

# **Juego de damas en c++**

**Abarca Arias Felipe**

**Quiñelen Villar Jaime**

**Ortega Bustamante Ignacio**

*Universidad Tecnológica Metropolitana de Chile*

*Facultad de Ingeniería*

## **Abstrac**

*¿Ha jugado alguna vez damas chilenas contra un computador? , de ser así es seguro que más de alguna vez se ha preguntado cómo internamente el pc decide cual es la mejor jugada. Pues bien en este paper se hará un análisis interno de un programa escrito en c++ para jugar damas. ¿Cómo evitar perder fichas? , ¿Cómo comer la mayor cantidad de fichas al enemigo?, ¿Podrá la maquina superar al ser humano?, estas y otras preguntas serán resultas a continuación.*

## **1.- Introducción**

En el siguiente paper, el lector se insertara en el mundo de la programación, se le mostraran las estrategias utilizadas por los programadores para desarrollar un algoritmo en lenguaje c++ que sea difícil de ganar por el humano, se discutirá cual fue la complejidad algorítmica utilizada, mecanismo de decisión que llevaron a los desarrolladores inclinarse por cierta solución o no otra, estudio de funciones utilizadas, rendimiento del juego, etc.

## **2.- Contenido principal**

### **2.1-Como y porque se desarrolló el juego**

El proyecto fue desarrollado en la universidad tecnológica metropolitana de chile, para la asignatura ingeniería de software, dicho juego fue programado en el lenguaje c++, sistema operativo linux y el IDE utilizado fue NetBeans 8.02. El juego consta de más de 5000 líneas de código, además se quiso abarcar un área mayor de personas que lo puedan utilizar y se le doto de 2 idiomas, español e inglés, además de constar con un menú de juego, donde se pueden obtener las instrucciones de juego, los datos de los desarrolladores, etc.

### **2.2-rendimiento de la aplicación**

El software posee un rendimiento perfecto, cada jugada de la computadora carga en menos de un segundo, de hecho, en decisión unánime entre los programadores, se decidió crear una función tiempo( ), que tarda 3 segundos en cargar, para simular el pensamiento de la máquina, y así no mostrar tan de prisa la jugada escogida por el algoritmo.

### **2.3.- complejidad algorítmica**

Luego de estudiar el tema y debatir con los demás desarrolladores, se tomó la decisión de utilizar un algoritmo “defensivo”, consiguiendo que las fichas no ataquen, esperando la jugada del rival, y así minimizar al máximo los riesgos de perder fichas.

Además se investigó cuáles eran las mejores estrategias defensivas, utilizadas por los jugadores en la vida real, para así programarlas en nuestro juego, dichas estrategias se enumeraron de 1 al 7, como por ejemplo, **estrategiaCinco( )**, y cada una de ellas tiene funcionalidades específicas para el desarrollo del juego.

El tablero al ser una matriz, llamado `tablero [ ] [ ]`, cada casilla es representada por un par de índices, en este caso *i* para las coordenadas horizontales y *j* para las coordenadas verticales, consiguiendo el par ordenado  $\rightarrow (i, j)$ . ver imagen 1

$i-1, j-1$		$i-1, j+1$
	$i, j$	
$i+1, j-1$		$i+1, j+1$

**Imagen 1**

En el juego de las damas solo existen 4 movimientos:

- 1) Arriba izquierda ( $i-1, j-1$ )
- 2) Arriba derecha ( $i-1, j+1$ )
- 3) Abajo izquierda ( $i+1, j-1$ )
- 4) Abajo derecha ( $i+1, j+1$ )

Todos estos movimientos en torno a una ficha central, cuya posición es (*i, j*).

Absolutamente todo nuestro algoritmo trabaja bajo esta regla de posiciones, ya sea comer, hacer la sopladita, moverse, coronar dama, validaciones, etc.

