Licenciatura em Engenharia Informática e de Computadores

Fundamentos de Programação

Projecto - Segunda Parte

22 de Novembro de 2010

Damas havaianas

O Konane, ou damas havaianas, é um jogo de estratégia do antigo Havai, jogado num tabuleiro axadrezado quadrado, entre dois jogadores. Durante o jogo, os jogadores fazem jogadas, capturando peças do adversário, de acordo com determinadas regras. O objectivo do jogo é fazer com que o adversário não possa fazer mais jogadas.

1 Regras do jogo

As damas havaianas jogam-se num tabuleiro axadrezado, 8×8. O jogo é disputado entre 2 jogadores, começando cada um deles com 32 peças, de uma das cores, branco ou preto. No início do jogo, o tabuleiro está completamente preenchido, com as peças brancas nas casas pretas, e as peças pretas nas casas brancas.

O jogador que tem as peças pretas é o primeiro a jogar. A primeira jogada de cada jogador é diferente das restantes e consiste em escolher uma das suas peças para retirar do tabuleiro. A escolha da primeira peça a retirar obedece às seguintes regras:

- O jogador que tem as peças pretas deve escolher uma peça da diagonal principal, quer dos extremos, quer do centro. As 4 possibilidades encontram-se representadas na Figura 1.
- O jogador que tem as peças brancas terá que retirar uma peça de uma posição adjacente à retirada pelo adversário. Por exemplo, se o adversário tiver retirado a peça que falta no tabuleiro da Figura 2, as várias possibilidades para a peça a retirar pelo jogador da peças brancas são as assinaladas nesta figura.

Após esta fase inicial, o jogo decorre fazendo cada um dos jogadores uma jogada à vez, começando pelo jogador das peças pretas. Todas as jogadas são saltos por cima de uma ou mais peças do adversário, todas na mesma linha ou na mesma coluna, em qualquer sentido; não são permitidas jogadas na diagonal, nem é permitido mudar de direcção durante uma jogada. Não é obrigatório saltar por cima do número máximo de peças possível. Isto quer dizer que um jogador pode parar a sua jogada, apesar de ser possível continuá-la, capturando mais peças do adversário. Na Figura 3, estão assinaladas a verde escuro as posições finais das duas únicas possíveis jogadas, para a peça assinalada a verde claro. Se for feita a jogada mais longa, a situação após a jogada é a representada na Figura 4.

Quando um jogador não pode fazer mais jogadas, o adversário ganha o jogo.

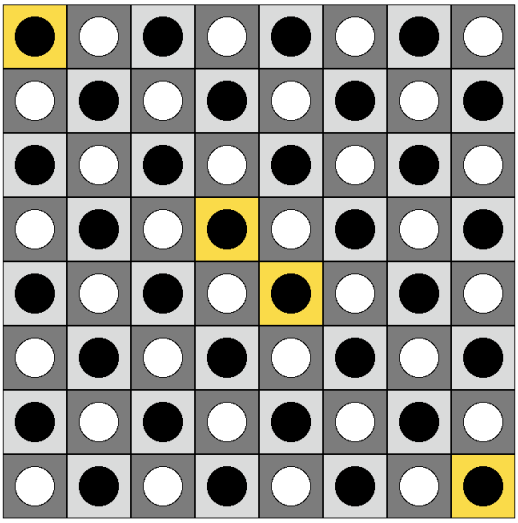

Figura 1: Possíveis posições da primeira peça a retirar pelo jogador das peças pretas.

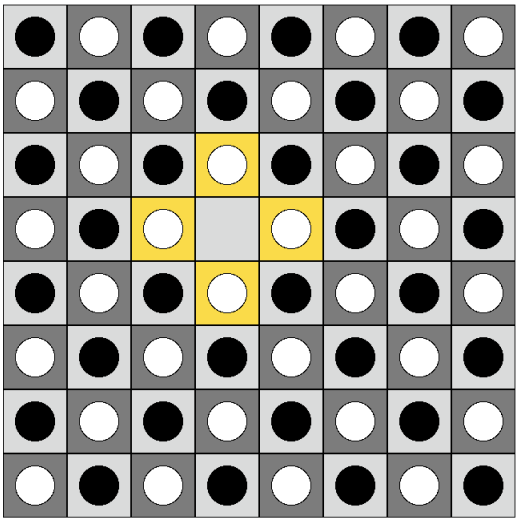

Figura 2: Possíveis posições da primeira peça a retirar pelo jogador das peças brancas, após o adversário ter tirado uma peça.

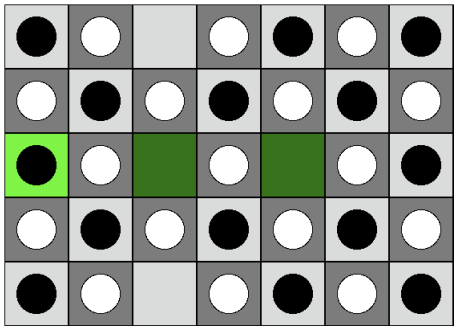

Figura 3: Jogadas possíveis.

