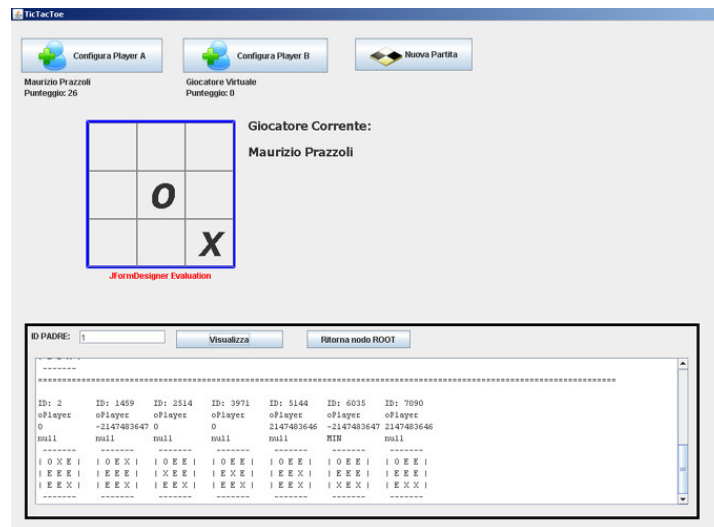


# Progetto:

# TRIS TIC-TAC-TOE



## Copyright and License

Copyright (c) 2010 MAURIZIO PRAZZOLI

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at <http://www.gnu.org/copyleft/fdl.html>.

## Disclaimer

No liability for the contents of this document can be accepted. Use the concepts, examples and information at your own risk. There may be errors and inaccuracies, that could be damaging to your system. Proceed with caution, and although this is highly unlikely, the author(s) do not take any responsibility. All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

## Informazioni

Progetto ideato e sviluppato da Maurizio Prazzoli, iscritto all'Università degli studi di Parma, facoltà di Ingegneria Informatica. Il seguente progetto è stato presentato per l'esame di Ingegneria del Software A relativamente all'anno accademico 2009/2010. Per ogni informazione / suggerimento / commento / codice sorgente, scrivere a: [maurizio@prazzoli.it](mailto:maurizio@prazzoli.it)

# INDICE CONTENUTI

Introduzione.....	3
Algoritmo Minimax .....	3
Implementazione.....	4
Classe board.....	4
Classe AI .....	5
Classe cell .....	6
Classe player .....	6
Classe game .....	6
Diagramma Classi Pacchetto it.prazzoli.tictactoe.....	7
Interfaccia Grafica .....	8
Diagramma Classi Implementazione Interfaccia Grafica .....	9
Punti di interesse .....	10
Esecuzione Progetto.....	12
Esempi di funzionamento.....	13
Gioco a due .....	13
Giocatore con Avversario Virtuale - algoritmo Ricerca Completa .....	14
Giocatore con Avversario Virtuale - algoritmo Alpha-beta pruning .....	16
Analisi dei risultati .....	18
Strumenti e Riferimenti .....	19

# Introduzione

I giochi a due, sono giochi in cui i due giocatori competono a vicenda uno contro l'altro. L'obiettivo di ciascun giocatore è quello di scegliere una mossa che massimizza il proprio punteggio, cercando allo stesso tempo di minimizzare il punteggio del giocatore avversario. Tra questi giochi è possibile definire come sottocategoria, i giochi a somma zero. Con questo si intende che il guadagno ( o perdita ) di un giocatore risulta esattamente controbilanciato dalla perdita ( o vincita ) dell'altro giocatore. Consideriamo ora il gioco del Tis (Tic-Tac-Toe). In questo gioco, i giocatori giocano a turni alterni eseguendo una mossa ciascuno. L'analisi di questo gioco è considerata interessante in quanto il numero di “ possibili mosse ” risulta ridotto se paragonato a quello di altri giochi come gli scacchi. Questa condizione consente di ricercare e valutare, data una qualunque configurazione del gioco, tutte le possibili scelte come “ prossima mossa ”. Una volta completata la ricerca, la scelta avverrà mediante una analisi dei risultati in modo da selezionare la mossa che conduce con maggiore facilità ad una vittoria, consentendoci talvolta di potere decidere il percorso più breve. Per raggiungere questo risultato risulta conveniente valutare alcuni tipi di algoritmi utilizzati nella soluzione di questi problemi.

