Universidade do Porto

Faculdade de Engenharia

FEUP

Quadtria

Relatório Final

Inteligência Artificial

3º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo:

Marcos Brandão Duarte – 100509007 – ei10007@fe.up.pt

Pedro Miguel Salgado Dias - 090509055 – ei09055@fe.up.pt

Ricardo José Moreira Pinho – 090509045 – ei09045@fe.up.pt

20 de Maio de 2012

1. Objetivo

O objetivo deste trabalho foi a criação de um jogo de tabuleiro, o *Quadtria*, um jogo para 2 jogadores, onde cada um deles move as 5 peças que tem disponíveis de maneira a que 3 delas fiquem em posições adjacentes, formando um triângulo. Mais a frente serão enunciadas as regras em pormenor.

O jogo tem três modos possíveis: 2 jogadores humanos, 1 jogador humano contra o computador e computador contra computador. A inteligência artificial do computador foi feita usando um algoritmo de pesquisa com adversários, o Minimax, com Cortes Alfa-Beta.

O projeto está então dividido em três secções: a componente da lógica, onde estão feitas todas as funções necessárias para o funcionamento do jogo, a componente da inteligência artificial do computador e a componente gráfica, para uma melhor visualização do jogo.

A linguagem escolhida para a implementação foi Java.

2. Descrição

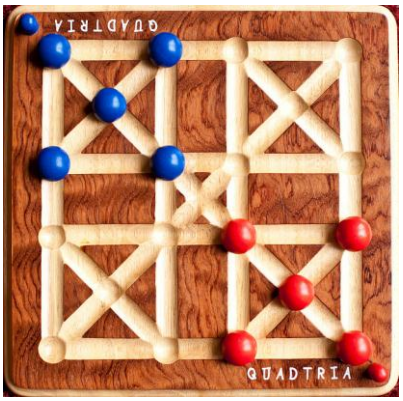
Este jogo tem regras de fácil compreensão, o que pode fazê-lo parecer mais simples do que realmente é. Para melhor perceção, passaremos a uma indicação pormenorizada das regras que o compõe, da maneira como está dividido o trabalho para fluidez do seu desenvolvimento e de uma explicação sucinta do algoritmo utilizado.

2.1. O jogo e suas regras

2.1.1. O tabuleiro

O tabuleiro é um quadrado composto por outros quatro, cada um com cinco "pontos", que formam uma espécie de pirâmide. Estes pontos são as posições em que cada jogador pode colocar as suas peças.

Cada uma dessas posições é ligada a outras, através de "caminhos" que indicam por onde se pode movimentar as peças. Fora deste quadrado, temos mais duas posições em cantos opostos onde são colocadas duas peças especiais que sinalizam os quadrados iniciais de cada jogador, onde se colocam todas as suas cinco peças.



Exemplo de tabuleiro no início do jogo.

2.1.2. Movimentos das peças

Existem algumas regras que se devem seguir para a movimentação das peças:

- Cada jogador, alternadamente, joga uma peça;
- Só é possível movimentar por onde haja um "caminho" para qualquer "ponto" livre adjacente;
- Não é possível saltar "pontos" já ocupados.

2.1.3. Objetivo

O objetivo que o jogador tem de alcançar é, com a utilização de 3 das suas peças, a formação de um triângulo em um dos quatro que formam o tabuleiro ("quadtria"). Quem conseguir fazê-lo primeiro é o vencedor.

Há uma limitação que consiste no facto de esta "quadtria" no quadrado inicial do próprio jogador, só poder ser feita após a retirada de todas as suas 5 peças do mesmo, ficando assim livre de qualquer peça sua. Quando isto acontece, o sinalizador do quadrado inicial deve ser retirado, mostrando que já se pode fazer "quadtria" no mesmo.

2.2. Partes do Trabalho

Conforme já foi enunciado, o desenvolvimento deste jogo e suas regras tem uma componente essencial relacionada com o jogador representado pelo computador, cuja inteligência artificial é baseada no algoritmo de pesquisa de soluções com adversário, Minimax, na sua variante, com Cortes Alfa-Beta.

Portanto, o trabalho está dividido nas seguintes partes, correspondentes cada uma a um pacote do projeto Java:

- Parte lógica / *package* Quadtria – onde se implementam as regras do jogo;
- Parte de inteligência artificial / *package* Inteligence – contém o desenvolvimento do algoritmo de pesquisa em causa;
- Parte gráfica / *package* GUI – para proporcionar uma melhor jogabilidade, através de uma mais fácil interação do jogador com o jogo em si.