En la siguiente imagen ilustraremos como una pieza puede comer a la otra, entendiendo esto, se podrán entender las demás funciones, ya que todas trabajan de igual forma. Ver imagen 2

		O
	X	

**Imagen 2**

Siguiendo el patrón descrito en la imagen 1, en la casilla (*i, j*), se encuentra una ficha X, y en la casilla ( $i-1, j+1$ ) se encuentra una ficha rival O, además se cumple la condición de que en la casilla ( $i+1, j-1$ ) hay un espacio libre para poder comer, algorítmicamente se vería así:

```

If ( tablero [ i ] [ j ] == 'X' &&
    tablero [ i-1 ] [ j+1 ] == 'O' &&
    tablero [ i+1 ] [ j-1 ] == ' ' )
{
    tablero [ i ] [ j ] = ' ';
    tablero [ i-1 ] [ j+1 ] = ' ';
    tablero [ i+1 ] [ j-1 ] = 'O'
}

```

En la primera instrucción, borramos la X, colocando un espacio, en la segunda instrucción borramos la O, ya que debe moverse a la otra casilla, y en la tercera instrucción colocamos la ficha O en su nueva posición. Ver imagen 3

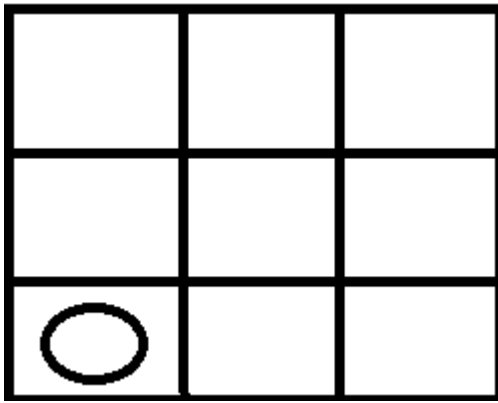


Imagen 3

### 2.3.1- Análisis de funciones utilizadas como estrategias

La función estrategiaUno( ), lo que hace es adelantar toda la fila 3 a la fila 4, y la fila 2 a la fila 3 lo que se pretende con estas dos funciones es formar un zic zag con las fichas, de modo que si una ficha rival nos come una ficha, atrás haya otra para comer dicha ficha rival. Ver imagen 4

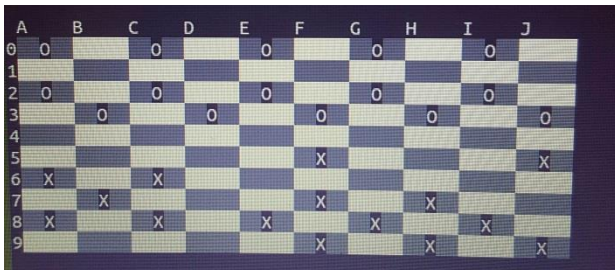


Imagen 4

La función estrategiaDos( ), hace lo mismo pero con las demás filas, la estrategia es ir adelantando fila por fila, JAMAS, enviar fichas solas al ataque, y siempre formando el zic zag de protección, para que haya una ficha atrás de la otra y se puedan proteger.

Si ya no se pueden mover más fichas, porque falta de espacios o por haber fichas enemigas cerca, la estrategiaTres( ), lo que hace es mover la última fila, dejamos la estrategiaTres( ) para el final ya que mover la última fila es lo último que se debe hacer, para evitar que el otro

jugador forme una dama.

El criterio de movimiento de las estrategias mencionadas anteriormente, es ir moviéndose fila por fila, pero SIEMPRE, teniendo en cuenta que no haya ninguna ficha enemiga en la casilla subsiguiente, para no ser comido, dichos movimientos se encuentran validados, para evitar errores como, casillas ocupadas, movimientos inválidos, salirse del tablero, etc.

La estrategiaCuatro( ), la llamamos estrategia critica, si ya no hay más movimientos, por falta de espacios, fichas enemigas cercas o si o si me van a comer, muevo la primera ficha O que me encuentre mi doble ciclo for( ) para “sacrificarla”, talvez se pierda una ficha, pero producto de las estrategias anteriores siempre abra una atrás para comer a dicha ficha rival.

La estrategiaCinco( ), lo que hace es detectar si hay fichas amigas a punto de convertirse en dama, de ser así, las mueve a la última fila, para transformarla en dama. Ver imagen 5