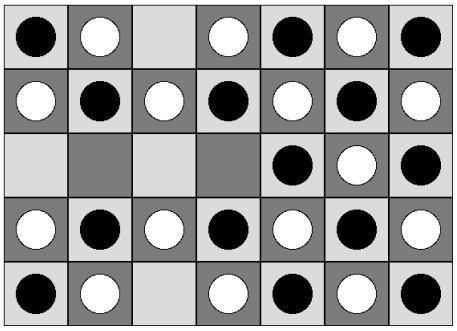

Figura 4: Situação após jogada.

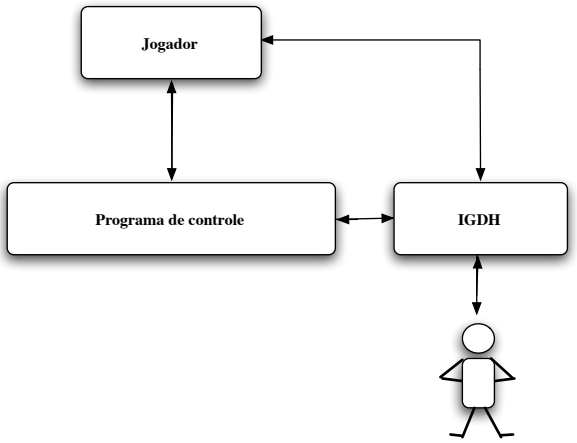

Figura 5: Módulos e suas interacções.

2 Objectivos do projecto

O projecto tem dois objectivos:

1. o desenvolvimento de um programa para controlar um jogo de damas havaianas entre dois jogadores, sendo um deles um jogador humano, e o outro um jogador automático;
2. o desenvolvimento de um programa que corresponda a um jogador de damas havaianas, o jogador automático.

Será disponibilizado um programa que produz uma interface gráfica para o jogo, programa esse que é designado por *IGDH (Interface Gráfica para Damas Havaianas)*.

A arquitectura do programa é apresentada na Figura 5.

2.1 O programa de controle

O programa de controle é responsável por controlar um jogo, desde o início ao fim.

Para tal, deverá começar por uma fase inicial, em que são determinadas as cores das peças de cada um dos jogadores, e as primeiras peças retiradas por cada um dos jogadores. Nesta fase, se algum dos jogadores escolher uma posição inválida para a primeira peça a retirar, a IGDH termina o jogo, com a indicação de que esse jogador perdeu.

Depois da fase inicial, o programa de controle deve entrar num ciclo em que pede a cada um dos jogadores para fazer uma jogada, começando pelo jogador que tem as peças pretas. O pedido de jogadas ao jogador humano é feito através da IGDH, e ao jogador automático através do envio de uma mensagem.

Após receber uma jogada de qualquer dos jogadores, o programa de controle deverá verificar a sua validade:

- se a jogada for válida, o programa de controle deve pedir à IGDH para actualizar o tabuleiro, com a execução da jogada;
- se a jogada não for válida, o programa de controle deve pedir à IGDH para assinalar o erro, com a indicação de que o jogador que tentou fazer a jogada perdeu;
- se a jogada de um jogador for nula,[1] indicando que esse jogador já não consegue fazer nenhuma jogada, o programa de controle deve pedir à IGDH para terminar o jogo, com a indicação de que o adversário do jogador ganhou.

2.2 O jogador automático

O jogador automático deve ser um programa que sabe jogar damas havaianas. Este programa pode ser tão inteligente quanto quiser. No entanto, para efeitos de classificação, qualquer jogador que respeite as regras do jogo e o protocolo de comunicação descrito na Secção 3.7 terá a classificação completa.

Os jogadores que obtenham a classificação completa poderão participar num torneio a realizar entre vários jogadores automáticos. As condições de participação neste torneio estão descritas na Secção 5.

2.3 A IGDH

A IGDH permite ao programa de controle usar uma interface gráfica para o jogo das damas havaianas. O programa de controle deve usar a IGDH para:

- Pedir ao jogador humano para escolher a cor das suas peças.
- Pedir a ambos os jogadores para escolherem as primeiras peças a retirar, validando a IGDH as posições destas peças.
- Aceitar uma jogada do jogador humano.
- Actualizar o tabuleiro da janela da interface com a execução de uma jogada.
- Escrever as mensagens adequadas quando o jogo termina.

[1] Recorda-se que uma jogada é considerada nula se a posição de início e de fim da jogada são iguais.

3 Protocolos de comunicação

Como acontece com muitos programas desenvolvidos no seio de empresas e de organizações, o seu programa terá de comunicar com outros programas. Para gerir a comunicação entre programas, é definido um protocolo de comunicação, vulgarmente conhecido por API (do inglês *Application Programming Interface*), que define de um modo exaustivo todas as interacções possíveis entre os programas, bem como o conteúdo da informação partilhada.

