

**Universidad Peruana de Ciencias Aplicadas**

**Ciclo 2020-2**

**Complejidad Algorítmica**

**Ingeniería de Sistemas de Información y Software**

**PROFESOR DEL CURSO:**

Luis Martin Canaval Sanchez

**Integrantes:**

· Costa Morales, Juan Josemaria (U201822001)

# Introducción:

En el curso de Complejidad algorítmica de las carreras de Ingeniería de software y Ciencias de la computación tenemos el proyecto de Quoridor, un juego de mesa que se creó en el año 1997 en Francia. Este juego consta de una cuadrícula de 9x9, cada jugador controla un peón, que debe comenzar en alguno de los bordes del tablero. Los peones deben ubicarse en el centro de la fila y enfrentados el uno del otro. Antes de comenzar, se dividen los 20 bloques en partes iguales (así, cada jugador tiene 10 si se juega de a 2, y 5 si se hace de a cuatro). Los bloques deben ubicarse en la ranura ubicada entre cada casillero. Por su ancho, cada bloque corta el camino de dos filas o columnas (según si se ubica en horizontal o vertical).  
Nuestro proyecto consiste en implementar este juego haciendo uso de los conocimientos adquiridos durante esta parte del semestre, así como de enfoques y paradigmas de programación y tomando conciencia de la importancia de estos en la industria.

# Estado del Arte:

Durante el proyecto implementaremos una inteligencia artificial, la cual ha ido evolucionando en muchos aspectos en los últimos años desde que se escribieron los primeros programas de ajedrez, ha habido un progreso constante en el nivel de juego hasta el punto en que los sistemas actuales pueden desafiar a los jugadores humanos de primera clase. Recientemente, grupos académicos han creado competiciones para evaluar y comparar métodos de IA, como Ms Pac-Man versus Ghost Team y Student StarCraft AI Tournament. Sin embargo el uso de inteligencia artificial hoy en día es un estándar para la mayoría de videojuegos, tanto en juego de disparos en primera persona como en juegos de mesa y de estrategia como los mobas.

Un gran ejemplo de que la inteligencia artificial y la industria de los videojuegos están de la mano, es la IA The ANGELINA desarrollada por Michael Cook en 2011, Esta I.A.es capaz de diseñar videojuegos desde cero de una manera que resulta sorprendentemente simple, basta con darle información como fotografías o texto para que pueda crear juegos completamente nuevos, con un lenguaje de programación única y niveles y reglas totalmente originales.

Otra tecnología que usaremos durante el proyecto son los algoritmos de búsqueda en grafos. La cual se remonta al siglo XVII con el problema de los puentes de Konigsberg, con el pasar de los años su uso ha ido evolucionando, siento en la actualidad la principal fuente para los algoritmos de búsqueda.

# Metodología:

Nuestro proyecto se enfocará en aplicar los algoritmos trabajados en clase para la recreación del juego Quoridor. Debido al enfoque del juego se trabajará principalmente la búsqueda y recorrido del camino óptimo asegurando de esta forma una alta probabilidad de victoria. Adicionalmente, se tomarán en cuenta la posibilidad de que el camino óptimo cambie a medida que se desarrolle la partida.

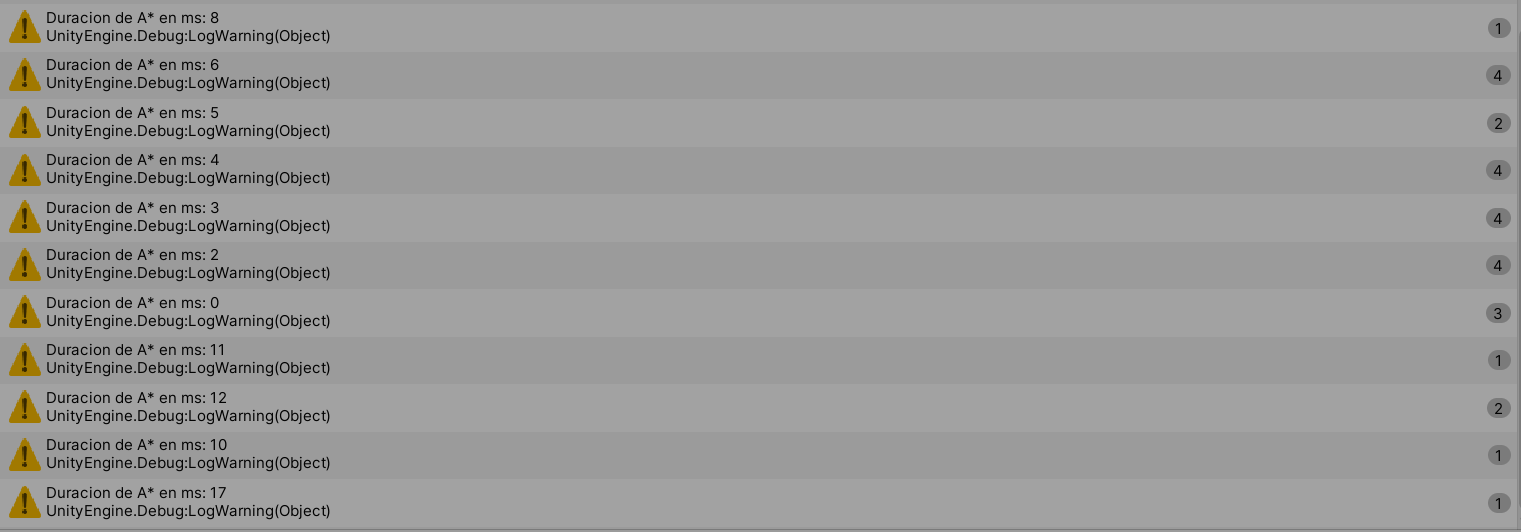
Nuestra IA determinará el movimiento siguiente en base al algoritmo A\*(A estrella) el cual le permite encontrar el camino óptimo a su objetivo. Además, debido a que se tienen nueve posiciones para ganar la partida la IA utilizará divide y vencerás para determinar su mejor objetivo. Se manejará mediante una máquina de estados la cual tendrá tres estados específicos:

