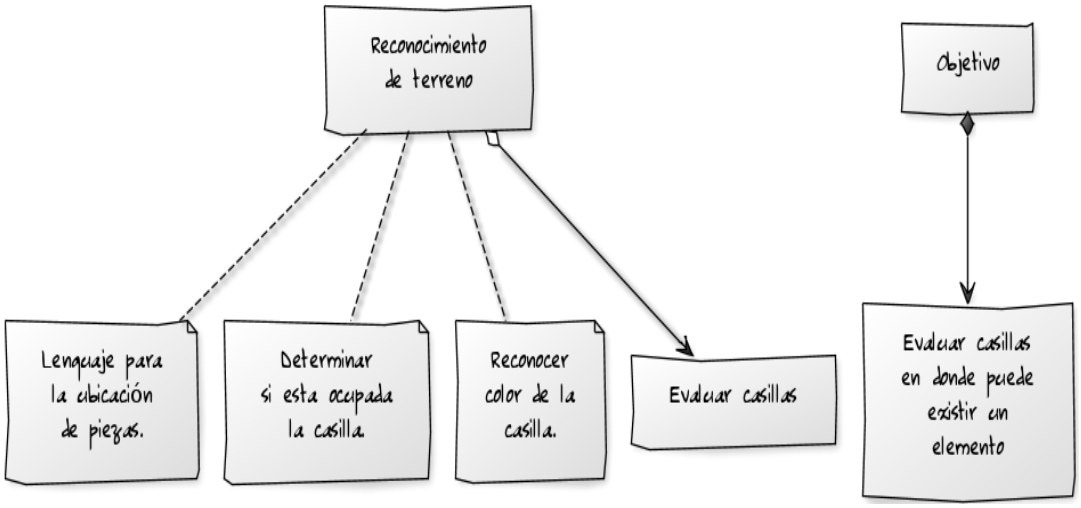
Interacción del subsistema de entrada de datos con el sistema 1, maquina autómata.



