|  |
| --- |
| faculdade de engenharia da universidade do porto |
|  |
| Hasami Shogi |
| Relatório Final 1º Projecto – Programação em Lógica (3º Ano – 1º Semestre) |
|  |
|  |
| **Autores:** |
| **Carlos Tiago Rocha Babo** |
| **Felipe de Souza Schmitt** |
| **Hélder Alexandre dos Santos Moreira** |
| **ID do Grupo: 106** |
| **10/3/2010** |

# Resumo

Este projecto tem como objectivo implementar uma versão do jogo “Hasami Shogi” em linguagem Prolog no contexto da unidade curricular “Programação em Lógica” do 3º Ano, 1º Semestre, do Mestrado Integrado em Engenharia Informática e Computação na Faculdade de Engenharia da Universidade do Porto. Deste modo, este relatório contém uma visão geral do funcionamento do jogo, as várias formas utilizadas para abordar os problemas ao longo do desenvolvimento do trabalho, assim como as expectativas iniciais para o trabalho acompanhadas com as suas resoluções.

# Abstract

This project has as its goal to implement the game “Hasami Shogi” using Prolog programming language learnt during the discipline of “Programação em Lógica” on the 1st Semester of the 3rd year in Integrated Master in Informatics and Computing Engineering at Faculty of Engineering at University of Porto. This paper has a global detail about the rules and methods of the game, the several ways of approach to the problems that we encountered during the development of the program, as well as the initial expectations and its resolutions.

Indíce

[Resumo 1](#_Toc276837113)

[Abstract 2](#_Toc276837114)

[1. Introdução 4](#_Toc276837115)

[1.1 Enquadramento 4](#_Toc276837116)

[2. Descrição do Problema 5](#_Toc276837117)

[2.1 Movimento das Peças 5](#_Toc276837118)

[2.2 Capturando Inimigos 6](#_Toc276837119)

[2.3 Modos de Jogo 7](#_Toc276837120)

[2.4 Realizar um Jogo 7](#_Toc276837121)

[3. Representação do Estado de Jogo 8](#_Toc276837122)

[3.1 O tabuleiro 8](#_Toc276837123)

[4. Representação de um Movimento 10](#_Toc276837124)

[5. Visualização do Tabuleiro 11](#_Toc276837125)

[6. Conclusões e Perspectivas de Desenvolvimento 12](#_Toc276837126)

[Bibliografia 13](#_Toc276837127)

[Anexo 14](#_Toc276837128)

[Ilustração 1 - Movimento no tabuleiro 4](#_Toc273726462)

[Ilustração 2 - Capturar uma peça na horizontal 4](file:///C:\Users\Tiago\Desktop\relatório_plog.docx#_Toc273726463)

[Ilustração 3 - Capturar uma peça na vertical 4](#_Toc273726464)

[Ilustração 4 - Capturar várias peças em simultâneo 5](#_Toc273726465)

[Ilustração 5 - Capturar uma peça num canto 5](#_Toc273726466)

[Ilustração 6 - Tabuleiro inicial 6](#_Toc273726467)

[Ilustração 7 - Tabuleiro numa fase intermédia 7](#_Toc273726468)

[Ilustração 8 - Tabuleiro numa situação final 7](#_Toc273726469)

[Ilustração 9 - Tabuleiro em linha de código 9](#_Toc273726470)

# Introdução

## Enquadramento

O trabalho descrito neste relatório surge no âmbito da disciplina de Programação em Lógica, do 1º semestre do ano lectivo de 2010/2011 do curso de Engenharia Informática e de Computação. Tendo como objectivo principal desenvolver um jogo de tabuleiro, foi utilizado o programa Swi-Prolog para o desenvolvimento de código bem como para os testes.

## Motivação

A motivação para a realização deste trabalho provêm da importância para adquirir novos conhecimentos em estilos de programação diferentes do que estamos habituados a trabalhar.

Durante o desenvolvimento deste trabalho escolhemos um jogo que fosse divertido e interessante assim como fizesse com que o utilizador tivesse que se manter concentrado durante o jogo pois é necessário realizar estratégias para se obter um melhor resultado. Uma das principais razões da nossa motivação foi a programação da inteligência artificial pois é algo do qual é considerado interessante e o resultado é gratificante.

No final espera-se com uma grande expectativa a implementação da interface gráfica a realizar na disciplina de Laboratório de Aplicações com Interface Gráfica, desta forma o trabalho tornar-se-á mais interessante devido ao aspecto visual.