1. Waiting: Definirá el momento de juego del jugador, de esta forma el concepto de turnos será manejado por la misma IA.
2. Calculating: Este estado determina el camino óptimo para que la IA llegue de un lado a otro
3. Moving: Con este estado podremos ejecutar un movimiento en base al camino determinado en el estado dos, al finalizar, se cambiará al estado uno para que el jugador pueda realizar su movimiento y el ciclo se repita.

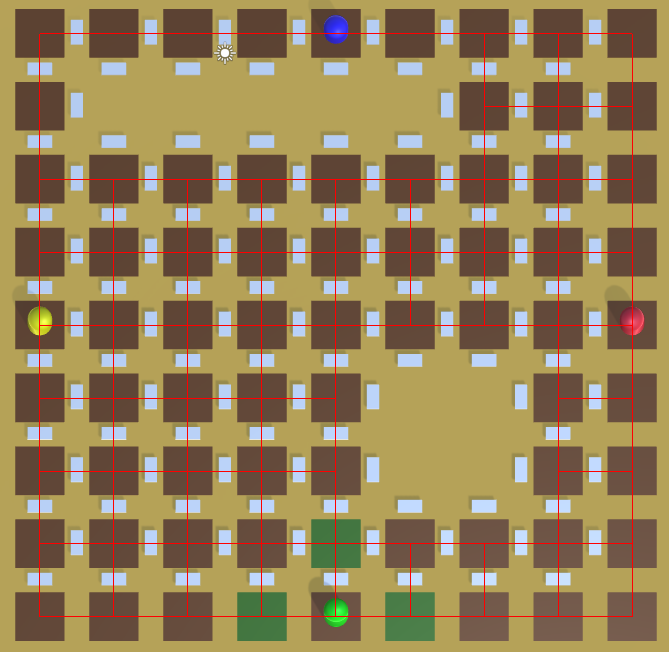
Por parte del jugador, se podrán efectuar los movimientos básicos del juego además de la opción de alterar el tablero mediante unas barreras que bloquean las aristas y prohíben el paso entre dos vértices.

El algoritmo A\* consiste en asignar dos valores a los nodos cuya suma determina el orden en el que son explorados, estos valores son la distancia recorrida del punto inicial hacia el nodo y el otro valor es la distancia entre el nodo y el punto final. A continuación, el algoritmo suma los valores para obtener el costo de cada nodo lo cual emplea para recorrer los nodos adyacentes a los nodos recorridos utilizando el de menor valor primero.