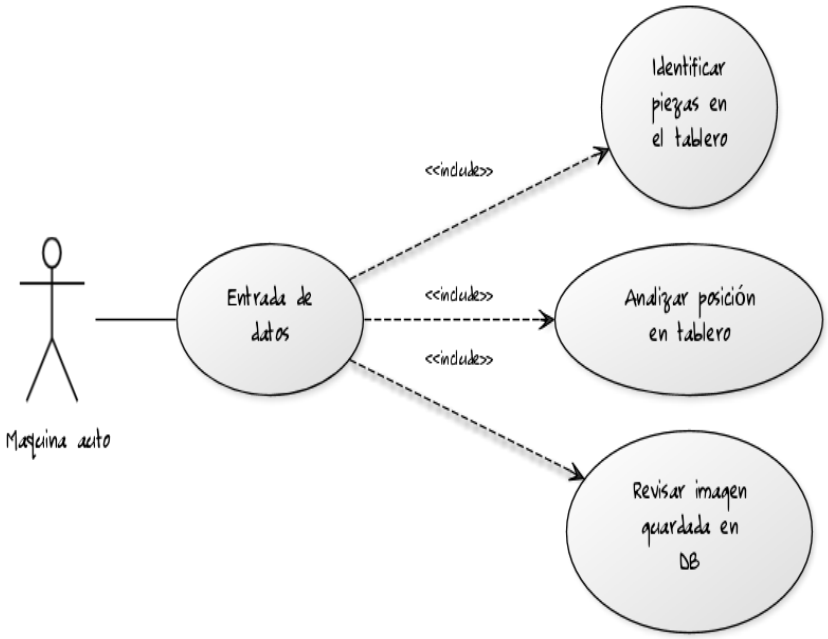
Subsistema de Movimiento y Acción

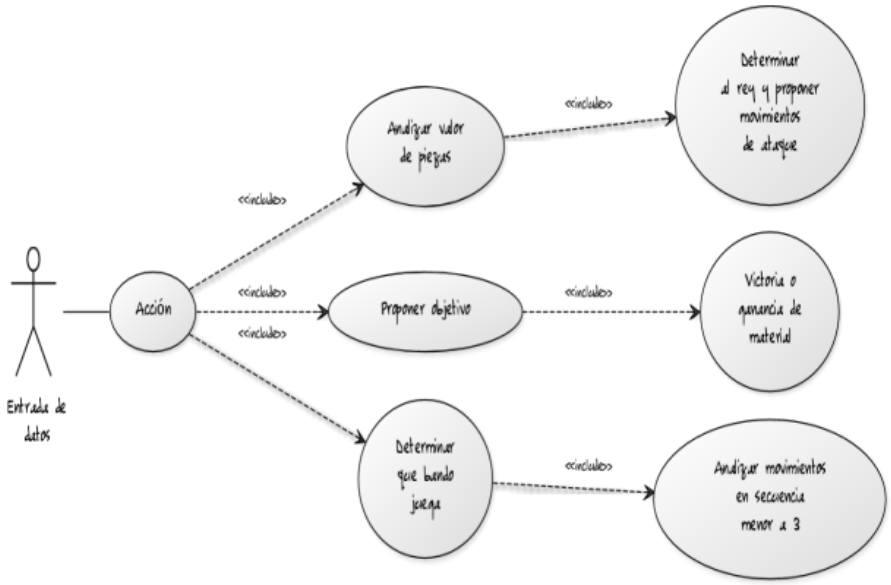


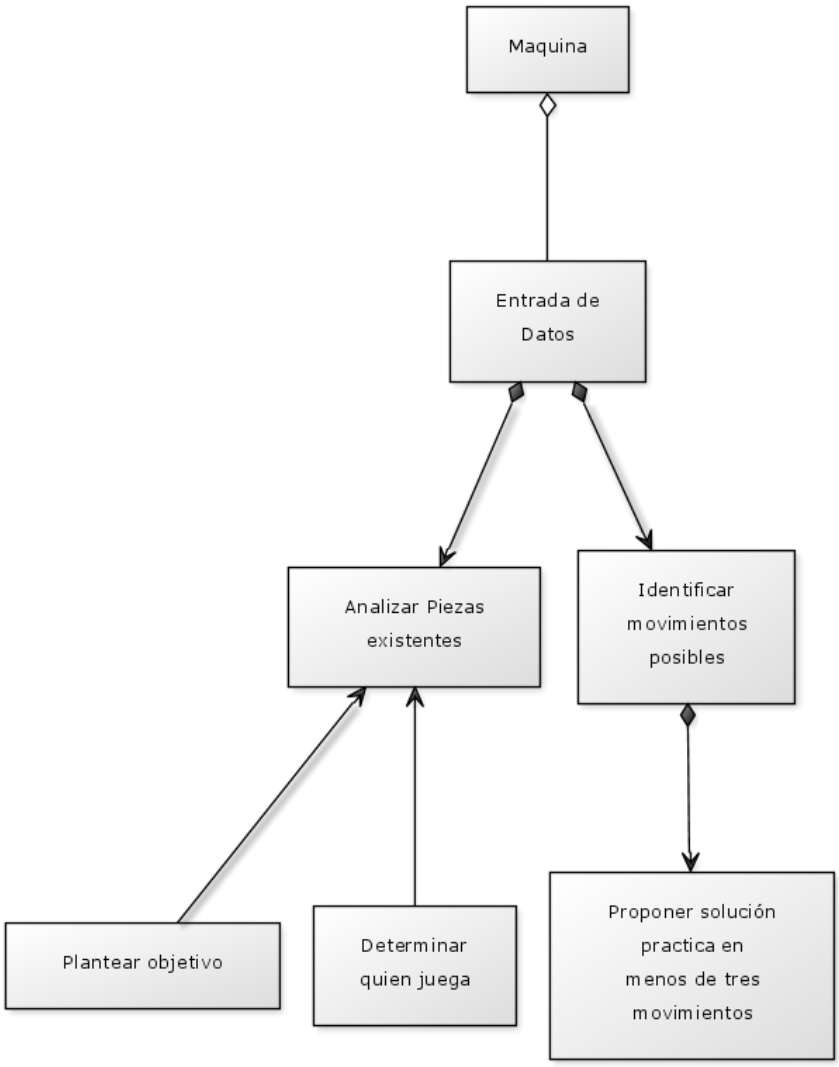