Um protocolo de comunicação é constituído, quer por tipos abstractos de informação, quer por procedimentos com nomes e parâmetros previamente acordados, o que garante que os vários programas podem invocar procedimentos uns dos outros.

Nesta secção apresentam-se os protocolos que o seu programa tem de seguir. Para além dos tipos e procedimentos definidos nesta secção, cada grupo é livre de definir os tipos de informação e procedimentos que desejar, apenas sujeitos às restrições definidas nesta secção.

Tenha em atenção que durante a avaliação o seu jogador automático vai ser usado por um programa que não é o seu programa de controle, e este vai usar um jogador de testes. Assim, terá de respeitar escrupulosamente os protocolos de comunicação definidos nesta secção. Se não o fizer, pode acontecer que embora o seu programa de controle e o seu jogador funcionem de uma forma correcta em conjunto, sejam ambos avaliados com zero.

3.1 Paradigmas de programação e de comunicação

3.1.1 Programa de controle e jogador automático

Tanto o programa de controle como o jogador automático correspondem a procedimentos com estado interno, com os quais a comunicação é feita através de mensagens. O envio de mensagens é feito pelo procedimento `em` (de envia mensagem):

```
(define (em proc mens . args)
   (apply (proc mens) args))
```

Esta definição deve fazer parte do seu código.

Nesta definição, `proc` é o procedimento com estado interno ao qual a mensagem é enviada, `mens` é o símbolo correspondente à mensagem, e `args` são os argumentos da mensagem (zero ou mais). Por exemplo:

```
(em jogador 'joga)
(em controle 'primeiras-pecas (faz-pos 0 0) (faz-pos 0 1))
```

Algumas destas mensagens devolvem um valor. Por exemplo, `(em jogador 'joga)` devolve uma posição.

A descrição detalhada das mensagens para o programa de controle e para o jogador automático é feita nas secções 3.5 e 3.7.

3.1.2 IGDH

A IGDH é constituída por um conjunto de procedimentos, alguns dos quais se destinam a ser usados pelo programa de controle. Assim, a comunicação com a IGDH é feita através da invocação destes procedimentos. Estes procedimentos são baseados em efeitos que dizem respeito à alteração da janela de interface e não devolvem nada.

A descrição detalhada destes procedimentos é feita na Secção 3.6.

3.2 Nomes

No seu programa *não podem ser definidos* nomes começados por `dh`. Também não podem ser definidos nomes correspondentes a procedimentos primitivos do Scheme; mesmo que o seu programa não utilize estes procedimentos, pode acontecer que o programa de avaliação automática o faça, provocando um conflito de nomes.

3.3 Código pré-definido a utilizar

O código abaixo deve ser incluído no ficheiro com o seu programa:

```
(define controle null)

(define (damas-havaianas)
    (set! controle (cria-controle)))
```

Para iniciar um novo jogo, deve ser invocado o procedimento `damas-havaianas`. Esta invocação não deve fazer parte do seu programa, isto é, a forma `(damas-havaianas)` não deve aparecer no nível de topo do seu programa, sob pena de falharem todos os testes feitos ao seu programa.

Para carregar a IGDH coloque a seguinte forma no seu programa:[2]

```
(require "interface-damas.zo")
```

3.4 Tipos abstractos de informação

Para além dos tipos pedidos para a primeira parte do projecto, deverá ainda definir o tipo mutável tabuleiro.[3]

Tal como na primeira parte do projecto, as operações básicas, excepto os construtores, não necessitam de verificar a validade dos argumentos que recebem.[4]

[2] O ficheiro `interface-damas.zo` deve estar na mesma directoria que o seu programa.

[3] Se desejar, pode adaptar a implementação do tipo matriz apresentada no Apêndice C do livro (solução do exercício 7.5)

[4] Como o único construtor pedido não recebe argumentos, nenhuma das operações abaixo descritas necessita de verificar a validade dos seus argumentos.

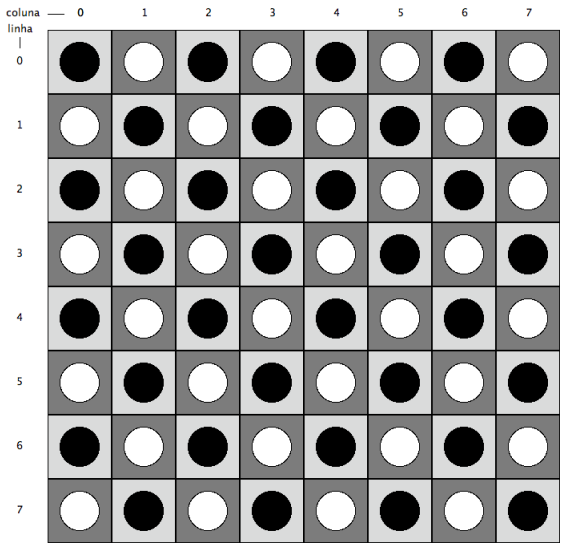

Figura 6: Tabuleiro inicial.

