**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 do Mestrado Integrado em Engenharia Informática e Computação. Este relatório coném uma visão geral do funcionamento do jogo assim como as nossas primeiras ideias relativamente à forma como iremos abordar o problema.

**Introdução**

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 devido ao facto de não ser necessário um tabuleiro, podendo ser jogado utilizando simplesmente lápis e papel. O nome Shogi é herdado devido ao tabuleiro onde é jogado (Shogi-ban), um tabuleiro com 9X9 espaços onde podem ser jogados diversos jogos.

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

Este jogo torna-se interessante e chamou-nos à atenção visto que possui um conjunto de regras extremamente simples e que podem ser aprendidas rapidamente, sendo no entanto bastante aliciante de jogar devido à simplicidade aliada ás multiplas estratégias possiveis.

**2. Descrição do Jogo**

**2.1 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.



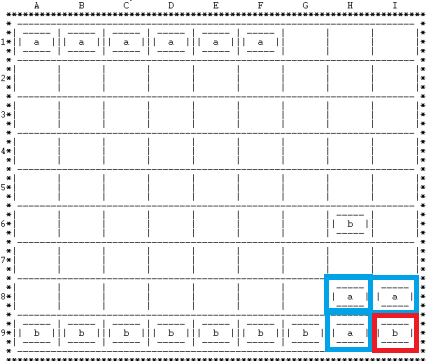
**2.2 Capturando inimigos**

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.

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



O mesmo acontece se uma peça for encurralada num canto.



**2.4 Níveis de dificuldade**

O projecto terá pelo menos dois níveis de dificuldade:

* Fácil: o computador efectua as jogadas, calculando cerca de 4 iterações de jogo, verificando qual a melhor opção de jogada;
* Difícil: semelhante à dificuldade fácil, mas com maior número de iterações.

Se houver oportunidade, será ainda adicionada a dificuldade “Imbatível” em que o computador percorre toda a árvore de escolhas e selecciona a melhor.

**2.3 Realizar um jogo**

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

**3. Tabuleiro**

**3.1 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:



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:



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



Neste caso, ganhou o jogador b.

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 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.

**3.2 Desenho do tabuleiro**

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

desenha:-

tabuleiro(T),

linhaLetrasV(X),

linhaLetras, nl,

linhaLimite, nl,

linhaDivH, nl,

printPecas(T,X),

linhaLimite.

Basicamente é criado um novo tabuleiro e todos os componentes desenhados. A regra printPecas(T,X) é recursiva:

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).

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

**4. Representação de um movimento**

**Schmitt**

**5. Conclusões e perspectivas de desenvolvimento**

**Schmitt**

**6. Bibliografia**

**Falta**