2.2.1. Parte Lógica / package Quadtria

Esta parte está dividida através de cinco ficheiros:

- AI.java – trata da jogada do computador utilizando os recursos fornecidos pela parte de inteligência artificial;
- Board.java – contém o tabuleiro e todas as suas especificidades, relativas as jogadas e suas condições e verificações do fim de jogo;
- Game.java – reúne os elementos que compõe o jogo, tabuleiro e jogador, fazendo a sua ligação;
- Main.java – de onde se executa o jogo e onde se produz a saída gráfica em modo de texto;
- Player.java – possui as informações necessárias para o jogador, como nome, se é jogador humano ou computador

2.2.2. Parte de Inteligência Artificial / package Inteligence

Esta parte, como já foi dito é a que trata do algoritmo a desenvolver e está dividida em dois ficheiros:

- Inteligence.java – onde se implementa o algoritmo de pesquisa de soluções;
- Node.java – ficheiro que complementa o primeiro, responsável pela manutenção de uma estrutura de árvore que é necessária para realização da pesquisa.

2.2.3. Parte Gráfica / package GUI

Parte que é responsável pela parte gráfica, fazendo toda a ligação com as outras duas partes, lógica e de inteligência, tornando a interação com o jogo mais fácil.

- Interface.java – faz a ligação da parte gráfica com as outras duas partes;
- Background.java – trata do desenho do estado do jogo em cada momento.

2.4. O Algoritmo Minimax com Cortes Alfa-Beta

Através do Minimax, pretende-se que o computador seja capaz de fazer a melhor jogada, baseando-se em heurísticas que permitem avaliar corretamente o valor de cada jogada, sendo que há a possibilidade de escolher o número de níveis de pesquisa que se pretende utilizar, já que traz consequências para a necessidade de processamento.

Com os cortes Alfa-Beta, a árvore gerada pelo Minimax não será visitada totalmente, poupando assim tempo do processamento de cada pedido de jogada, já que este elimina partes da árvore que não serão as escolhas mais favoráveis para um jogador.

Também foi utilizado o conceito de killer heuristic que pega nas jogadas que foram cortadas com alfa beta, que normalmente são as melhores/piores jogadas, conforme estejamos a verificar o max ou o min e executa essas jogadas primeiro, podendo gerar cortes mais cedo.

No nosso caso, quando os 2 jogadores são representados pelo computador, há 3 níveis de dificuldade que influenciam o nível de profundidade utilizado para pesquisa na árvore de jogadas:

- Fácil – nível de profundidade é 4;
- Médio - nível de profundidade é 5;
- Difícil - nível de profundidade é 6;

e 3 tipos de funcionamento do Minimax:

- AI s/cortes – não faz os cortes Alfa-Beta na árvore de jogada;
- AI c/cortes – realiza os cortes;
- AI c/Killer heuristic – realiza os cortes e utiliza o killer heuristic.

A heurística(H) utilizada tem a seguinte fórmula:

$$H = n1 * maxQ1 – n2 * maxQ2 + Qinit$$

Onde:

- n1 – número de jogadas possíveis para o jogador da vez;
- maxQ1 – o número de peças que o jogador da vez possui no quadrante para o qual vai jogar;
- n2 – número de jogadas possíveis para o jogador adversário, após essa jogada;
- maxQ2 – o número de peças que o jogador adversário possui no quadrante para o qual se vai jogar;
- Qinit – valor atribuído por se ter saído do quadrante inicial.

Portanto, com as combinações de nível de dificuldade e tipos de funcionamento do Minimax, temos os seguintes 9 tipos de jogo (computador contra computador):

Tipo de jogo (nível, tipo de AI)	**Número máximo de nós pesquisados por jogada**	**Número médio de jogadas por jogador até o fim do jogo**
Fácil, AI s/cortes	30.000	45
Fácil, AI c/cortes	8.000	45
Fácil, c/killer heuristic	2.500	45
Médio, AI s/cortes	50.000	9
Médio, AI c/cortes	16.000	22
Médio, c/killer heuristic	10.000	22
Difícil, AI s/cortes	50.000	7
Difícil, AI c/cortes	50.000	13
Difícil, c/killer heuristic	50.000	∞

Com base na análise destes dados, podemos verificar que a utilização dos cortes ou da combinação cortes/killer heuristic reduz em muito o número de nós analisados e consequentemente o tempo de processamento das jogadas realizadas.

No nível médio s/cortes e em todos do nível difícil, os valores dos números máximos de nós são sempre 50.000 devido a uma limitação introduzida definindo esse valor como máximo, para não se comprometer o tempo de processamento de uma jogada.

No nível difícil c/killer heuristic, o jogo não acaba.