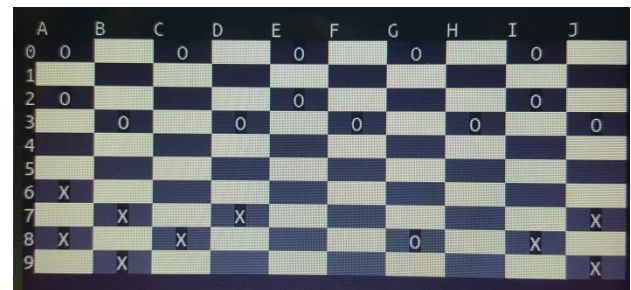


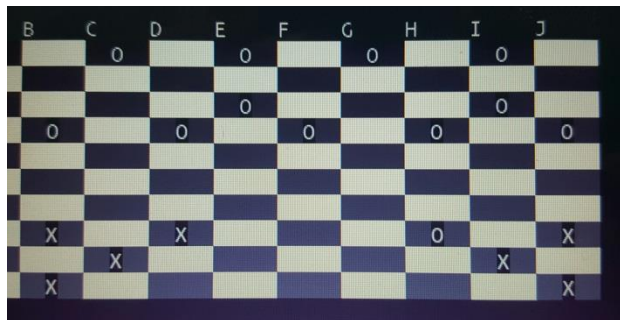
Imagen 5

Como se puede observar en la imagen 5, la ficha G-8, está a punto de convertirse en dama, ahí entra la estrategiaCinco( ), que la moverá inmediatamente a la fila 9, privilegiando a cualquier otra jugada, pues lo más importante es formar una dama lo más rápido posible.

La estrategiaSiete( ), lo que hace es adelantar la la fila 1, a la fila 2, la dejamos para el final, ya que siempre hay que tratar de dejar fichas al fondo, para evitar cualquier formación de dama del rival.

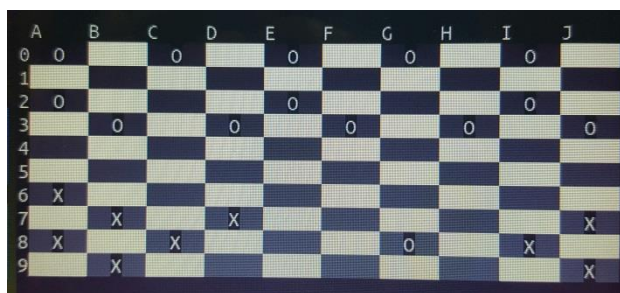
Arrancar( ), lo que hace esta función es hacer

que una ficha arranque si detecta que hay una ficha rival en la casilla siguiente.



**Imagen 6**

Se puede observar en la imagen 6 , que en la casilla H-7 hay una ficha que está a punto de ser comida, pues aquí se activa la función arrancar( ), para que la ficha arranque y no sea comida. Ver imagen 7



**Imagen 7**

Se puede observar en la imagen 7, como la ficha arranco a G-8 para evitar ser comida.

### 2.3.2- Análisis de algoritmos de validación

La función cpu( ), es la utilizada por la máquina para mover fichas, en ella se encuentran todas las estrategias descritas anteriormente, y posee todas las validaciones de movimientos, como por ejemplo que una casilla ya este ocupada, que haya una ficha rival en la casilla siguiente, por lo tanto mover para el otro lado.

Función sopladita( ) , se activa una vez terminado el turno del jugador humano esta función lo que hace es recorrer el tablero (matriz) y detectar si al jugador humano se le

paso una oportunidad para comer una ficha. Ver imagen 8



**Imagen 8**

Se puede observar en la imagen 8 que I-8 tiene la opción de comer, si el jugador humano no come, se activara la sopladita

Función comerFicha( ), función que se activa antes de la función cpu (ya una vez en el turno de la maquina) , lo que hace esta función es recorrer el tablero y buscar si hay alguna opción de comer, de ser así, lo hace y finaliza el turno. Se puede apreciar en la imagen 2 y 3.

La función preguntar( ), es la que le pregunta al usuario cual ficha desea mover y hacia donde, en esta función se encuentran todas las validaciones de entrada de teclado, por ejemplo si se necesita teclear un número y se ingresa un letra, que muestre el mensaje de error correspondiente, o si se introduce un carácter accidentalmente como “ , ' ¿ # ” , muestra el mensaje de error.