## Algoritmo Minimax

Definiamo albero del gioco: l'albero costruito utilizzando come nodo radice l'attuale configurazione del gioco, i nodi figli di questo “ nodo radice ” sono tanti quanti le possibili mosse per il giocatore corrente. Per ciascun nodo figlio è possibile analizzare gli eventuali nodi figli, che corrispondono a tutte le possibili mosse per il giocatore avversario. Iterando il ragionamento per tutti i nodi, otteniamo l'albero completo del gioco. Per i giochi semplici, come il Tris, esistono algoritmi che eseguendo una analisi, più o meno completa dell'albero del gioco, consentono di estrarre la mossa più conveniente da eseguire. L'algoritmo più diffuso in questo contesto è chiamato MINIMAX. A ciascun nodo è possibile assegnare un valore che misura la bontà di tale configurazione. Nel nostro esempio supporremo che detta funzione  $f(n)$  restituisca il numero di posizioni di vincita non bloccate dall'avversario valutate dal giocatore corrente meno il numero di posizioni di vincita non bloccate dall'avversario valutate dal giocatore avversario.

# Implementazione

Per eseguire l'implementazione del gioco vengono utilizzate diverse classi. Nel seguito verranno dapprima rappresentate le classi delegate alla gestione della logica del gioco contenute nel package `it.prazzoli.tictactoe`, successivamente sarà descritta l'interfaccia grafica. Le classi utilizzate per l'astrazione del gioco sono le seguenti:

- **classe board**  
Questa classe rappresenta una generica tavola da gioco.
- **classe cell**  
Questa classe rappresenta una generica cella, intesa come un singolo riquadro di una generica tavola da gioco.
- **classe player**  
Questa classe rappresenta un generico giocatore.
- **Classe AI**  
Questa classe rappresenta il modulo di AI (Artificial Intelligence o Intelligenza Artificiale) utilizzato per l'implementazione dell'algoritmo MINIMAX.
- **Classe AIConfiguration**  
Questa classe rappresenta la configurazione relativa al modulo AI. Il motivo per cui si è deciso di tenere la configurazione distinta dal modulo AI è dovuta al fatto che un singolo modulo AI può lavorare in modo differente variando la configurazione.
- **Classe game**  
Questa classe contiene gli oggetti sopra descritti e consente di implementare il gioco.

Mediante la composizione degli oggetti sopra elencati, riusciamo a descrivere il gioco del Tris ed i rispettivi giocatori. Possiamo vedere che questa composizione avvenga nella classe `game`, la quale assembla le altre classi. Per la descrizione dettagliata delle classi e dei relativi metodi, si rimanda alla documentazione della classe. Nel seguito saranno illustrati concetti più importanti e i metodi più rilevanti delle singole classi, inoltre nella seguente descrizione non sono descritte tutte le classi, ma solamente quelle necessarie per consentire il gioco.

## ***Classe board***

La classe `board` descrive la tavola del gioco, questa è composta da nove celle. Ciascuna cella può contenere X O oppure E ( qui si suppone di giocare con i simboli { X , O , E } il simbolo E rappresenta il valore vuoto ). Sebbene sia possibile utilizzare qualunque combinazione di caratteri si decide questo in quanto standard, in particolare si utilizza E per rendere più chiara la rappresentazione. La classe consente di gestire `board` in due modi:

- **come matrice 3X3.** Vi sono tre righe e tre colonne, le dimensioni di questa matrice sono state parametrizzate, sarebbe pertanto possibile utilizzare matrici di dimensioni differenti modificando un parametro della classe.
- **come numero intero.** Risulta conveniente riuscire a rappresentare e gestire la configurazione attuale della board utilizzando un numero intero; il funzionamento questa caratteristica si basa sul fatto che ogni casella può contenere O, X, E. Conveniamo di utilizzare un numero a 32 bit per rappresentare la board, la gestione di valori interi risulta comoda e talvolta atomica. Se consideriamo la seguente mappatura tra valori e posizioni:

0	1	2
3	4	5
6	7	8

X = 01b      O = 10b      E = 11b

Possiamo ora, in modo agevole, eseguire tale conversione concatenando i valori e formando un numero binario dove le cifre meno significative LSB sono a destra :

AA	BB	CC	DD	EE	FF	GG	HH	II
Cella_0	Cella_1	Cella_2	Cella_3	Cella_4	Cella_5	Cella_6	Cella_7	Cella_8

Il numero binario che rappresenta la board risulterà dalla concatenazione dei valori:

AABBCCDDEEFFGGHHII

A questo punto il numero binario così ottenuto viene convertito in intero.