2.5. Ambiente de desenvolvimento

O desenvolvimento do trabalho foi todo realizado utilizando o Microsoft Windows 7, no IDE Eclipse com a linguagem Java.

A parte gráfica utiliza o Java Swing e também o pacote extra MiG Layout Java.

Para se poder compilar o código no IDE é preciso adicionar as livrarias do projeto, a livraria do MiG Layout que está no ficheiro miglayout-4.0.jar.

Foram também utilizadas diversas máquinas para os testes, entre elas os computadores das salas da FEUP, que possuem processador Intel Core 2 Quad (Q9300) 2.5Ghz, 6 MB Cache L2, placa Gráfica NVIDIA Quadro NVS 290 256MB PCIe 16x e memória RAM 4 GB (2 x 2GB) DDR2 800 Mhz.

2.6. Avaliação do programa

Durante o desenvolvimento do programa, foram sendo analisados os resultados obtidos, através de mensagens passadas pelo programa, relativas a diversos dados, como número de nós analisados, valores da heurística, ramos da árvore cortados.

Com base nestes dados foi possível verificar e corrigir diversos problemas que foram surgindo ao longo do tempo.

No anexo C, pode ser consultada uma tabela das jogadas realizadas com as configurações descritas para o programa no modo de jogo CPUvsCPU para alguns dos modos de jogo possíveis.

3. Conclusões

Com a observação dos dados produzidos pelo programa, podemos verificar a eficácia dos algoritmos implementados e sem os quais a utilização do computador para a geração de jogadas seria lenta e consumiria muitos mais recursos do que aqueles que os necessários para a produção de resultados satisfatórios.

Se as máquinas utilizadas possuíssem mais recursos, talvez se pudesse aumentar os níveis de profundidade utilizados no algoritmo. Embora já estejam sendo utilizados níveis bastante bons.

A utilização de diferentes funções de heurística seria um dos pontos a considerar e que poderia levar a melhor/pior eficácia do algoritmo, podendo reduzir/aumentar a qualidade das jogadas realizadas pelo computador.

4. Recursos – Bibliografia e Software

- Programa feito em Java, usando o Eclipse como IDE.
- MiG Layout Java, http://www.miglayout.com/
- André Ferreira, Francisco Pinto, José Bateira, http://f.rancis.co/iart/#/, Minimax and Friends, visitado em Abril de 2012.
- http://stackoverflow.com/questions/9247816/alpha-beta-pruning-implementation-upon-minmax-in-tic-tac-toe, Alpha Beta pruning implementation upon Minmax in Tic Tac Toe.
- http://www.educationallearninggames.com/how-to-play-quadtria-game-rules.asp, How To Play Quadtria Game* - strategy board games, strategy games for kids, quadtria wooden games, visitado em Março de 2012.

5. Anexos

A. Manual do utilizador

Para jogar, a melhor maneira de o fazer e perceber o funcionamento e aplicação de regras é utilizando o modo gráfico. Portanto, basta executar o ficheiro run.bat que se encontra na pasta Quadtria.

Primeiro, surge uma janela onde se escolhe o modo de jogo:

- 1 jogador, corresponde ao modo, 1 jogador humano contra o computador;
- 2 jogadores, humano contra humano;
- CPUvsCPU que é o modo computador contra computador.

Em qualquer uma das escolhas, surge a janela de jogo, onde pode ver-se:

- o tabuleiro;
- uma barra na lateral direita com as informações relativas a cada um dos jogadores;
- acima desta um botão MENU, onde se pode voltar a janela inicial de jogo;
- e abaixo, um botão AJUDA que ativa/desativa uma barra na parte inferior da janela com mensagens de ajuda no decorrer do jogo.
- e uma barra de estado em cima do tabuleiro de jogo onde se mostra o jogador da vez, ou o jogador que ganhou.

Juntamente com esta janela de jogo, surge uma janela de opções do jogo. Escolhendo os modos em que há jogador(es) humano(s), nessa janela de opções é perguntado o nome desse(s). Escolhendo o modo em que há um jogador representado pelo computador, é perguntado o nível que queremos (fácil, médio e difícil) e o tipo da inteligência artificial a ser utilizado (AI s/cortes, AI c/cortes, AI c/killer heuristic). No modo CPUvsCPU, é ainda perguntado se quer jogar no modo rápido.