Subsistema de reconocimiento de terreno



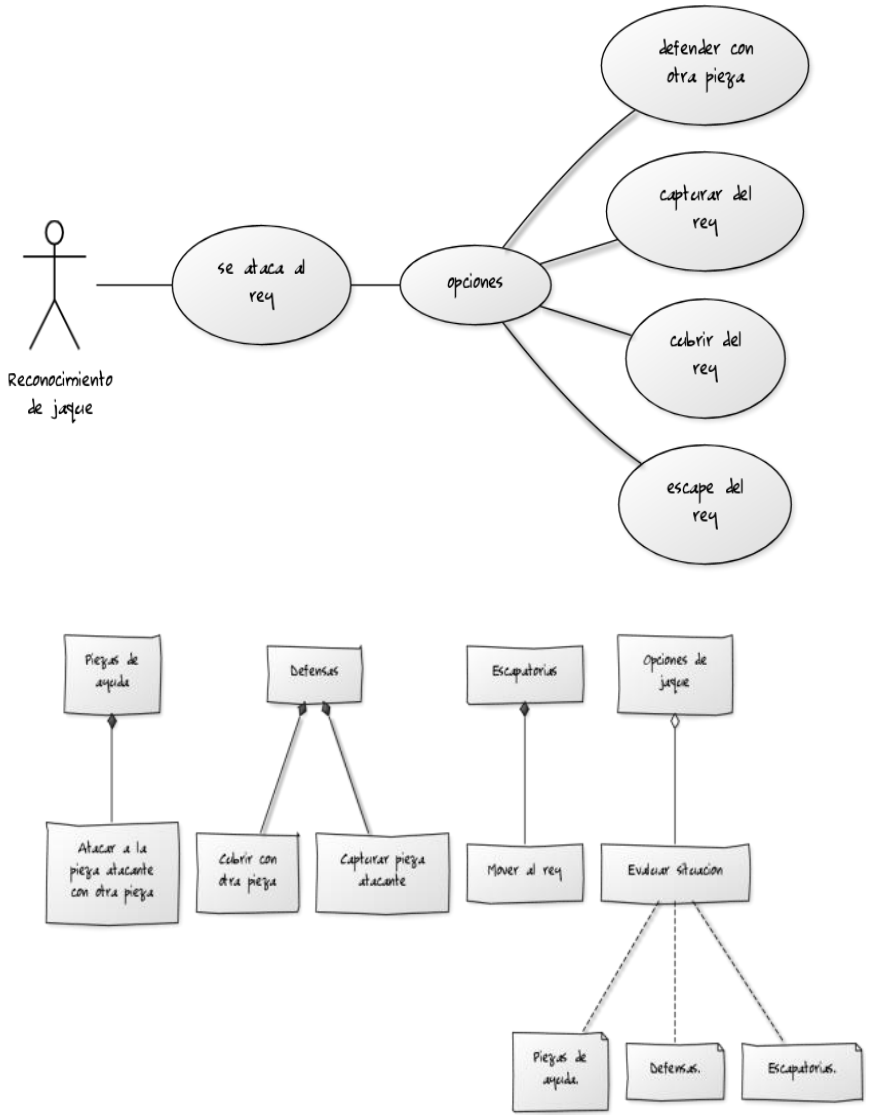
Subsistema de reconocimiento de pieza atacante