3.4.1 O tipo tabuleiro

Os elementos do tipo tabuleiro são matrizes com 8 linhas e 8 colunas, numeradas de 0 a 7, e representam um tabuleiro de jogo. Os elementos de um tabuleiro devem ser do tipo conteúdo. O tipo tabuleiro deve disponibilizar as seguintes operações básicas:

1. *Construtor*:
   - $tabuleiro\text{-}inicial : \{\} \mapsto tabuleiro$
     $tabuleiro\text{-}inicial()$ devolve um tabuleiro em que cada posição contém ou o símbolo *p* ou o símbolo *b*, correspondendo ao tabuleiro inicial do jogo. Neste, todas as posições estão preenchidas com uma peça: peças brancas nas casas pretas, e vice-versa. Na figura 6 mostra-se um tabuleiro inicial.
2. *Selectores*:
   - $conteudo\text{-}tabuleiro : tabuleiro \times posição \mapsto conteúdo$
     $conteudo\text{-}tabuleiro(t, p)$ devolve o conteúdo da posição p do tabuleiro t.
3. *Modificadores*:
   - $coloca\text{-}peca! : tabuleiro \times posição \times conteúdo \mapsto tabuleiro$
     $coloca\text{-}peca!(t, p, c)$ altera destrutivamente o tabuleiro t colocando c na posição p.
   - $retira\text{-}peca! : tabuleiro \times posição \mapsto tabuleiro$
     $retira\text{-}peca!(t, p)$ altera destrutivamente o tabuleiro t colocando o símbolo v na posição p.

3.5 Interacção com o programa de controle

O programa de controle deve corresponder a um procedimento com estado interno. Este procedimento deve ser criado através da invocação do procedimento `(cria-controle)`, sem argumentos. Ao ser criado o programa de controle deve ter início um novo jogo.

O procedimento correspondente ao programa de controle deverá ser o valor da variável de nome `controle`, que deverá ser acessível à IGDH, para que esta possa enviar mensagens ao programa de controle (ver Secção 3.3).

Na descrição que apresentamos, o símbolo não terminal correspondente ao argumento de uma mensagem indica o nome do tipo desse argumento.

Todas as mensagens enviadas ao programa de controle são-no pela IGDH, e estão divididas em 2 grupos: mensagens para a incialização do jogo, e mensagem durante o jogo. Nenhuma das mensagens enviadas ao programa de controle devolve nada.

As possíveis mensagens para o programa de controle são descritas de seguida:

3.5.1 Inicialização do jogo

Estas mensagens destinam-se a informar o programa de controle da cor das peças escolhida pelo jogador humano, e das posições das primeiras peças retiradas por ambos os jogadores.

`(em controle 'humano-escolheu-cor` <símbolo>**`)`** Esta mensagem é enviada ao programa de controle pela IGDH no início do jogo, e serve para informar o programa de controle da cor das peças escolhida pelo jogador humano. O argumento <símbolo>, um dos símbolos `p` ou `b`, indica a cor escolhida, preto ou branco, respectivamente.

`(em controle 'primeiras-pecas` <posição> <posição>**`)`** Esta mensagem é enviada ao programa de controle pela IGDH no início do jogo, e serve para informar o programa de controle das posições das primeiras peças retiradas pelos jogadores. O primeiro argumento diz respeito ao jogador humano, e o segundo ao jogador automático.

3.5.2 Jogo

`(em controle 'recebe-jogada-humano` <jogada>**`)`** Esta mensagem é enviada ao programa de controle pela IGDH durante o jogo, e serve para informar o programa de controle de uma jogada, <jogada>, feita pelo jogador humano. O programa de controle deve verificar a validade desta jogada.