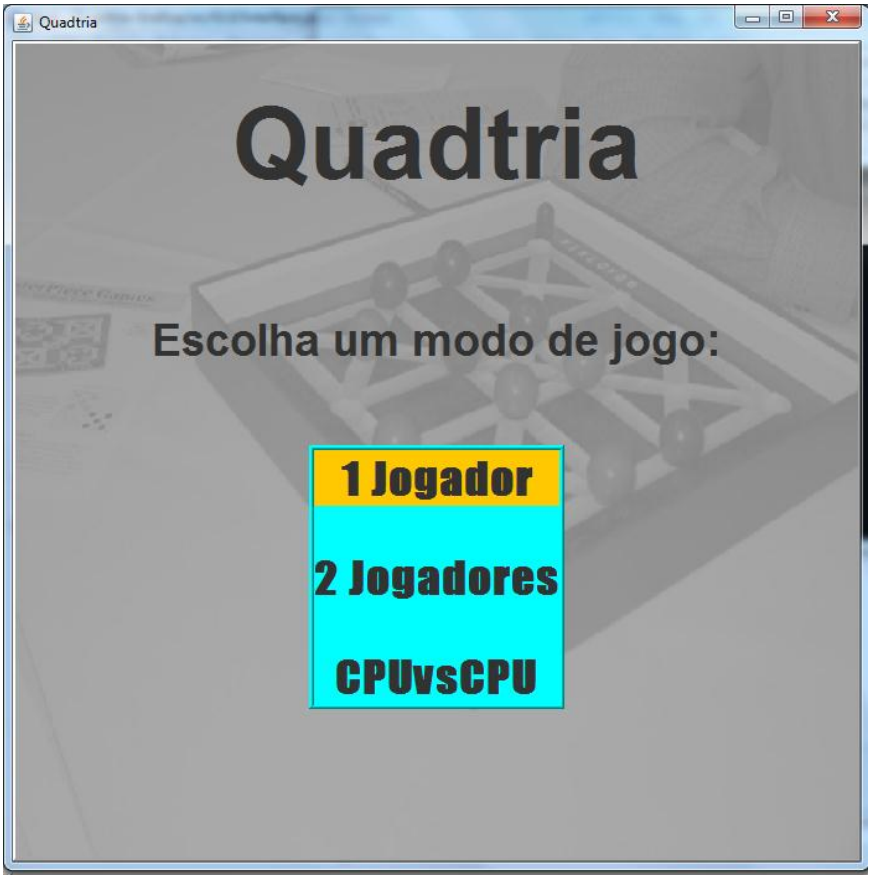
Depois destas escolhas, um jogador humano procede da seguinte maneira:

- Ativa/desativa a peça que quer jogar, clicando na mesma;
- São marcadas as casas para as quais pode jogar;
- Escolhe qual delas pretende e clica na mesma.

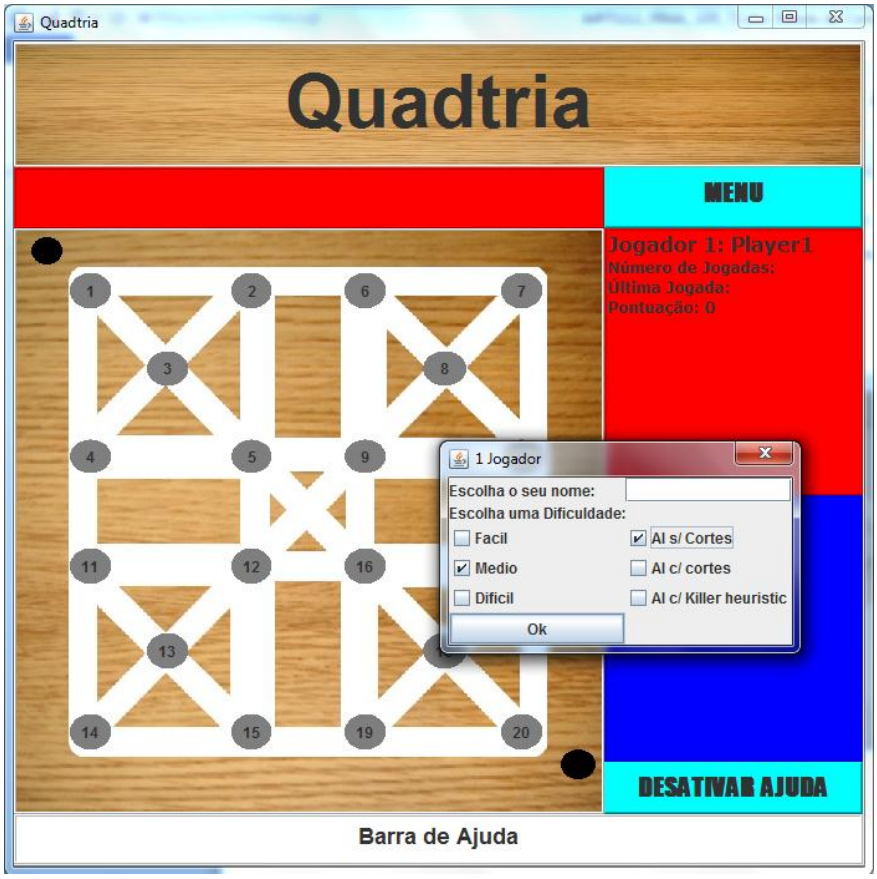
O jogador representado pelo CPU joga automaticamente, a não ser que se esteja no modo CPUvsCPU em que não se ativou o modo rápido e portanto é preciso pressionar a barra de espaços para seguir para a próxima jogada.