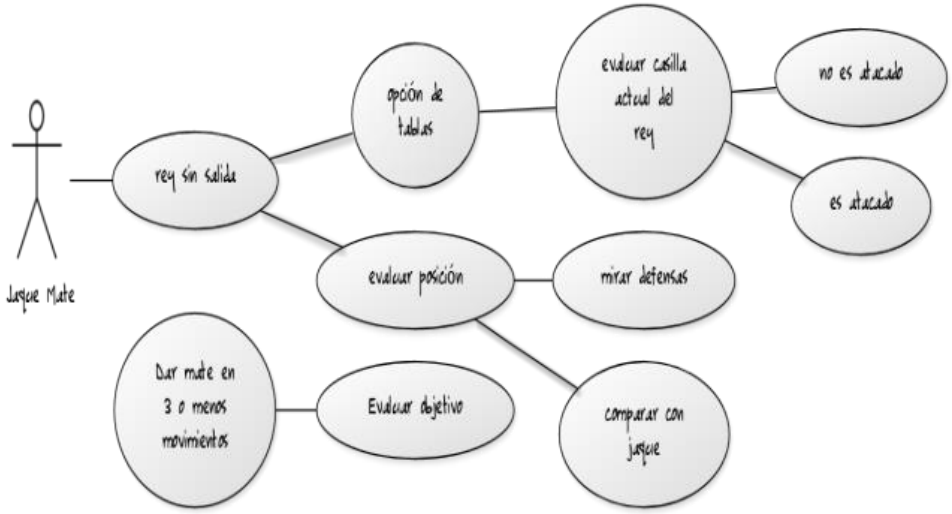

Entrada de datos
Acción
«incluido»
«incluido»
«incluido»
Analizar valor de piezas
Proponer objetivo
Determinar que bando juega
«incluido»
«incluido»
«incluido»
Determinar si rey y proponer movimientos de ataque
Victoria o ganancia de material
Analizar movimientos en secuencia menor a 3

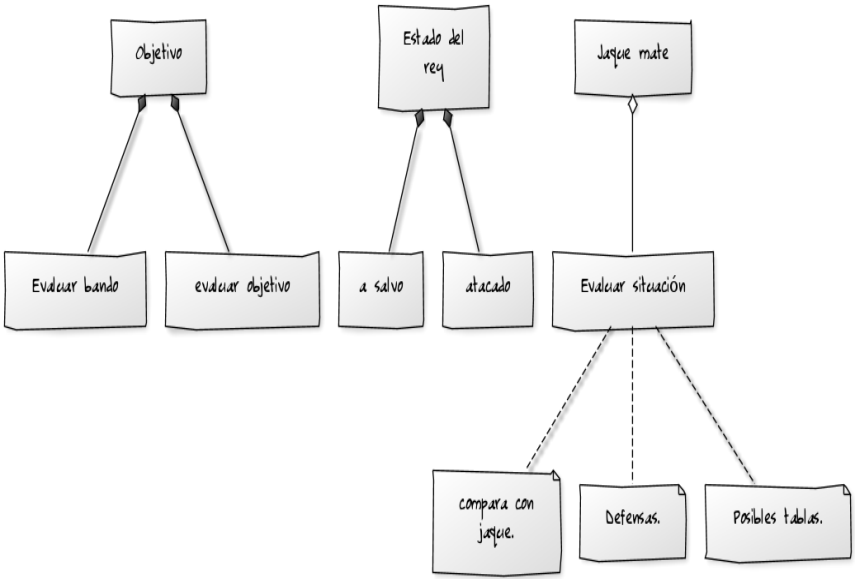

Maquina
Entrada de Datos
Analizar Piezas existentes
Identificar movimientos posibles
Plantear objetivo
Determinar quien juega
Proponer solución practica en menos de tres movimientos