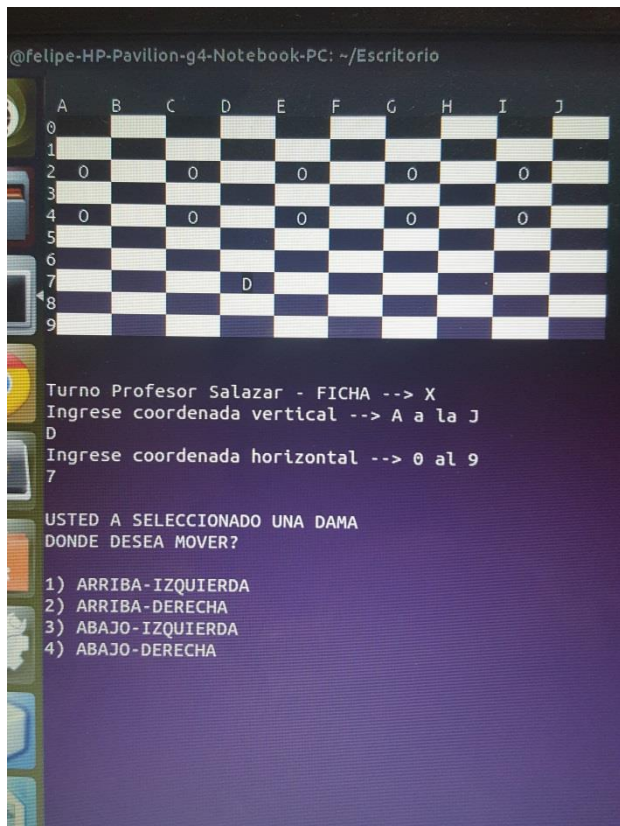
Al llegar al otro extremo del tablero, la ficha peón, se convierte en dama, ¿que significa eso? que puede moverse libremente por una diagonal, respecto a su posición, cumpliendo ciertas condiciones, como por ejemplo no puede saltar a fichas aliadas (ya sea peones o dama), tampoco si hay dos fichas juntas, sin un espacio para comer.

En el codigo a esta función la llamamos moverDamaHumano( ), y moverDamaMaquina( ) , la primera para el jugador humano y la segunda para la máquina.

Dentro de la función MoverDamaHumano( ) utilizamos subsunciones, para validar los



movimientos que puede hacer la dama, la primera función es `perteneceLaDiagonal1()`. Como dijimos anteriormente, en el juego de damas, solo son permitidos 4 movimientos, y la única ficha que puede realizar esos 4 movimientos es la dama, lo que hace la función `perteneceLaDiagonal1()`, es verificar si la posición seleccionada por el usuario pertenece realmente a la diagonal en la cual se puede mover la dama, si nos retorna `false`, es porque se ha producido un error, y nos mostrara el mensaje de error, en caso contrario, nos pedirá ingresar la nueva posición de la dama. Esta función consta de 3 variantes más, `perteneceLaDiagonal2()`, `perteneceLaDiagonal3()`, `perteneceLaDiagonal4()`, las únicas diferencias son que cada función analiza una diagonal, ya sea la diagonal 1, 2, 3 o 4.



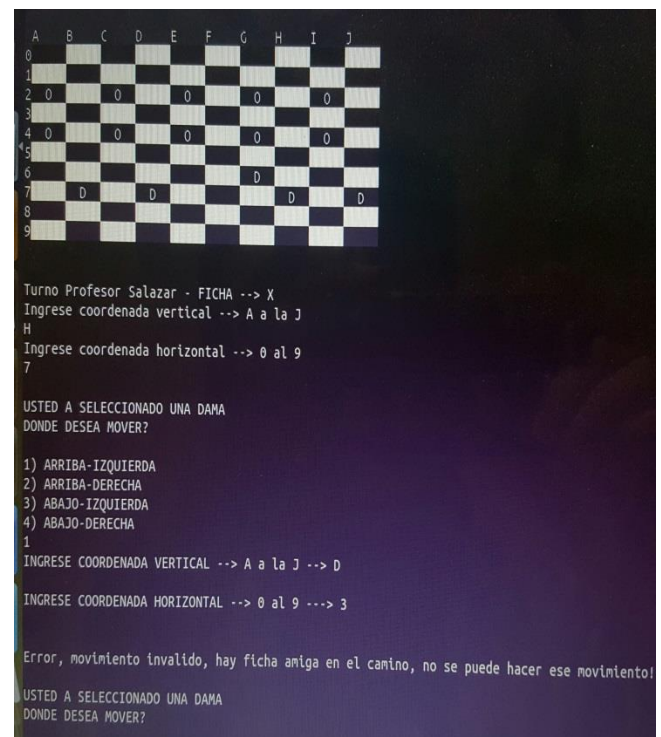
**Imagen 9**

En el siguiente ejemplo, como se puede observar hemos seleccionado la ficha D-7, y

deseamos moverla con la opción 3 a la casilla D-0, esto nos producirá un error, debido a que la dama solo se puede mover en una diagonal, ¿Cómo conseguimos que nos muestre el mensaje de error?