## Objectivos

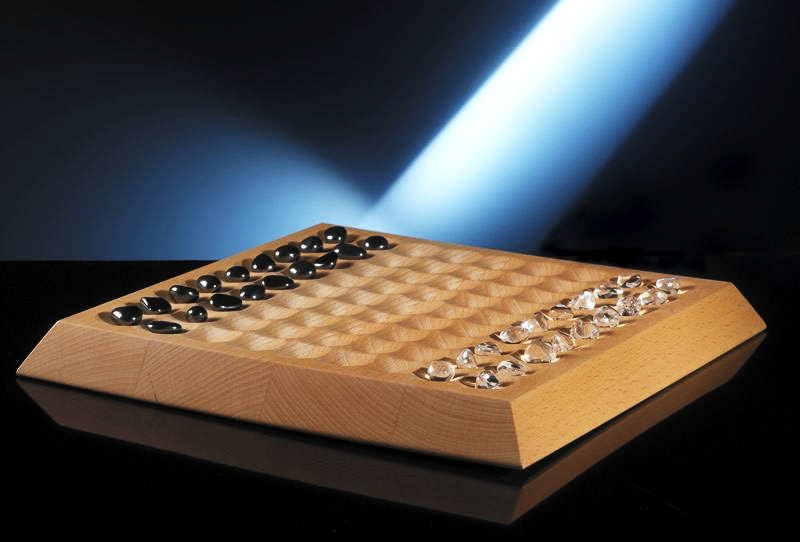
Os objectivos deste projecto relacionam-se essencialmente com o estudo e compreensão da programação em lógica, recorrendo ao Prolog, em contraste à programação funcional já estudada. No final do projecto esperamos conseguir discernir quais os problemas que podem ser resolvidos recorrendo à programação em lógica e, efectivamente, encontrar soluções. Para além disso, e em relação à vertente interdisciplinar deste projecto, pretendemos ser capazes de ligar duas linguagens de programação diferentes (Prolog e C++), recorrendo a *sockets*.

## Estrutura do relatório

O relatório contém uma descrição do problema, a representação do estado do jogo e de um movimento, a visualização do tabuleiro, as funções desenvolvidas para a aplicação da inteligência artificial e as conclusõe obtidas durante o desenvolvimento. Em anexo pode ser encontrado o código-fonte desenvolvido.

# Descrição do Problema

Hasami Shogi (はさみ将棋 *hasami shōgi*, *sandwiching chess*) é uma variante do jogo popular japonês denominado GoBang, que se tornou popular ao ser jogado nas ruas pelas crianças, pelo facto de não ser necessário um tabuleiro, podendo ser jogado utilizando simplesmente lápis e papel. O nome Shogi provém do tabuleiro (Shogi-ban), que para além de ser constituído por 9X9 espaços, é utilizado em diversos jogos.



Para as peças são usadas as 9 *fhuyo* (peões) brancas e 9 pretas, provenientes de um *shogi set* sendo que estas são posicionadas inicialmente na linha mais próxima do seu jogador.

Este jogo torna-se interessante, chamando-nos à atenção, visto que possui um conjunto de regras extremamente simples e que podem ser aprendidas rapidamente. No entanto, o Hasami Shogi é bastante aliciante devido à simplicidade aliada às múltiplas estratégias possíveis, mas que requerem grande concentração e previsão do desenrolar das jogadas.

## Movimento das Peças

O objectivo do Hashami Shogi é simples: conseguir retirar as peças do adversário prensando-as com as nossas. Para isso, podemos mover as peças para a frente, trás, esquerda, e direita, sem restrições de espaço.



Ilustração 1 - Movimento no tabuleiro

## Capturando Inimigos

O processo de captura passa por rodear a peça inimiga com duas nossas (a diagonal não é considerada). No entanto, se o inimigo mover uma peça para um espaço entre duas peças nossas, a captura não é efectuada.

Ilustração - Capturar uma peça na horizontal

Ilustração 3 - Capturar uma peça na vertical

É também possível usar a mesma técnica com várias peças inimigas.



Ilustração 4 - Capturar várias peças em simultâneo

O mesmo acontece se uma peça (e apenas uma) for encurralada num canto.

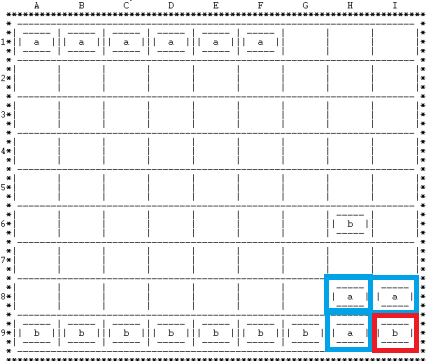


Ilustração 5 - Capturar uma peça num canto

## Modos de Jogo

O projecto terá pelo menos dois modos de jogo:

* Jogador vs Computador;
* Jogador vs Jogador;
* Computador vs Computador.