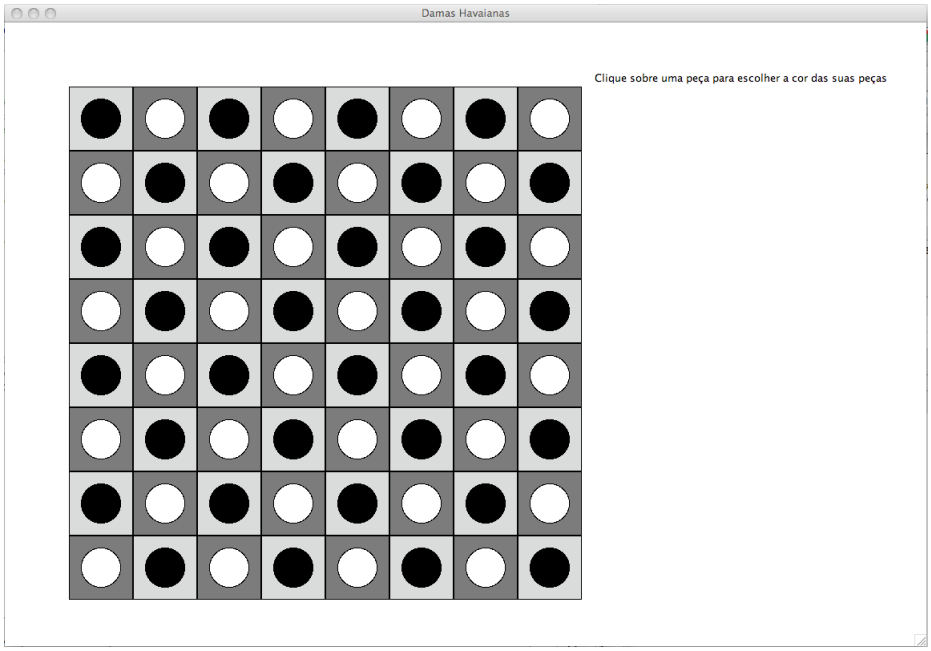

Figura 7: Tabuleiro inicial e pedido de cor ao utilizador.

3.6 Interacção com a IGDH

A comunicação do programa de controle com a IGDH é feita através da invocação de procedimentos. Estes procedimentos estão agrupados em 3 grupos: procedimentos para a inicialização do jogo, procedimentos usados durante o jogo, e procedimentos para terminar o jogo.

Na descrição que apresentamos, o símbolo não terminal correspondente ao argumento de um procedimento indica o nome do tipo desse argumento, à excepção de <jogador>, que representa o procedimento correspondente a um jogador automático.

3.6.1 Inicialização do jogo

Estes procedimentos são invocados pelo programa de controle com o fim de dar início a um jogo, permitindo o desenho do tabuleiro inicial na janela de interface, a obtenção da cor das peças do jogador humano, e a obtenção das primeiras peças retiradas por ambos os jogadores.

`(dh-inicio)` Este procedimento cria a janela de interface com a representação gráfica do tabuleiro inicial. Para além disto, pede ao utilizador para escolher a cor das peças com que quer jogar. Este procedimento não recebe argumentos, e termina enviando ao programa de controle a mensagem `humano-escolheu-cor` com um argumento que é um dos símbolos `p` ou `b`, com o significado óbvio. Na Figura 7 mostra-se a janela de interface nesta fase do jogo.

`(dh-pede-primeiras-pecas` <jogador>**`)`** Este procedimento pede, primeiro ao jogador que tem as peças pretas, e depois ao jogador que tem as peças brancas, para

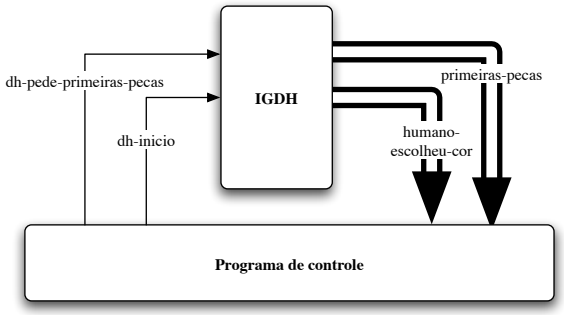

Figura 8: Interacção entre a IGDH e o programa de controle durante a fase de inicialização do jogo.

escolherem a primeira peça a retirar. O pedido ao jogador humano é feito através de uma mensagem escrita na janela de interface; o pedido ao jogador automático, o argumento <jogador>, é feito através do envio da mensagem `tira-primeira-peca`.

O procedimento termina de uma de duas formas possíveis:

- se um dos jogadores escolher uma posição errada para a primeira peça a retirar, o procedimento escreve uma mensagem dando esse jogador como perdedor;
- senão, o procedimento envia ao programa de controle a mensagem `primeiras-pecas` com dois argumentos: a posição da peça retirada pelo jogador humano, e a posição da peça retirada pelo jogador automático.

A interacção entre a IGDH e o programa de controle durante a fase de inicialização do jogo está representada na Figura 8. Nesta figura as setas simples representam a invocação de procedimentos, e as setas duplas representam o envio de mensagens.

3.6.2 Jogo

Estes procedimentos são invocados pelo programa de controle durante o jogo, permitindo a actualização do tabuleiro na janela de interface em resultado de uma jogada, e a obtenção das jogadas do jogador humano.

`(dh-executa-jogada` <jogada> <lista de posições>**`)`** Este procedimento altera o tabuleiro da interface gráfica, de forma a reflectir a execução da jogada <jogada>. O argumento <lista de posições> representa as posições das peças capturadas durante a jogada. Antes de executar a jogada, a IGDH assinala, durante alguns instantes, a jogada no tabuleiro, pintando a casa de início da jogada de verde claro, e a casa de fim da jogada de verde escuro, como se mostra na Figura 9.

Tenha em atenção que o procedimento `dh-executa-jogada` **não** verifica se a jogada é válida. Esta verificação é da responsabilidade do programa de controle, que só deve invocar este procedimento com jogadas válidas.