E' possibile applicare l'operazione inversa trasformando l'intero in binario, andando poi a popolare i valori nella matrice.

L'utilizzo di un modo piuttosto che l'altro è indifferente ai fini del funzionamento. Esempio: nella memorizzazione di board dal database è conveniente utilizzare notazione intero, per rappresentare le board su schermo è conveniente utilizzare notazioni matriciali. La notazione intero risulta migliore ai fini di prestazioni di esecuzione, ma non risulta interpretabile direttamente.

## Classe AI

Il funzionamento della logica della classe è basato su due funzioni ricorsive che si chiamano vicendevolmente con lo scopo di eseguire una analisi dell'albero del gioco. Questa analisi, viene compiuta secondo la modalità desiderata contenuta nel modulo di configurazione. Per analizzare il funzionamento, possiamo supporre di giocare una partita tra un giocatore ( GIOCATORE1 ) e un avversario virtuale, le cui decisioni sono affidate all'oggetto AI. Supponiamo di trovarci nella condizione in cui GIOCATORE1 ha appena eseguito la scelta della mossa, il modulo AI deve dunque decidere la mossa successiva partendo dalla configurazione corrente del gioco.

Si verifica dapprima se il gioco può proseguire oppure se la mossa appena eseguita da parte di GIOCATORE1 ha portato ad una condizione di termine del gioco ( game over ). In caso di game over, si valuta lo stato della board corrente e supposto che il gioco non può proseguire deve

necessariamente contenere una condizione di vincita o pareggio. In caso di vincita significa che GIOCATORE1 ha vinto la partita, la funzione restituirà un valore di ritorno corrispondente a – INFINITO, se la partita è pareggiata si restituirà ZERO. I valori restituiti dalla funzione che si occupa di analizzare la board, sono dovuti al fatto che il modulo AI si occupa di massimizzare le vincite per il giocatore virtuale. Se al contrario il gioco non è terminato, il modulo AI analizza l'albero del gioco selezionando una mossa tra quelle possibili, successivamente la configurazione del gioco così ottenuta, viene passata alla funzione che si occupa di analizzare la board dal punto di vista del GIOCATORE1.

E' possibile selezionare differenti tipologie di algoritmi tra cui: ricerca completa ed esaustiva dell'albero del gioco, ricerca eseguita con algoritmo Alpha-beta pruning.

### ***Classe cell***

Questa classe rappresenta una generica cella, intesa come un singolo riquadro di una generica tavola da gioco. L'oggetto offre metodi pubblici per dialogare con questo oggetto. Le celle sono caratterizzate dal numero di riga e di colonna, nel dettaglio, queste sono disposte come segue:

	Colonna 0	Colonna 1	Colonna 2
Riga 0	0	1	2
Riga 1	3	4	5
Riga 2	6	7	8

Risulta dunque possibile fare riferimento alla singola cella in due modi. Mediante riga e colonna oppure mediante il riferimento assoluto.

L'utilizzo di un modo piuttosto che l'altro è indifferente ai fini del funzionamento. La notazione mediante il riferimento assoluto, risulta migliore ai fini di prestazioni di esecuzione, ma non risulta interpretabile direttamente.

### ***Classe player***

Questa classe rappresenta un generico giocatore sia esso virtuale oppure umano. Sono implementati alcuni metodi per facilitare la lettura e l'aggiornamento delle informazioni personali legate al giocatore quali nome, cognome, userID, password e punteggio. Dal momento che le informazioni devono essere persistenti, queste sono mantenute in un database, mediante appositi metodi è consentito l'accesso in lettura e scrittura delle informazioni da e verso la base dati. Nel caso venga utilizzato un giocatore virtuale, la classe player, “ possiede un ” oggetto di tipo **AIConfiguration** che consente di mantenere la configurazione del modulo AI, che verrà fornita alla procedura di AI quanto necessario.

### ***Classe game***

Questa classe rappresenta la partita, funziona da classe contenitore. Serve per astrarre una partita del gioco, assemblando tutti gli oggetti necessari.