B. Exemplo de uma execução

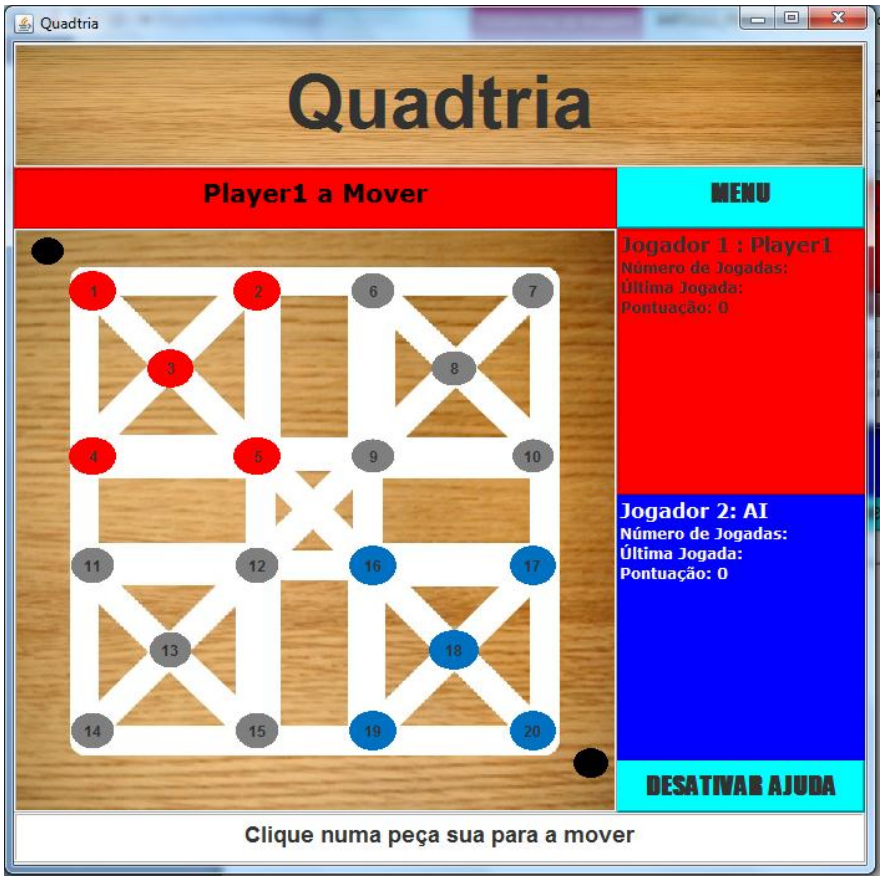
Neste exemplo, iremos mostrar o funcionamento no modo gráfico, escolhendo a opção de 1 Jogador e deixando sempre ativa a opção da barra de ajuda.