Subsistema de Reconocimiento de Jaque



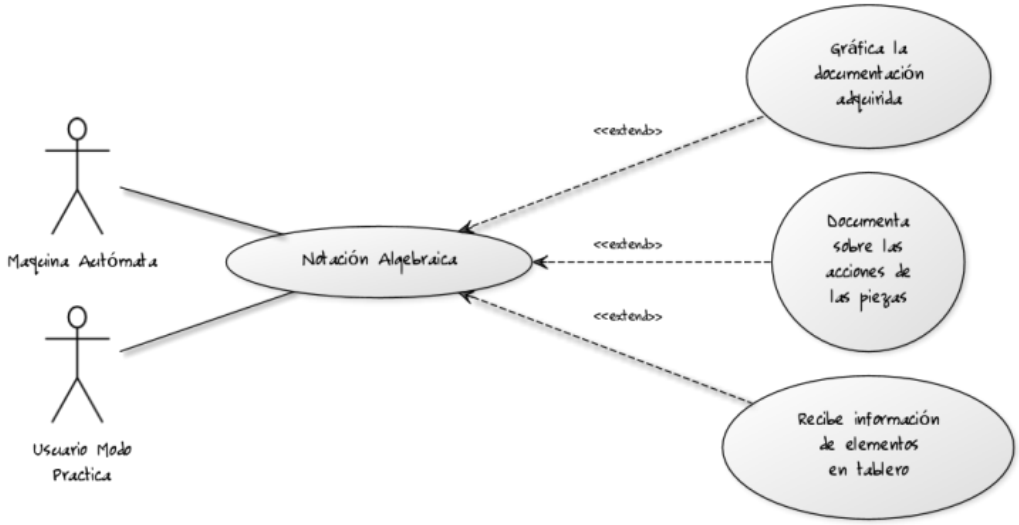
Subsistema de reconocimiento de Jaque Mate