En tablero `[i][j]='D'`, si presionamos la opción 3 significa que nos queremos mover a la posición `tablero[i-1][j+1]`, por lo tanto vamos recorriendo la diagonal disminuyendo el índice `i` y aumentando el índice `j` `tablero[i-2][j+2]`, `tablero[i-3][j+3]`, `tablero[i-n][j+n]`... si recorremos toda la diagonal y no encontramos la posición D-0 que vendría siendo `tablero[0][3]`, nos retornara el mensaje de error.

La siguiente función es `hayFichaAmiga()`, como dijimos anteriormente, la dama no puede pasar sobre una ficha aliada, lo que hace esta función, es analizar si hay fichas aliadas en el camino, de ser así retorna `true`, y nos envía el mensaje de error, de ser `false`, se ejecuta el movimiento de la dama. Esta función también tiene las variantes 1, 2, 3 y 4.



**Imagen 10**

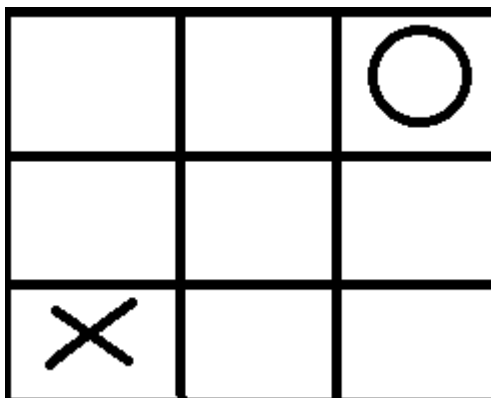
sePuedeComer( ) es la función que nos retornar true para comer una ficha enemiga, las condiciones que se deben cumplir es que perteneceLaDiagonal() nos retorne true, que no hayan dos fichas juntas y que haya un espacio al otro lado.

SePuedeMover( ), es la función que me valida el movimiento de la dama, para que retorne true, debe estar toda la diagonal vacía, ya que esta función lo único que hace es moverme la dama, sin realizar el acto de comer, para esto perteneceDiagonal( ) debe retornar true.

### 3.- Mecanismo de toma de decisiones

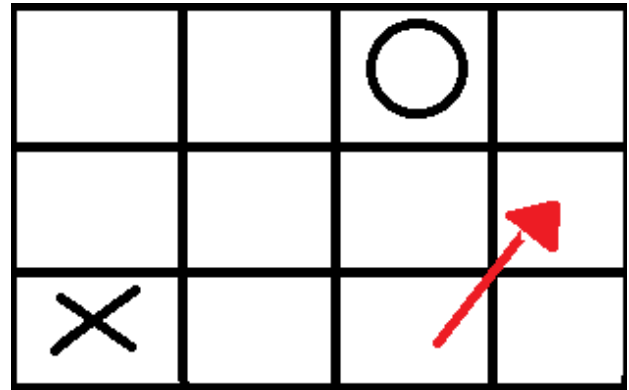
¿Cómo nuestro algoritmo toma la mejor decisión?

El mecanismo que utilizamos es jamás mover una ficha si hay una ficha enemiga en la casilla subsiguiente.



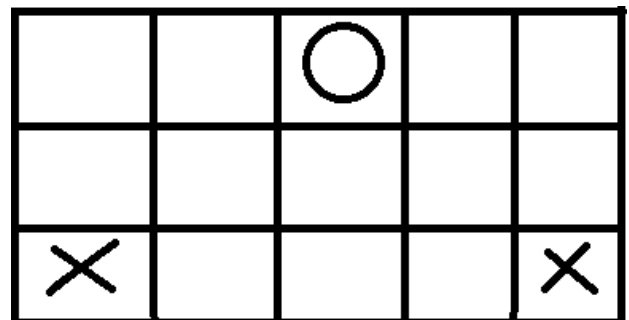
**Imagen 11**

Mirando la imagen 11, si en `tablero[i][j]='O'` y en `tablero[i+2][j-2]='X'` y en `tablero[i+1][j-1]=' '` (hay un espacio disponible), la ficha O no es candidata a realizar el movimiento 3 que es abajo - Izquierda (`tablero[i+1][j-1]=' '`). Debido a que sería una ficha perdida.. Por lo tanto, realizaría el movimiento 4 (abajo-derecha). Ver imagen 12



**Imagen 12**


Otro caso que podría ocurrir es que la ficha no tenga opciones de moverse, como se muestra en la imagen 13.



**Imagen 13**

En este caso, nuestro `for( )` que recorre el tablero no seleccionaría dicha ficha, ya que no es candidata a moverse para ningún lado.