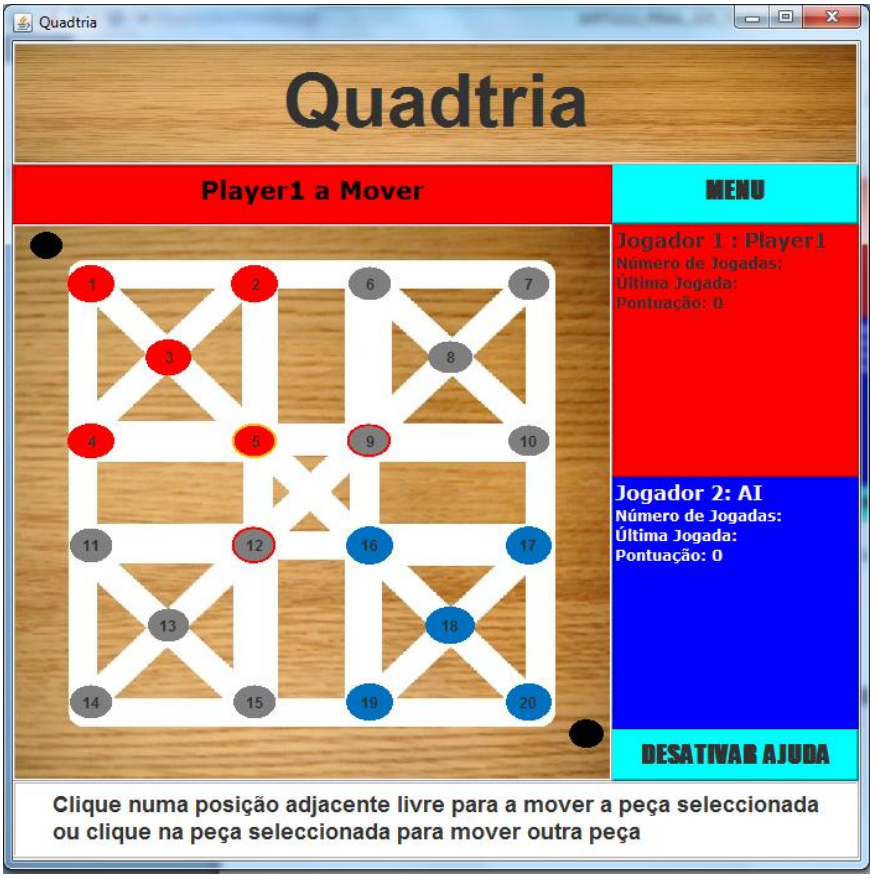
Janela de escolha do modo de jogo.



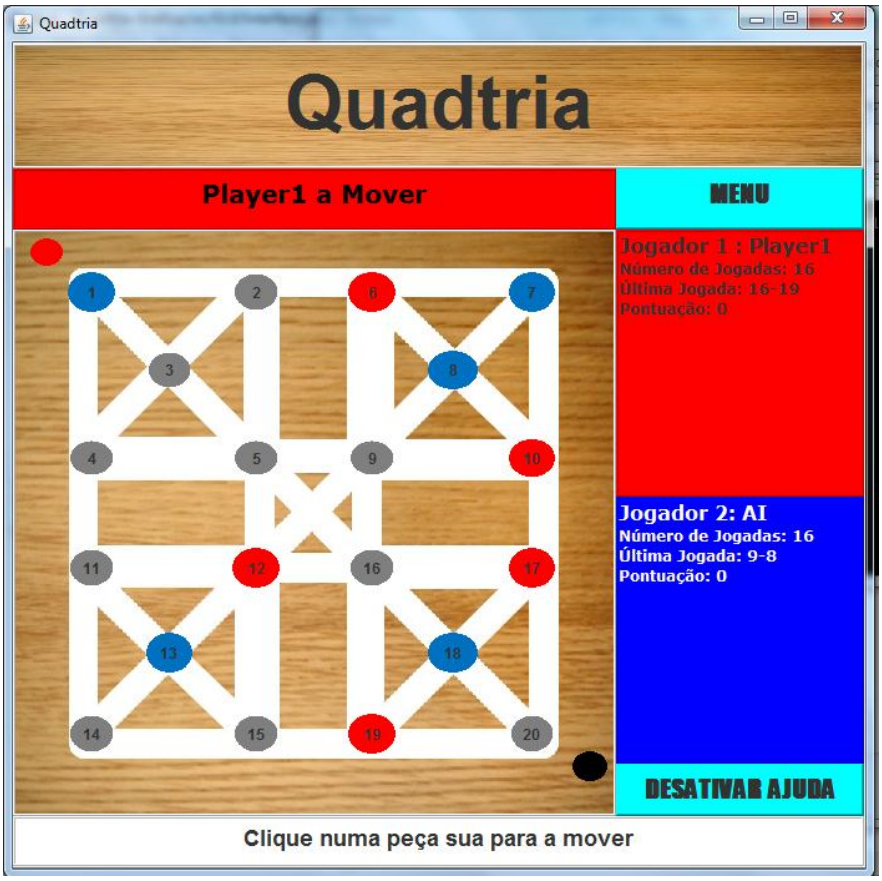
Janela de jogo e janela de escolha de opções.