Se surgir mais alguma ideia para novos modos, a sua implementação será ponderada. Para além disso, o modo Jogador vs Computador terá, idealmente, mais do que uma dificuldade.

## Realizar um Jogo

O primeiro jogador é sorteado aleatoriamente e o primeiro a ficar com apenas duas peças em jogo, perde.

# Representação do Estado de Jogo

## O tabuleiro

O tabuleiro é constituído por oitenta e uma células, das quais dezoito se encontram inicialmente preenchidas. As peças são posicionadas de acordo com a seguinte imagem:



Ilustração 6 - Tabuleiro inicial

Matriz inicial do tabuleiro:

tabuleiro(

[[1,1,1,1,1,1,1,1,1],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[2,2,2,2,2,2,2,2,2]]).

Para construir o tabuleiro recorremos a uma lista de listas, onde o valor 1 e 2 indicam os jogadores e o 0 a posição vazia. Cada número tem associado um desenho específico, para que seja mais fácil e intuitivo jogar.

Com o decorrer do jogo, muitas das peças deixam o tabuleiro e num passo intermédio, o tabuleiro assemelha-se a isto:



Ilustração 7 - Tabuleiro numa fase intermédia

Já num estado final, quando apenas restarem duas peças de um dos lados, o jogo termina:



Ilustração 8 - Tabuleiro numa situação final

Neste caso, ganhou o jogador b.

Para se deslocar no tabuleiro, o jogador deve indicar a peça que pretende mover e o seu destino. Para isso, o tabuleiro contem indicações - alfabeto para as colunas e números para as linhas - para padronizar o método de jogo.

# Representação de um Movimento

Tal como já referido nas regras, no Hasami Shogi o jogador pode efectuar as jogadas na direcção vertical e horizontal, de quantas casa quiser, tendo em conta as peças do adversário no caminho e o limite do tabuleiro. Assim, cada jogada é representada pela peça a ser movida e pelas coordenadas para onde se pretende desloca-la.

O processo associado a cada jogada envolve três passos: validação da jogada, verificando se obdece às restrições descritas anteriormente; movimento definitivo da peça para a sua nova posição no tabuleiro, e, finalmente, é verificado se há peças do adversário conquistadas.

Deste modo, o primeiro passo chama a regra **valida\_jogada(X,Y,Tabuleiro,A,B),** verificando para a peça situada nas coordenadas X e Y do Tabuleiro, se é possível desloca-la para a nova posição - **posicao\_ocupada(X,Y,Tabuleiro,Resposta)**. O terceiro argumento é a matriz do tabuleiro, onde é guardada toda a informação sobre as peças e as suas posições. Por conseguinte, e depois de identificar se o movimento cumpre as regras, confirma-se se existe alguma peça entre as coordenadas iniciais e finais – **verifica\_caminho(X,Y,Tabuleiro,A,B, Resposta)**, e se não ultrapassam os limites do tabuleiro. Caso alguma destas averiguações falhe, a jogada é invalidada.

Se houver peças conquistadas, é feita a actualização do tabuleiro e da estrutura que guarda o número de peças conquistadas pelo jogador recorrendo à regra **verifica\_conquistas(Tabuleiro)**. Por fim, chama-se a regra **desenha(Tabuleiro)** para representar graficamente a jogada.

# Visualização do Tabuleiro

Para desenhar o tabuleiro, recorremos a uma série de factos, regras e à recursividade. A regra principal para o seu desenho é a seguinte:

**desenha(Tabuleiro)**:-

linhaLetrasV(X),

linhaLetras, nl,

linhaLimite, nl,

linhaDivH, nl,

printPecas(Tabuleiro,X),

linhaLimite.

Basicamente é criado um novo tabuleiro e todos os componentes desenhados. Depois de ser criado um novo tabuleiro e desenhados os limites superiores, é chamada a regra **printPecas**(T,X) que recebe o tabuleiro e percorre a matriz de jogo, imprimindo as peças respectivas:



Ilustração 9 - Tabuleiro em linha de código

As funções **printLinhaPecaX(Y)** auxiliam o desenho das três componentes de cada peça. O código completo do tabuleiro pode ser encontrado em anexo.