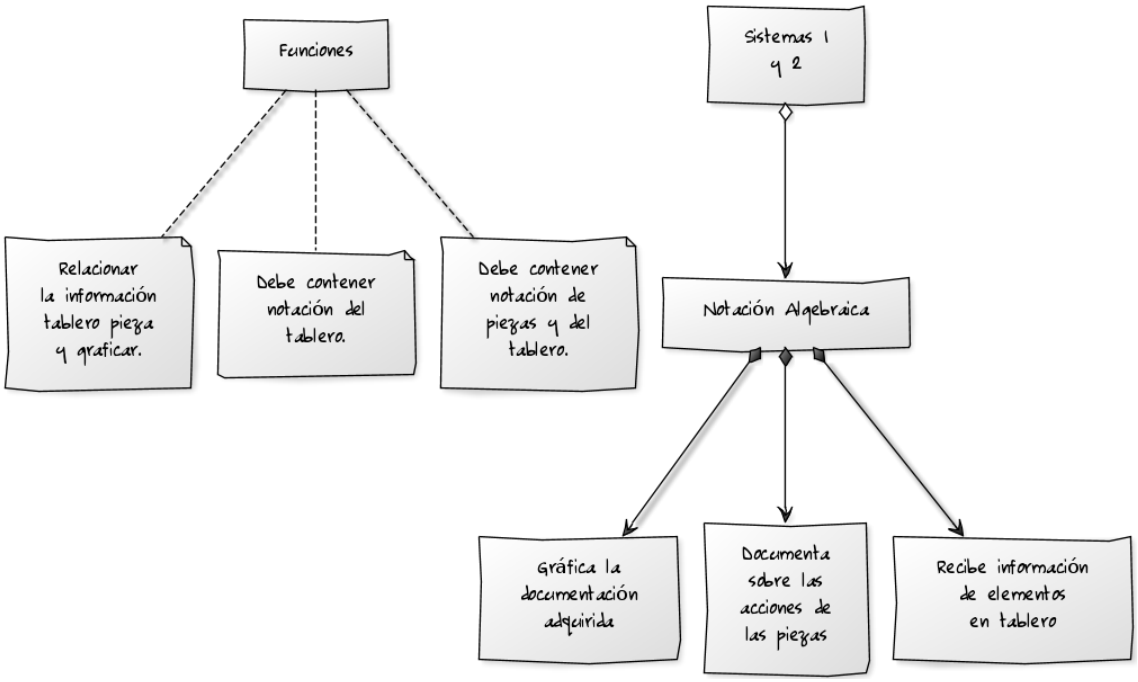




Subsistema de Notación Algebraica

Caso de uso del subsistema interactuando con ambos sistemas





ADAPTACIÓN

El código de Chenard cuenta por debajo con un algoritmo de búsqueda minmax y poda-alpha beta.

Una de las mejores estrategias de búsqueda de mejor respuesta hoy en día es el “Minmax” [14], que utiliza el método de árbol de búsqueda, este técnica denomina a los dos jugadores, uno como Max y al otro como Min, quien inicie el juego será Max, este deberá buscar la repuesta ganadora al problema que se le presenta, para evitar problemas de tiempo en procesamiento se requiere de una nueva herramienta, como es el Poda-Alpha-Betha:

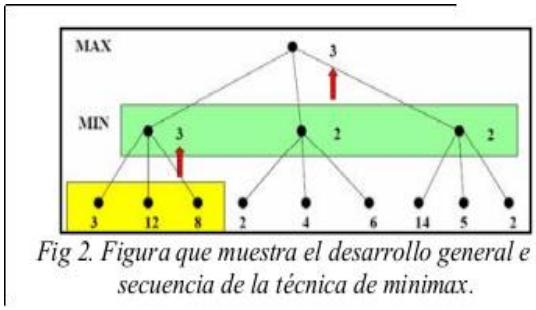


Fig 2. Figura que muestra el desarrollo general e la secuencia de la técnica de minimax.

Extraído de Algoritmos Minmax y Poda Beta.

PODA-ALPHA BETA

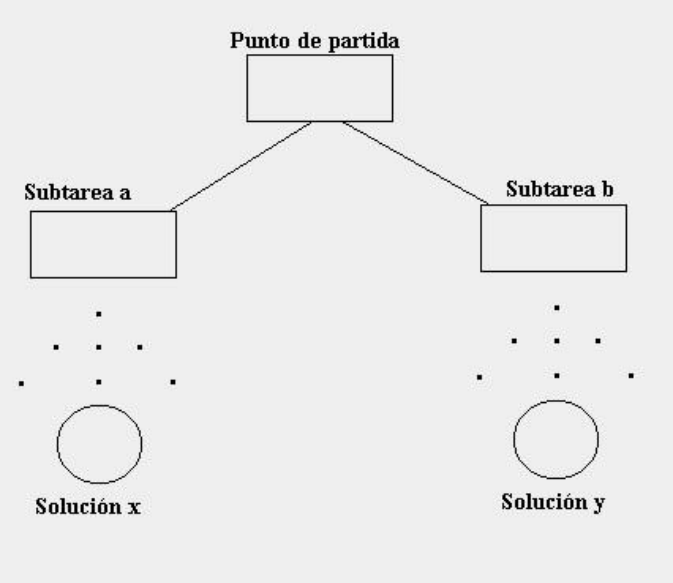
Enfocado a los juegos, conocido por su velocidad de procesamiento sin pérdida de datos, se considera una extensión del algoritmo de búsqueda de Minmax, básicamente lo que busca al analizar un nodo es comprobar si la descendencia de ese nodo será de mejor utilidad que el padre, si no es así, se considera que no será de utilidad para el futuro entonces es descartado, o “podado” que hace referencia a su nombre, esto requiere ingresar al padre a ser evaluado, luego de ello, siguen los hijos del mismo y se decidirá si continuar o “podar” los resultados para continuar con los siguientes nodos.