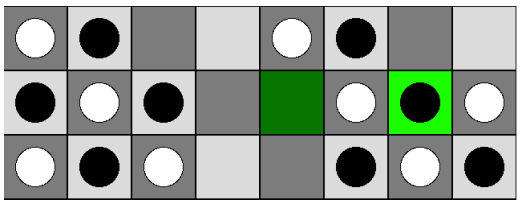

Figura 9: Jogada assinalada.

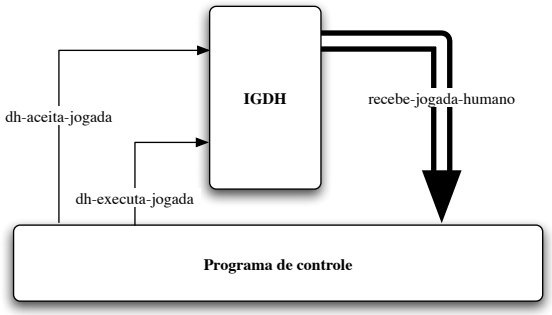

Figura 10: Interacção entre a IGDH e o programa de controle durante a fase de jogo.

(dh-aceita-jogada) Este procedimento aceita uma jogada do jogador humano, isto é, regista as posições de dois cliques do rato. Este procedimento não recebe argumentos, e termina enviando ao programa de controle a mensagem `recebe-jogada-humano` com um argumento que é a jogada feita pelo jogador humano. A verificação da validade da jogada deve ser feita pelo programa de controle. Quando o utilizador não encontrar nenhuma jogada para fazer, deve carregar duas vezes sobre uma posição qualquer do tabuleiro.

A interacção entre a IGDH e o programa de controle durante a fase de jogo está representada na Figura 10.

3.6.3 Fim de jogo

Estes procedimentos são invocados pelo programa de controle com o fim de terminar um jogo. O jogo pode terminar por duas razões distintas: porque um jogador não consegue fazer mais jogadas (neste caso, o adversário é declarado vencedor); porque um jogador tenta fazer uma jogada inválida (neste caso, o jogador é declarado perdedor).

(dh-termina-jogo <símbolo>**)** Este procedimento escreve uma mensagem na janela de interface anunciando o vencedor do jogo. Este procedimento é chamado pelo programa de controle quando um dos jogadores faz uma jogada nula, fazendo com que o adversário vença. O argumento <símbolo>, um dos símbolos `humano` ou `automatico`, indica o vencedor do jogo.

(dh-assinala-erro <símbolo> <jogada>**)** Este procedimento é chamado pelo programa de controle quando um dos jogadores faz uma jogada inválida, perdendo

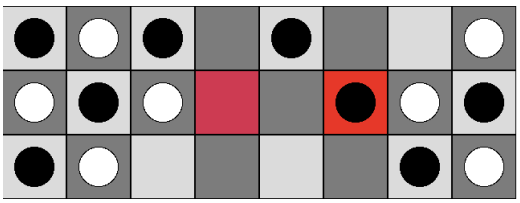

Figura 11: Jogada inválida assinalada.

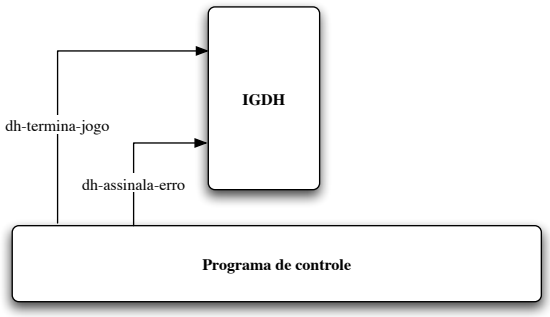

Figura 12: Interacção entre a IGDH e o programa de controle no fim do jogo.

consequentemente o jogo. O argumento <símbolo>, um dos símbolos `humano` ou `automatico`, indica o jogador que fez a jogada inválida, e o argumento <jogada> indica qual foi a jogada.

O procedimento `dh-assinala-erro` escreve uma mensagem indicando que o jogador perdeu, e assinala a jogada inválida, pintando a casa de início da jogada de vermelho, e casa de fim da jogada de vermelho escuro, como se mostra na Figura 11.

A interacção entre a IGDH e o programa de controle no fim do jogo está representada na Figura 12.

3.7 Interacção com o jogador automático

Um jogador deve corresponder a um procedimento com estado interno. O procedimento correspondente ao seu jogador deve ser criado através da invocação de um procedimento de nome `cria-jogador<número de grupo>`, em que `<número de grupo>` é a identificação do seu grupo.[5] Este procedimento recebe um argumento, um dos símbolos `p` ou `b`, e deve devolver um procedimento correspondente a um jogador automático, tal como descrito abaixo. O símbolo recebido pelo procedimento `cria-jogador<número de grupo>` indica qual a cor das peças do jogador automático. Por exemplo, as duas formas abaixo criam jogadores automáticos (do grupo 7) de peças brancas e pretas, respectivamente:

```
(cria-jogador7 'b)
(cria-jogador7 'p)
```

As possíveis mensagens para o jogador automático são descritas de seguida (<jogador> representa o procedimento correspondente a um jogador automático). Estas mensagens dividem-se em dois grupos: mensagens enviadas ao jogador pela IGDH durante a inicialização do jogo, e mensagens enviadas ao jogador pelo programa de controle durante o jogo.