# **Diagramma Classi Pacchetto it.prazzoli.tictactoe**

## Interfaccia Grafica

L'interfaccia grafica è realizzata mediante design pattern MVC. e' stato utilizzato il pattern che prevede il controller come mediatore. Informazioni dettagliate sono disponibili al seguente indirizzo:

<http://java.sun.com/developer/technicalArticles/javase/mvc/#3>



# **Diagramma Classi Implementazione Interfaccia Grafica**

## Punti di interesse

Vengono riportate di seguito alcune funzionalità implementate nelle classi:

- **Gestione Eccezioni:** dove necessario sono gestite le eccezioni nei vari moduli e gestite all'interno del modulo stesso oppure, ne viene richiesta la gestione ad un livello superiore propagando l'eccezione mediante la clausola throws.
- **Organizzazione classi in Java Package.**
- **Modulo Debug:** questo modulo si occupa di eseguire il tracciamento delle operazioni di Intelligenza Artificiale consentendo di memorizzare tutte le board analizzate ed il punteggio a loro assegnato. Al termine di ogni elaborazione eseguita dal modulo di Intelligenza Artificiale vengono fornite informazioni circa il numero di board analizzate ed il tempo impiegato.
- **Metodi sincronizzati:** viene fatto uso di metodi sincronizzati ove necessario.
- **Multithread:** le chiamate al modulo AI vengono eseguite in un thread separato.
- **Javadoc:** le classi contenute all'interno del package it.prazzoli.tictactoe sono commentate mediante l'utilizzo di Javadoc.
- **MVC:** model-view-controller (MVC) design.
- **Diverse configurazioni per il modulo AI:** il modulo AI consente ricevere un oggetto di tipo AIConfiguration che contiene la configurazione di tale modulo. E' possibile selezionare differenti tipologie di algoritmi tra cui: ricerca completa ed esaustiva dell'albero del gioco, ricerca eseguita con algoritmo Alpha-beta pruning.
- **Integrazione con Database:** l'applicativo è integrato con un Database Microsoft Access per consentire la gestione dei giocatori e dei punteggi ad essi associati
- **Utilizzo di JUnit test framework.**
- **Utilizzo di eUML2 Class Diagram Creation.**
- **Utilizzo Jform Designer.**
- **Gestione del cambio giocatore in modi differenti:** siccome il gioco si svolge tra due giocatori è possibile assegnare il valore 0 ad un giocatore ed il valore 1 all'altro. Per eseguire il cambio turno da un giocatore all'altro è possibile eseguire la negazione di una variabile di tipo boolean, oppure sottrarre 1 dal valore assoluto di una variabile di tipo int.

- **Punteggio giocatore:** al termine di ogni partita, viene aggiornato il punteggio associato ai giocatori, aggiungendo due punti in caso di vittoria e nessun punto in caso di perdita o pareggio. Tali informazioni sono persistenti in quanto sono memorizzate nel database.

## Esecuzione Progetto

Per eseguire il progetto è necessario completare le seguenti operazioni:

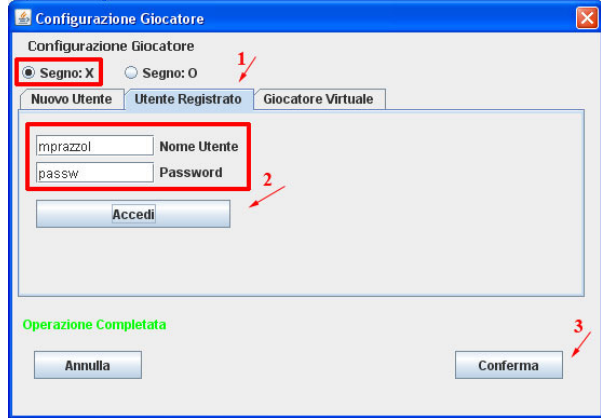
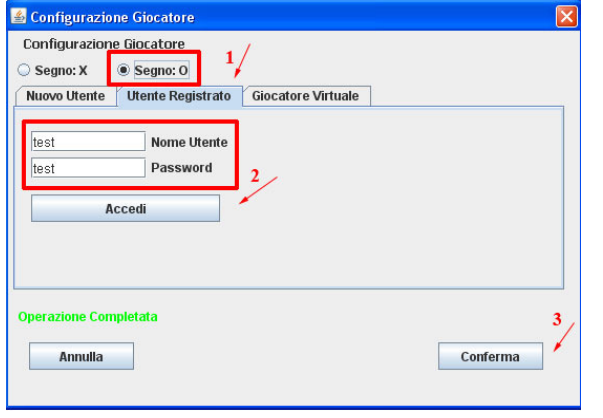
- Copiare il database AI\_DEBUG.mdb in c:\
- Estrarre il contenuto del file progetto.zip in una cartella
- Eseguire Eclipse:
- Creare una nuova Workspace ed importare il progetto selezionando la cartella utilizzata precedentemente..
- Aprire il file main contenuto nel (default package)
- Eseguire il progetto.