Extraído de Algoritmos MiniMax y Poda Beta

De este modo la dificultad del juego, o complejidad de búsqueda de la maquina será igual a la profundidad que esta le imponga a la búsqueda de resultados, porque si la maquina calcula previamente 5 o 6 movimientos tendrá una ventaja considerable frente al usuario que en sus limitaciones puede no lograr la misma capacidad de cálculo.

ALGORITMO PARA EL MOVIMIENTO BASICO DE LAS PIEZAS – SOPORTE MATEMATICO COMPLEMENTARIO

Estos algoritmos se asemejan al recorrido en profundidad dentro de un gráfico, siendo cada sub-tarea un nodo del gráfico. El caso es que el grafico no está definido de forma explícita (como lista o matriz de adyacencia), sino de forma implícita, es decir, que se irá creando según avance el recorrido. A menudo el grafico es un árbol, o no contiene ciclos, es decir, al buscar una solución es, en general, imposible llegar a una misma solución (x) partiendo de dos sub-tareas distintas (a) y (b); o de la sub-tarea (a) es imposible llegar a la sub-tarea(b) y viceversa.



Cada vez que nos encontramos con un nuevo movimiento y una nueva decisión que tomar, nos encontramos con la posibilidad de diferentes movimientos posibles, de vez en cuando no necesitaremos encontrar la solución más óptima sino una bastante buena que nos ayude a futuro con el cumplimiento del objetivo.

DESARROLLO IDE

En el siguiente espacio se muestra una versión beta del entorno de la aplicación, el cual se basa en un ejemplo de compilación y constitución de un ajedrez de código abierto realizado por el señor Don Cross, llamado Chenard, recopilación tomada de Ivan Cachicatari[26], el cual propone código abierto de desarrollo, al igual que optimizaciones que han sido tomadas de algoritmos como el del señor Keith Rule, el cual es una clase agregada al proyecto para la generación de Double Buffering, la cual también es de código abierto de desarrollo, CPOL (The Code Project Open License), el cual esta implementado, hasta el punto en donde se obtiene la interfaz del tablero, y el movimiento de las piezas, sin embargo no cuenta con un espacio de carga de ejercicios. La utilidad indispensable para el objetivo puntual del soporte al curso virtual.

Versión inicial y final del código tras la implementación.

Aplicación inicial

Aplicación Final tras adaptación.

CONCLUSIONES

En el presente documento se exponen los avances realizados, tras la obtención del cogido abierto, y su modificación con el fin de dar soporte a la partica de los contenidos expuestos en el curso E-learning para aprender ajedrez en la UMNG, sin embargo cabe la pena resaltar que la modificación al código inicial fue minúscula, ya que este contaba con gran parte del algoritmo necesario para el adecuado uso del simulador, sin embargo se optimizo una pestaña de carga de ejercicios, se tradujo el sistema, se adecuaron las piezas realizadas para el diseño de los OVA, se construyeron los respectivos ejercicios en pgn, y se realizó la compilación y generación del ejecutable para su funcionalidad en sistema operativo Windows.

REFERENCIAS

[1] Definición de ajedrez por María Lucila: http://www.monografias.com/trabajos61/ajedrez/ajedrez.shtml#ixzz2wBRUOiND

[2] Artículos IEEE, porRosaleen Ortiz http://spectrum.ieee.org/slideshow/computing/software/how-computer-chess-changed-programming, How computer chess changed programming.

[3] Articulos IEEE, por Robert Charette, http://spectrum.ieee.org/riskfactor/computing/it/electronic-cheating-scandal-hits-french-chess-federation, Electronic cheating scandal hits French chess federation.

[4] Articulos IEEE, por Philip E. Ross, http://spectrum.ieee.org/computing/software/psyching-out-computer-chess-players, Psyching-out computer chess players.