Lo mismo ocurre con el caso de la dama, solo que por ser dama, puede elegir donde caer, y debe tener cuidado no ser comida. Ver imagen 14

	$i-1, j-1$		$i-1, j+1$	
		$i, j$		
	$i+1, j-1$			
				D

**Imagen 14**

Las damas en nuestro juego se representan con una letra D, se puede observar que la dama D tiene la posibilidad de comer, si cae en la casilla (i,j), debe tener cuidado y analizar que no hayan fichas rivales en los extremos. Algorítmicamente se vería así

```
If (tablero [i-1][j-1]!=O &&
tablero [i+1][j-1]!=O &&
tablero [i-1][j+1]!=O )
{
tablero [i][j]=D; }
```

Por último la función moverDamaMaquina(), es la que me mueve la dama de la computadora, esta función hace lo mismo que moverDamaHumano(), con la diferencia que aquí el algoritmo determina en que diagonal mover.

Dentro de esta función, hay una sub función llamada, sePuedeComerMaquina(), lo que hace es recorrer la diagonal de la dama, y si hay una opción de comer(retorna true), desplaza la Dama a la nueva posición.

#### 4.- Explicación de alternativas estudiadas

Se estuvieron estudiando varias alternativas de implementación del algoritmo, una de ellas era realizarla con un árbol de búsqueda, pero luego de pruebas y estudio nos dimos cuenta que no nos entregaba una solución rápida, tardaba mucho en ejecutarse, y era necesaria una función de poda ( ) de ramas del árbol,

para disminuir el tiempo de búsqueda, otra razón por no elegir esa opción fue que el grupo de desarrolladores no contaba con los conocimientos necesarios de árboles de búsqueda, e implementarla requería tiempo en capacitación,

y estudio para aprender y manejar bien los árboles, llevar a cabo esta opción ponía en riesgo no cumplir con los plazos de entrega del proyecto.

Luego de debatir y conversar con el grupo de desarrolladores, se optó por la opción de manejar el tablero como una matriz, accediendo a cada casilla con los sub índices i , j y aplicando todas las funciones descritas anteriormente.

Además nos permitía programar nuestras propias jugadas, manejar a nuestra voluntad el movimiento de fichas, donde iban a caer, en qué dirección se van a mover, etc.

#### 5.- Conclusión

Como grupo de trabajo este proyecto fue un verdadero desafío, en los 5 años de universidad jamás habíamos implementado algoritmos de inteligencia artificial, que nos simularan el pensamiento de cuál podría ser la mejor jugada, en un comienzo fue difícil, se tuvo que investigar mucho, repasar el olvidado lenguaje c++ , aprender las reglas del juego, investigar las mejores jugadas, aprender estrategias, etc.

Al momento de llevar lo estudiado a la programación del juego en sí, más de un dolor de cabeza nos dio, eran demasiados detalles que no se podían dejar pasar, validaciones de movimientos, fichas, tener cuidado a no perder peones, formar lo más rápido posible una dama y por sobre todo, abarcar todo el reglamento del juego en sí.

Afortunadamente este grupo de desarrolladores logro sacar el proyecto adelante, entregándolo en el plazo estipulado y logrando crear un algoritmo de IA que no es fácil de vencer.

## 6.- Bibliografías

Nos basamos en un algoritmo encontrado en youtube, del cual contaba de muchos errores, las fichas se movían en modo random lo que conllevaba a que en ocasiones el programa se quedara pegado , no poseía la opción de la soplada, el tablero era de 8x8, no le daba la opción al usuario de elegir su jugada, las damas no “volaban” ,etc.

Vimos en este algoritmo una buena base para modificarlo y adaptarlo a los requerimientos de nuestro proyecto.

<https://www.youtube.com/watch?v=8qBVP6nEXsw>

Las estrategias que programamos en nuestro proyecto las obtuvimos de:

<https://www.youtube.com/watch?v=jA-zevc2fao>

<http://es.wikihow.com/ganar-en-damas>

[http://juegos.about.com/od/juegos\\_clasicos/fl/Estrategias-para-ganar-el-juego-de-Damas.htm](http://juegos.about.com/od/juegos_clasicos/fl/Estrategias-para-ganar-el-juego-de-Damas.htm)

Las reglas las obtuvimos de:

<http://www.magnojuegos.com/juegosonline/damas/reglas>

<http://reglasdejuegosimples.blogspot.cl/2013/05/damas-reglas-simples.html>