# Conclusões e Perspectivas de Desenvolvimento

Os objectivos propostos para a entrega do relatório intercalar foram alcançados. A representação do estado do jogo foi determinada, assim como foi realizada a visualização do tabuleiro através de caracteres de texto. Foram definidos alguns predicados em relação com as peças do tabuleiro.

Neste momento, apenas é possível visualizar o tabuleiro e as peças nas suas posições iniciais. Falta, assim:

* Implementar a jogada e os predicados provenientes da mesma;
* Recorrer a algoritmos específicos para avaliar a melhor jogada do computador;
* Interface com o utilizador (escolha do movimento, modo de jogo…);
* Estudar e implementar a ligação ao C++ (OpenGL), através de sockets.

Devido à falta de conhecimentos iniciais em relação à linguagem PROLOG, nesta fase inicial é necessária um estudo mais intenso para conseguirmos dominar bem a liguagem com o intuito a nas próximas etapas de desenvolvimento do projecto ser mais fácil conseguir realizar os objectivos. Estimamos que apenas 10% do projecto está concluído.

A abordagem a seguir passará por subdividir o projecto em problemas mais pequenos e depois de discutir em grupo a sua resolução, tentar distribui-los pelos três membros do grupo, para efectuar a sua implementação.

# Bibliografia

[1] Hasami Shogi, http://en.wikipedia.org/wiki/Hasami\_shogi (consultado em 30-09-2010)

[2] Hasami Shogi Web Game, http://www.afsgames.com/e/shogi.htm (consultado em 01-10-2010)

[3] Shogi, http://en.wikipedia.org/wiki/Shogi (consultado em 30-09-2010)

[4] Leon Sterling e Ehud Shapiro, The Art of Prolog Second Edition – Advanced Programming Thechniques: Capitulo 20, Secção 2 “Searching Game Trees”.

# Anexo

Código fonte já desenvolvido:

%DESENHAR O TABULEIRO

linhaLimite:-printLinha([' ',\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,' ']).

linhaLetras:-printLinha([' ',' ',' ',' ',' ',' ','A',' ',' ',' ',' ',' ',' ',' ','B',' ',' ',' ',' ',' ',' ',' ','C',' ',' ',' ',' ',' ',' ',' ','D',' ',' ',' ',' ',' ',' ',' ','E',' ',' ',' ',' ',' ',' ',' ','F',' ',' ',' ',' ',' ',' ',' ','G',' ',' ',' ',' ',' ',' ',' ','H',' ',' ',' ',' ',' ',' ',' ','I',' ',' ',' ',' ',' ']).

linhaNumerosV(['1','2','3','4','5','6','7','8','9']).

linhaDivH:-printLinha([' ',\*,' ',-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,' ',\*]).

tabuleiro(

[[1,1,1,1,1,1,1,1,1],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0],

[2,2,2,2,2,2,2,2,2]]).

piece1(1):- write(' ----- |').

piece1(2):- write(' ----- |').

piece1(0):- write(' |').

piece2(1):- write('| a ||').

piece2(2):- write('| b ||').

piece2(0):- write((' |')).

piece3(1):- write(' ----- |').

piece3(2):- write(' ----- |').

piece3(0):- write(' |').

printLinhaPeca([]).

printLinhaPeca([A|R]):-

piece1(A),

printLinhaPeca(R).

printLinhaPeca3([]).

printLinhaPeca3([A|R]):-

piece3(A),

printLinhaPeca3(R).

printLinhaPeca2([]).

printLinhaPeca2([A|R]):-

piece2(A),

printLinhaPeca2(R).

printLinha([]).

printLinha([A|R]):-

write(A),

printLinha(R).

printPecas([],[]).

printPecas([A|R],[X|Y]):-

write(' \*|'),

printLinhaPeca(A),

write('\*'), nl,

write(X),

write('\*|'),

printLinhaPeca2(A),

write('\*'), nl,

write(' \*|'),

printLinhaPeca3(A),

write('\*'), nl,

linhaDivH, nl,

printPecas(R, Y).

desenha(Tabuleiro):-

linhaNumerosV(X),

linhaLetras, nl,

linhaLimite, nl,

linhaDivH, nl,

printPecas(Tabuleiro,X),

linhaLimite.

%FIM DO DESENHO DO TABULEIRO

%AVALIACAO DA JOGADA%

%valida\_jogada(X,Y,Tabuleiro,A,B):-.

%posicao\_ocupada(X,Y,Tabuleiro,Resposta):-.

%verifica\_caminho(X,Y,Tabuleiro,A,B, Resposta):-.

%verifica\_conquistas(Tabuleiro):-.

%FIM DA AVALIACAO DA JOGADA%