[5] La Historia del ajedrez, por la Universidad de Chile: http://users.dcc.uchile.cl/~jegger/ajedrez/HistoriaAjedrez.htm, Antecedentes históricos del ajedrez.

[6] El movimiento del caballo,http://www.algoritmia.net/articles.php?id=33, creado el 7 de enero de 2003, por Maria Claudia Espejo.

[7] El árbol de ajedrez y el algoritmo de Minmax,http://users.dcc.uchile.cl/~jegger/memoria/node36.html, creado por la Universidad de Chile.

[8] Chess coach, ajedrez a la orden, por Raul Ocampo, extraidode :http://chesscom-chesscoach.blogspot.com/2009/11/los-algoritmos-en-ajedrez.html, creado en noviembre de 2009.

[9] Ajedrez en solo 5kb, http://p4wn.sourceforge.net/5k/, por Alberto Viches, creado en abril de 2010.

[10] Tutoriales de algoritmos en javascript, minimax, primera lección: tres en raya,por Jorge Rincon, creado el 26 de septiembre de 2013.

[11] Claude E. Shannon, http://www.chess-poster.com/spanish/historia/claude_e_shannon_e.htm, creado el año 2000.

[12] Inteligencia artificial y motores de ajedrez, http://mcyti.izt.uam.mx/proyectos/12I/2013P17.pdf, creado por Dr. John Goddard Close.

[13] Introduccion a la inteligencia artificial, http://dmi.uib.es/~abasolo/intart/1-introduccion.html,

[14] Backtracking y la vuelta del caballo, extraido de www.algoritmia.net/articles.php?id=33, creado el 2001.

BIBLIOGRAFIA GENERAL

[15] Francisco Astudillo Pacheco, Daniel Borrajo Millan, Carmen Tarta Alcalde y Irma Trueba Valle, “Informática Aplicada: Juegos Inteligentes en Microordenadores”, Ediciones Siglo Cultural, España, 1986.

[16] Stuart Russell and Peter Norvig, “Inteligencia Artificial: Un Enfoque Moderno”, Edición Prentice Hall, ISBN 0-13-10385-2, 1996.

[17] Francisco Jose Ribadas Pena, “Busqueda en Espacio de Estados”

[18] “Juegos y MiniMax: Poda Alpha Beta”, [en linea], Formato PDF, 2006,Disponible en: http://arantxa.ii.uam.es/~ia/p4/p4.pdf

[19] José Ignacio Nuñez Varela, “Introducción a la Inteligencia Artificial”, [en línea], Universidad Autónoma de Juan Luis Potosí, Formato PPT, 2007, Disponible en: www.itnuevolaredo.edu.mx

[20] “Inteligencia Artificial: Algoritmo Poda Alpha- Beta”

[21] “Definición de una Poda Alpha-Beta”, [en línea], Disponible en: http://www.lania.mx/~asanchez/IA/minimax3.html

[22] Inteligencia Artificial: Algoritmos MiniMax y Poda Beta,http://jsbsan.blogspot.com/2011/01/inteligencia-artificial-algoritmos.html, por Julio Sanchez, publicado el 30 de enero de 2011.

[23] Problematica de los juegos con inteligencia artificial, extraido de : http://www.monografias.com/trabajos75/problematica-juegos-inteligencia-artificial/problematica-juegos-inteligencia-artificial2.shtml, por Carlos Galindo,.

[24] Rafael Bello, Presentación Power Point, Inteligencia Artificial, Universidad Central de Las Villas, Cuba, Abril 2005.

[25] Rafael Bello, Métodos de Solución de Problemas de la Inteligencia Artificial, Universidad Central de Las Villas, 2007.

[26] Enlace de internet, foro latindeveplopers, http://www.latindevelopers.com/articulos/visualc/chess-4.php,pagina que sirvió como guía para la realización del ajedrez en su versión beta. Ivan Cachicatari

[27] Don Cross, Chenard for Windows, http://cosinekitty.com/chenard/.