## Esempi di funzionamento

Prima di iniziare una partita, è necessario completare la configurazione di entrambi i giocatori ( nel caso il giocatore sia nuovo è possibile registrare l'utente e selezionare un simbolo di gioco. Vedremo di seguito alcuni esempi:

### *Gioco a due*

Configurare i giocatori come di seguito:

<p>Utente Registrato Utente Registrato <b>Nome Utente:</b> mprazzol <b>Password:</b> passw <b>Segno:</b> X</p>	<p>Utente Registrato <b>Nome Utente:</b> test <b>Password:</b> test <b>Segno:</b> O</p>
	

Al termine della configurazione iniziare una nuova partita mediante il pulsante Nuova Partita.

## ***Giocatore con Avversario Virtuale - algoritmo Ricerca Completa***

Configurare i giocatori come di seguito:

<p>Utente Registrato          Utente Registrato  <b>Nome Utente:</b> mprazzol  <b>Password:</b> passw  <b>Segno:</b> X</p>	<p>Giocatore Virtuale  <b>Nome Utente:</b> test  <b>Algoritmo:</b> MINIMAX Full Search  <b>Segno:</b> O</p>

Al termine della configurazione iniziare una nuova partita mediante il pulsante Nuova Partita.

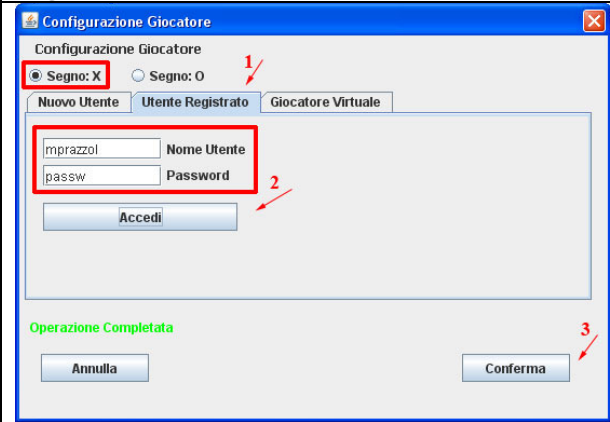

Seguire gli schemi di gioco indicati nella pagina seguente.

Schema gioco per partita configurata come Giocatore con Avversario virtuale  
 Algoritmo: MINIMAX Alpha-beta pruning