3.7.1 Inicialização do jogo

Estas mensagens são enviadas ao jogador automático pela IGDH durante a inicialização do jogo, destinando-se a obter a posição da primeira pela retirada pelo jogador automático, e a informar este jogador da posição da primeira pela retirada pelo jogador humano.

(em <jogador> **'tira-primeira-peca [**<posição>**])** Esta mensagem é enviada pela IGDH, no início do jogo, para obter a primeira jogada do jogador automático. Tem um argumento opcional,<posição>, e devolve uma posição, que indica a posição da peça que o jogador automático quer tirar para começar o jogo.

- Se o argumento <posição> estiver presente, isso significa que o jogador automático tem as peças brancas, e que o jogador humano escolheu tirar a peça da posição <posição> para começar o jogo. De acordo com as regras do jogo, nesta situação o jogador automático deve devolver uma posição do tabuleiro, adjacente a <posição>.
- Se o argumento <posição> não estiver presente, isso significa que o jogador automático tem as peças pretas, e, consequentemente, é o primeiro a tirar uma peça. De acordo com as regras do jogo, nesta situação o jogador automático deve devolver uma das posições `((0,0)(3,3)(4,4)(7,7))`.

(em <jogador> **'humano-tirou** <posição>**)** Esta mensagem é enviada pela IGDH para indicar ao jogador automático a posição da primeira peça retirada pelo jogador humano, quando este tem as peças brancas. Não devolve nada; ao receber esta mensagem o jogador deve actualizar a informação que tem sobre o jogo.

[5] Se o número de grupo for inferior a 10, escreva apenas um dígito, por exemplo, `cria-jogador3`, e não `cria-jogador03`.

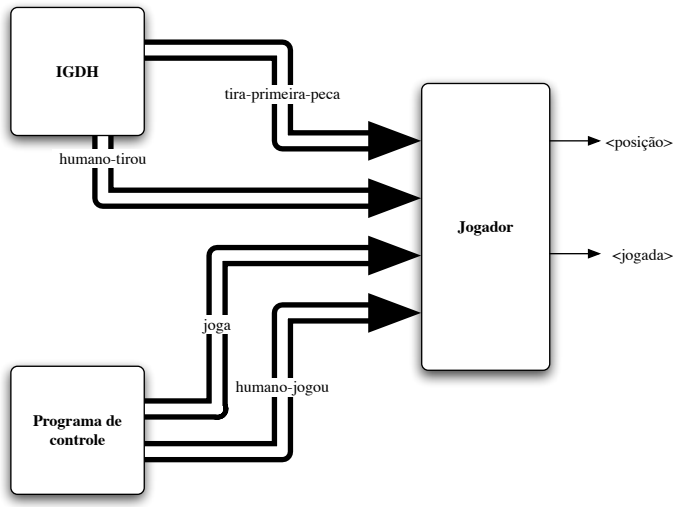

Figura 13: Mensagens para o jogador e valores devolvidos.

3.7.2 Jogo

Estas mensagens são enviadas ao jogador automático pelo programa de controle durante o jogo, destinando-se a obter as jogadas do jogador automático e a informar este jogador das jogadas feitas pelo jogador humano.

(em <jogador> **'joga)** Esta mensagem é enviada pelo programa de controle, no decurso do jogo, para obter a jogada do <jogador>. Não tem argumentos e devolve uma jogada, correspondente à jogada que o jogador automático quer fazer. Esta jogada deve ser válida, de acordo com as regras do jogo, caso contrário o jogador automático perde imediatamente o jogo. No caso de o jogador automático não poder fazer mais jogadas, deve devolver uma jogada nula, isto é, uma jogada em que a posição de início e a de fim são iguais.

(em <jogador> **'humano-jogou** <jogada>**)** Esta mensagem é enviada pelo programa de controle para informar o jogador automático da última jogada feita pelo jogador humano. Não devolve nada; ao receber esta mensagem o jogador deve actualizar a informação que tem sobre o jogo.

Na Figura 13 apresenta-se um resumo das mensagens recebidas pelo jogador, bem como o tipo dos valores devolvidos, quando for caso disso.

4 Sugestão de passos a seguir

Desenvolver um programa com a complexidade descrita pode parecer uma actividade assustadora. Nesta secção apresentamos algumas sugestões para os passos a seguir no desenvolvimento do programa.

1. Assegure-se que tem todos os tipos de informação requeridos na primeira parte do projecto. Aconselha-se vivamente a utilização do código disponibilizado pelo corpo docente.