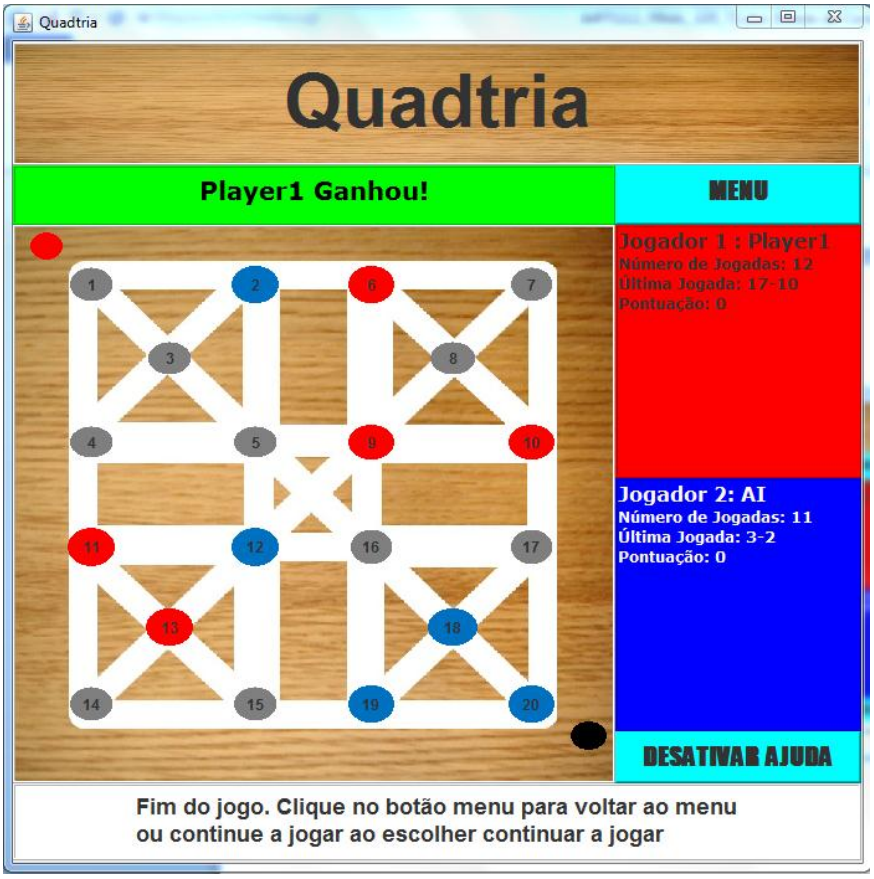
Janela de jogo na sua configuração inicial.



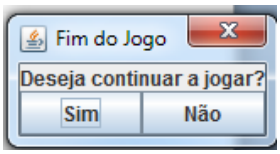
Janela de jogo, com escolha de peça pelo jogador humano Player1.



Janela de jogo após algumas jogadas.



Janela final.



Janela de opção após fim do jogo.

C. Exemplos de jogadas

Jogada nº	Jogador	Nível de dificuldade(1 - fácil, 2 - médio, 2 - difícil), AI(1 - AI s/cortes, 2 - AI c/cortes, 3- AI c/killer heuristic)					
		1, 3	2, 1	2, 2	3, 1	3, 2	3, 3
1	1	2 @ 6	5 @ 9	5 @ 9	2 @ 6	2 @ 6	2 @ 6
2	2	17 @ 10	16 @ 5	16 @ 5	16 @ 9	16 @ 9	16 @ 9
3	1	4 @ 11	4 @ 11	4 @ 11	1 @ 2	4 @ 11	4 @ 11
4	2	16 @ 9	5 @ 4	17 @ 10	17 @ 10	17 @ 10	17 @ 10
5	1	5 @ 12	11 @ 13	9 @ 16	3 @ 1	5 @ 12	5 @ 12
6	2	18 @ 16	4 @ 5	10 @ 9	18 @ 16	18 @ 16	10 @ 7
7	1	3 @ 5	9 @ 16	2 @ 6	4 @ 3	12 @ 15	6 @ 8
8	2	19 @ 15	17 @ 10	5 @ 12	19 @ 18	20 @ 17	18 @ 16
9	1	11 @ 14	1 @ 4	16 @ 5	5 @ 12	3 @ 5	1 @ 2
10	2	9 @ 8	10 @ 9	18 @ 16	20 @ 17	16 @ 12	9 @ 6
11	1	5 @ 9	2 @ 6	1 @ 2	6 @ 7	6 @ 7	12 @ 9
12	2	15 @ 13	9 @ 12	19 @ 15	9 @ 6	9 @ 6	7 @ 10
13	1	1 @ 4	6 @ 9	3 @ 1	3 @ 5	5 @ 9	3 @ 4
14	2	13 @ 11	12 @ 15	20 @ 19	16 @ 9	6 @ 8	16 @ 12
15	1	12 @ 15	16 @ 12	11 @ 13		1 @ 4	11 @ 14
16	2	16 @ 12	15 @ 14	15 @ 14		19 @ 16	19 @ 16
17	1	15 @ 19	4 @ 11	13 @ 15		11 @ 13	8 @ 7
18	2	20 @ 18		9 @ 8		17 @ 18	20 @ 17
19	1	6 @ 7		6 @ 9		7 @ 6	2 @ 5
20	2	8 @ 6		8 @ 6		10 @ 7	17 @ 18
21	1	14 @ 13		2 @ 3		15 @ 14	4 @ 11
22	2	11 @ 14		6 @ 2		12 @ 5	6 @ 2
23	1	4 @ 11		5 @ 4		9 @ 10	9 @ 6
24	2	14 @ 15		12 @ 11		8 @ 9	10 @ 9
25	1	19 @ 16		15 @ 12		4 @ 11	14 @ 13
26	2	15 @ 14		16 @ 5			16 @ 17
27	1	13 @ 15		9 @ 6			7 @ 10
28	2	18 @ 19		19 @ 16			18 @ 16
29	1	11 @ 4		12 @ 9			11 @ 14
30	2	19 @ 20		11 @ 12			2 @ 1
31	1	16 @ 18		4 @ 11			14 @ 15