PLAYER A - MOSSA 1 Giocare in questo modo:	<table><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td>X</td></tr></table>									X	PLAYER B - MOSSA 2 Il modulo AI giocherà nel segunte modo analizzando 59705 boards:	<table><tr><td></td><td></td><td></td></tr><tr><td></td><td>O</td><td></td></tr><tr><td></td><td></td><td>X</td></tr></table>					O				X
		X																			
	O																				
		X																			
PLAYER A - MOSSA 3 Giocare in questo modo:	<table><tr><td>X</td><td></td><td></td></tr><tr><td></td><td>O</td><td></td></tr><tr><td></td><td></td><td>X</td></tr></table>	X				O				X	PLAYER B - MOSSA 4 Il modulo AI giocherà nel segunte modo analizzando 1053 boards:	<table><tr><td>X</td><td></td><td></td></tr><tr><td></td><td>O</td><td></td></tr><tr><td></td><td>O</td><td>X</td></tr></table>	X				O			O	X
X																					
	O																				
		X																			
X																					
	O																				
	O	X																			
PLAYER A - MOSSA 5 Giocare in questo modo:	<table><tr><td>X</td><td>X</td><td></td></tr><tr><td></td><td>O</td><td></td></tr><tr><td></td><td>O</td><td>X</td></tr></table>	X	X			O			O	X	PLAYER B - MOSSA 6 Il modulo AI giocherà nel segunte modo analizzando 47 boards:	<table><tr><td>X</td><td>X</td><td>O</td></tr><tr><td></td><td>O</td><td></td></tr><tr><td></td><td>O</td><td>X</td></tr></table>	X	X	O		O			O	X
X	X																				
	O																				
	O	X																			
X	X	O																			
	O																				
	O	X																			
PLAYER A - MOSSA 7 Giocare in questo modo:	<table><tr><td>X</td><td>X</td><td>O</td></tr><tr><td></td><td>O</td><td></td></tr><tr><td>X</td><td>O</td><td>X</td></tr></table>	X	X	O		O		X	O	X	PLAYER B - MOSSA 8 Il modulo AI giocherà nel segunte modo analizzando 5 boards:	<table><tr><td>X</td><td>X</td><td>O</td></tr><tr><td>O</td><td>O</td><td></td></tr><tr><td>X</td><td>O</td><td>X</td></tr></table>	X	X	O	O	O		X	O	X
X	X	O																			
	O																				
X	O	X																			
X	X	O																			
O	O																				
X	O	X																			
PLAYER A - MOSSA 9 Giocare in questo modo:	<table><tr><td>X</td><td>X</td><td>O</td></tr><tr><td>O</td><td>O</td><td>X</td></tr><tr><td>X</td><td>O</td><td>X</td></tr></table>	X	X	O	O	O	X	X	O	X											
X	X	O																			
O	O	X																			
X	O	X																			

### ***Giocatore con Avversario Virtuale - algoritmo Alpha-beta pruning***

Configurare i giocatori come di seguito:

<p>Utente Registrato Utente Registrato <b>Nome Utente:</b> mprazzol <b>Password:</b> passw <b>Segno:</b> X</p>	<p>Giocatore Virtuale <b>Nome Utente:</b> test <b>Algoritmo:</b> MINIMAX Alpha-beta pruning <b>Segno:</b> O</p>
	

Al termine della configurazione iniziare una nuova partita mediante il pulsante Nuova Partita.

Seguire gli schemi di gioco indicati nella pagina seguente.

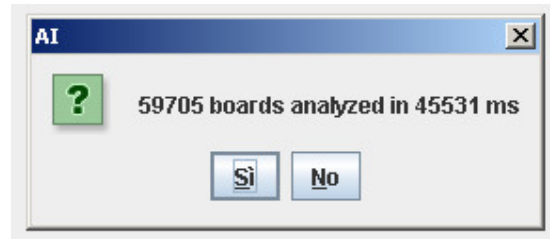


Schema gioco per test partita configurata come Giocatore con Avversario virtuale  
 Algoritmo: MINIMAX Alpha-beta pruning