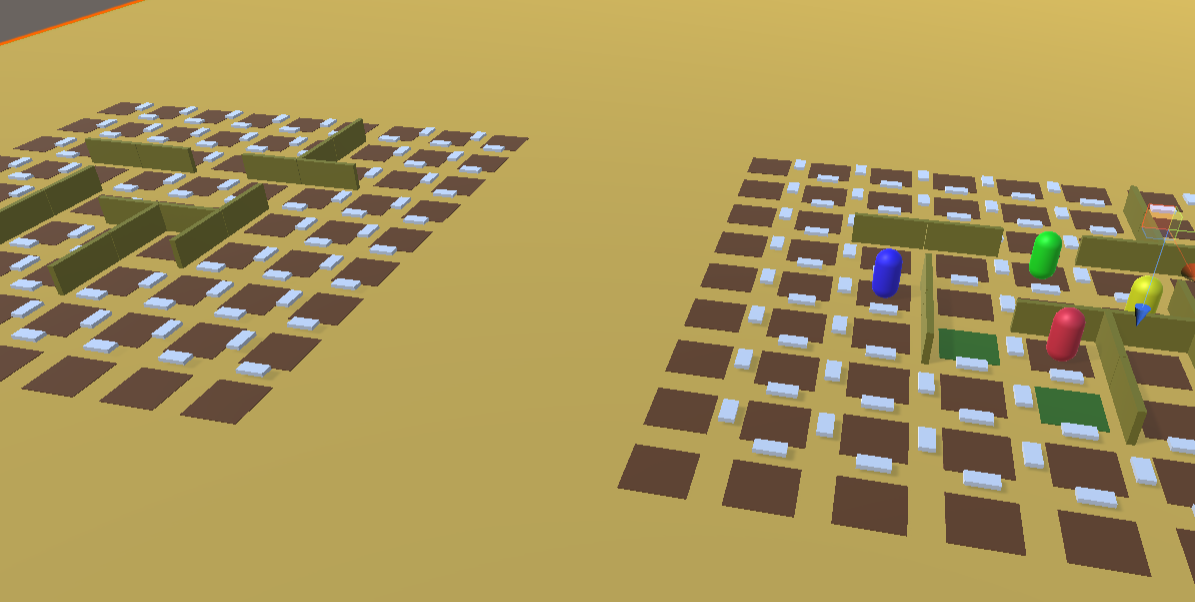
Luego de varias pruebas calculando el tiempo de ejecución del algoritmo se puede observar que a medida que la ficha se va viendo rodeada de más muros y curvas dentro de su recorrido mayor es el tiempo de ejecución. Esto se debe a que si la ficha se encuentra cerca de alguno de los puntos objetivo el número de nodos adyacentes se reduce considerablemente y el algoritmo termina en un tiempo reducido.



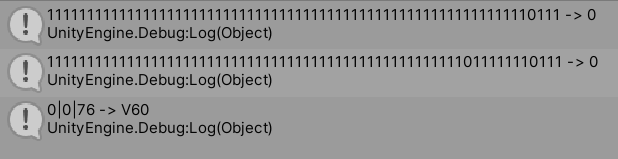
El tamaño del tablero también influye en este corto periodo de ejecución debido a que el algoritmo A\* cuenta con una complejidad algorítmica exponencial O(n^2) y el tablero cuenta con una dimensión de 9x9. En caso se quiera poner a prueba tamaños más grandes o grafos más desordenados es posible realizarlo debido al enfoque de comportamiento emergente. Ya que el grafo se genera mediante la interacción de entidades solo hace falta clonar dichas entidades y renombrarlas para que se pueda generar grafos de diferentes tamaños y formas.



Una de las mecánicas del juego es la habilidad de colocar muros los cuales restringen la movilidad de todas las fichas y permiten a los jugadores retrasar el avance de sus oponentes. Para que la IA determine dónde colocar un muro se planteo el uso de un algoritmo greedy el cual recorre las dos listas de posibles ubicaciones para los muros horizontales y verticales. Luego de seleccionar un muro se utiliza una copia del tablero que refleja las acciones realizadas en el tablero original y nos permite simular los diferentes escenarios hasta encontrar una posición que alargue la trayectoria del oponente con más posibilidades de ganar.



Luego de haber determinado el muro que se debe activar utilizamos el enfoque de programación dinámica y guardamos el estado de la lista de posibles posiciones para el muro en un diccionario el cual traduce el estado de los muros a un número binario para servir como key y se le asigna un numero que va incrementando conforme se encuentre más estados para la lista de posibles muros. Una vez realizada esta operación se combina el valor de ambas llaves junto con la posición del oponente para generar una tercera llave que guardará como valor el muro óptimo para esa situación. De esta manera el programa realizará la acción nuevamente a lo largo del tiempo en el que se use el programa. Para facilitar eso se implemento un manager con enfoque singleton que permite guardar la información recopilada a lo largo de varias partidas.



# 

# Bibliografía

Firmansyah, Eka & Masruroh, Siti & Fahrianto, Feri. (2016). Comparative Analysis of A\* and Basic Theta\* Algorithm in Android-Based Pathfinding Games.

Ian Millington and John Funge. 2009. Artificial Intelligence for Games, Second Edition (2nd. ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

F. Lu, K. Yamamoto, L. H. Nomura, S. Mizuno, Y. Lee and R. Thawonmas, "Fighting game artificial intelligence competition platform," 2013 IEEE 2nd Global Conferenc*e on Consumer Electronics (GCCE)*, Tokyo, 2013,