Jogada nº	Jogador	Nível de dificuldade(1 - fácil, 2 - médio, 2 - difícil), AI(1 - AI s/cortes, 2 - AI c/cortes, 3- AI c/killer heuristic)					
		1, 3	2, 1	2, 2	3, 1	3, 2	3, 3
32	2	14 @ 13		14 @ 13			16 @ 19
33	1	7 @ 8		1 @ 4			5 @ 16
34	2	12 @ 5		16 @ 19			1 @ 3
35	1	15 @ 19		3 @ 1			6 @ 2
36	2	5 @ 16		2 @ 3			9 @ 6
37	1	19 @ 15		6 @ 2			13 @ 11
38	2	13 @ 12		5 @ 16			3 @ 4
39	1	8 @ 7		9 @ 6			10 @ 8
40	2	12 @ 13		13 @ 14			17 @ 10
41	1	4 @ 3		11 @ 13			8 @ 9
42	2	20 @ 19		12 @ 11			10 @ 17
43	1	18 @ 17		4 @ 5			16 @ 18
44	2	16 @ 5		19 @ 15			6 @ 7
45	1	17 @ 16					2 @ 1
46	2	10 @ 17					17 @ 10
47	1	3 @ 1					18 @ 16
48	2	6 @ 2					7 @ 6
49	1	16 @ 18					1 @ 2
50	2	13 @ 11					10 @ 17
51	1	1 @ 4					9 @ 8
52	2	2 @ 6					6 @ 9
53	1	4 @ 1					16 @ 18
54	2	17 @ 10					12 @ 5
55	1	1 @ 2					2 @ 3
56	2	11 @ 13					9 @ 6
57	1	2 @ 1					3 @ 2
58	2	10 @ 17					5 @ 9
59	1	15 @ 12					8 @ 10
60	2	6 @ 2					6 @ 8
61	1	9 @ 6					15 @ 12
62	2	5 @ 9					9 @ 5
63	1	7 @ 10					11 @ 13

Jogada nº	Jogador	Nível de dificuldade(1 - fácil, 2 - médio, 2 - difícil), AI(1 - AI s/cortes, 2 - AI c/cortes, 3- AI c/killer heuristic)					
		1, 3	2, 1	2, 2	3, 1	3, 2	3, 3
64	2	19 @ 16					19 @ 15
65	1	18 @ 19					13 @ 11
66	2	13 @ 11					17 @ 16
67	1	1 @ 4					2 @ 6
68	2	17 @ 18					8 @ 9
69	1	12 @ 15					6 @ 2
70	2	16 @ 12					16 @ 17
71	1	10 @ 17					2 @ 6
72	2	18 @ 16					5 @ 2
73	1	6 @ 8					12 @ 5
74	2	2 @ 3					15 @ 13
75	1	8 @ 6					18 @ 19
76	2	9 @ 8					13 @ 12
77	1	19 @ 20					11 @ 14
78	2	16 @ 19					17 @ 16
79	1	4 @ 5					14 @ 11
80	2	12 @ 16					12 @ 15
81	1	17 @ 10					10 @ 17
82	2	8 @ 9					9 @ 8
83	1	20 @ 17					17 @ 18
84	2	11 @ 4					15 @ 12
85	1	10 @ 7					18 @ 17
86	2	3 @ 1					8 @ 9
87	1	15 @ 12					19 @ 18
88	2	19 @ 18					12 @ 13
89	1	17 @ 10					18 @ 19

D. Ficheiro de Código

O código criado está subdividido conforme já foi descrito e encontra-se na pasta Quadtria.