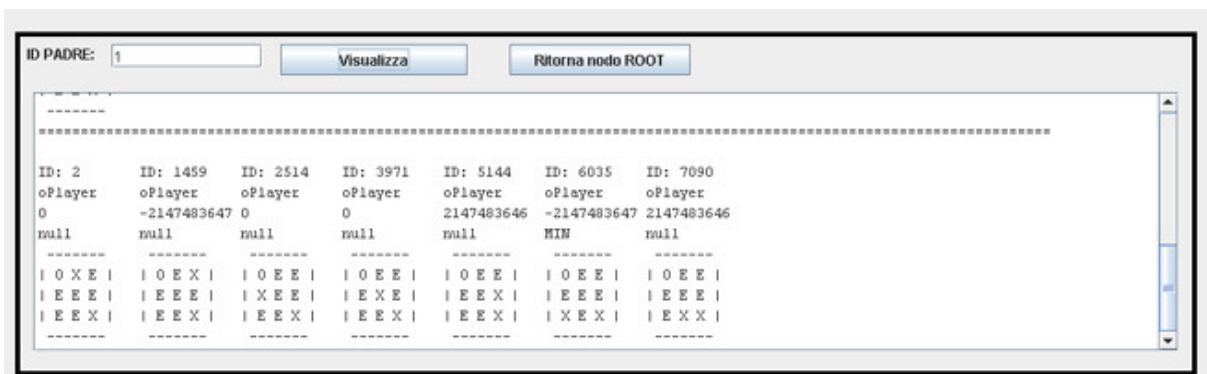
PLAYER A - MOSSA 1 Giocare in questo modo:	<table><tr><td>X</td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>	X									PLAYER B - MOSSA 2 Il modulo AI giocherà nel segunte modo analizzando 2338 boards:	<table><tr><td>X</td><td></td><td></td></tr><tr><td></td><td>O</td><td></td></tr><tr><td></td><td></td><td></td></tr></table>	X				O				
X																					
X																					
	O																				
PLAYER A - MOSSA 3 Giocare in questo modo:	<table><tr><td>X</td><td></td><td></td></tr><tr><td></td><td>O</td><td></td></tr><tr><td></td><td></td><td>X</td></tr></table>	X				O				X	PLAYER B - MOSSA 4 Il modulo AI giocherà nel segunte modo analizzando 318 boards:	<table><tr><td>X</td><td>O</td><td></td></tr><tr><td></td><td>O</td><td></td></tr><tr><td></td><td></td><td>X</td></tr></table>	X	O			O				X
X																					
	O																				
		X																			
X	O																				
	O																				
		X																			
PLAYER A - MOSSA 5 Giocare in questo modo:	<table><tr><td>X</td><td>O</td><td></td></tr><tr><td></td><td>O</td><td></td></tr><tr><td></td><td>X</td><td>X</td></tr></table>	X	O			O			X	X	PLAYER B - MOSSA 6 Il modulo AI giocherà nel segunte modo analizzando 42 boards:	<table><tr><td>X</td><td>O</td><td></td></tr><tr><td></td><td>O</td><td></td></tr><tr><td>O</td><td>X</td><td>X</td></tr></table>	X	O			O		O	X	X
X	O																				
	O																				
	X	X																			
X	O																				
	O																				
O	X	X																			
PLAYER A - MOSSA 7 Giocare in questo modo:	<table><tr><td>X</td><td>O</td><td>X</td></tr><tr><td></td><td>O</td><td></td></tr><tr><td>O</td><td>O</td><td>X</td></tr></table>	X	O	X		O		O	O	X	PLAYER B - MOSSA 8 Il modulo AI giocherà nel segunte modo analizzando 5 boards:	<table><tr><td>X</td><td>X</td><td>O</td></tr><tr><td></td><td>O</td><td>O</td></tr><tr><td>X</td><td>O</td><td>X</td></tr></table>	X	X	O		O	O	X	O	X
X	O	X																			
	O																				
O	O	X																			
X	X	O																			
	O	O																			
X	O	X																			
PLAYER A - MOSSA 9 Giocare in questo modo:	<table><tr><td>X</td><td>X</td><td>O</td></tr><tr><td>X</td><td>O</td><td>O</td></tr><tr><td>X</td><td>O</td><td>X</td></tr></table>	X	X	O	X	O	O	X	O	X											
X	X	O																			
X	O	O																			
X	O	X																			

## Analisi dei risultati

Mediante il modulo Debug, al termine di ogni elaborazione eseguita dal modulo di Intelligenza Artificiale, è possibile avere informazioni circa il numero di board analizzate ed il tempo impiegato.



E' possibile verificare tutte le board analizzate ed il punteggio a loro assegnato. Per accedere a tali informazioni, è necessario utilizzare il pulsante **Ritorna nodo ROOT** per visualizzare il primo nodo dell'albero del gioco, è poi possibile navigare all'interno dell'albero, inserendo l'ID del nodo che si desidera esplorare nella casella di testo e successivamente premere **Visualizza**.



## Strumenti e Riferimenti

Nella realizzazione del seguente progetto sono stati utilizzati i seguenti componenti, strumenti e riferimenti:

- **ARTIFICIAL INTELLIGENCE A Systems Approach M. TIM JONES**  
Utilizzato come testo di riferimento dove vengono introdotte e presentati approcci ai problemi di Intelligenza Artificiale.
- **Eclipse IDE for Java Developers**  
Utilizzato come ambiente di sviluppo
- **eUML2 Studio Edition 3.2**  
Utilizzato come plug-in Eclipse per la realizzazione dei diagrammi classi in UML.
- **JFormDesigner - Java/Swing GUI Designer**  
Utilizzato per disegnare le interfacce grafiche
- **Javadoc**  
Utilizzato per la generazione automatica della documentazione del codice sorgente.
- **JUnit.org Resources for Test Driven Development**  
Utilizzato per la creazione delle classi di testing