2. Implemente o tipo tabuleiro, tal como definido na Secção 3.4.1. Embora não seja pedido, será útil durante a fase de testes dispor de um tranformador de saída, que lhe permita inspeccionar o conteúdo do tabuleiro sem recorrer à IGDH. Não passe para o próximo passo sem se certificar que o tipo tabuleiro está correctamente implementado. Sendo este um tipo fundamental usado por todos os módulos do programa, não será possível testar nenhum dos módulos enquanto o tipo tabuleiro não estiver correctamente implementado.

3. Comece a pensar no desenvolvimento do programa que corresponde ao jogador. O jogador corresponde a um procedimento com estado interno. Decida qual a informação que deve corresponder a este estado interno. O seu jogador deve estar preparado para receber as mensagens apresentadas na Secção 3.7 e produzir os comportamentos indicados. Escreva cada um destes comportamentos, um de cada vez, testando cuidadosamente o comportamento do seu jogador através da simulação de envio de cada mensagem. Não tente, nesta fase, que o seu jogador seja minimamente inteligente, mas apenas que jogue. Poderá querer utilizar um procedimento primitivo que, corresponde a um procedimento com estado interno, existente em Scheme, `random`, que recebe um argumento inteiro, n, e que devolve aleatoriamente um inteiro no intervalo de 0 a $n-1$. O facto de random ser um procedimento com estado interno significa que duas avaliações seguidas de random com o mesmo argumento devolvem valores diferentes. O jogador automático poderá utilizar este procedimento para escolher a primeira peça a tirar.

4. Não comece esta fase sem ter os tipos de informação e o jogador automático implementados e testados. Familiarize-se com a IGDH, enviando, manualmente as várias mensagens possíveis para o programa que corresponde à interface. Como as respostas a algumas destas mensagens correspondem a mensagens enviadas ao programa de controle (ver Secção 3.6), sugerimos que use o seguinte código:

```
(define (cria-controle)

(define (humano-escolheu-cor cor)
   (display "controle: mensagem humano-escolheu-cor ")
   (display cor)
   (newline))

(define (primeiras-pecas pos-humano pos-automatico)
   (display "controle: mensagem primeiras-pecas ")
   (display pos-humano)
   (display " ")
   (display pos-automatico)
   (newline))

(define (recebe-jogada-humano jogada)
   (display "controle: mensagem recebe-jogada-humano ")
   (display jogada)
   (newline))
```

```
(lambda (m)
   (case m
      ((humano-escolheu-cor) humano-escolheu-cor)
      ((primeiras-pecas) primeiras-pecas)
      ((recebe-jogada-humano) recebe-jogada-humano))))
```

A partir deste momento pode enviar mensagens à IGDH, como o faria o programa de controle, e ver o que acontece tanto em termos da janela de interface, como das mensagens enviadas ao programa de controle. Deverá começar por avaliar as formas:

```
(damas-havaianas)
(dh-inicio)
```

Poderá mesmo controlar um jogo completo, desempenhando o papel do programa de controle.

5. Depois de já ter uma boa ideia do funcionamento dos procedimentos da IGDH e das mensagens enviadas por esta ao programa de controle, comece a desenvolver o programa de controle. Já tem o jogador a trabalhar, já percebeu como interaccionar com a interface, agora é apenas questão de escrever um programa que interage adequadamente com os outros dois.

6. Quando tiver concluído o passo anterior, faça uma cópia dos seu programa e não a modifique mais. Esta poderá ser, em último caso, a versão final do seu programa se o próximo passo não correr bem. Finalize também neste ponto toda a documentação do seu projecto. Verifique quais os aspectos que vão ser considerados na classificação (descritos na Secção 6) e veja como os pode maximizar. O seu trabalho deverá estar pronto para ser entregue e poderá obter a classificação máxima.

7. Se desejar dotar o seu programa de alguma inteligência, tentando ganhar alguns valores adicionais como resultado do torneio, comece agora a pensar no raciocínio que deverá ser seguido pelo jogador automático. Não se esqueça, não altere o programa que já fez e que deverá estar cuidadosamente guardado num ficheiro no seu computador.

5 Torneio

Será disputado um torneio de damas havaianas entre alguns dos jogadores automáticos. As regras detalhadas deste torneio serão publicadas na página da cadeira. Os quatro melhores jogadores (aqueles que forem seleccionados para as meias finais) serão recompensados com as seguintes notas adicionais à nota do projecto:

1. Primeiro classificado: 2,0 valores;

2. Segundo classificado: 1,5 valores;

3. Terceiro classificado: 1,0 valor;

4. Quarto classificado: 0,5 valores.

Para poder participar no torneio um jogador terá de satisfazer as seguintes condições:

1. O jogador tem de respeitar as regras do jogo:
   - Retirar a primeira peça de uma posição válida, qualquer que seja a cor das suas peças.
   - Fazer jogadas válidas, enquanto elas existirem.
   - Devolver uma jogada nula, quando não existirem mais jogadas válidas.
2. O jogador tem de respeitar o protocolo de comunicação descrito na Secção 3.7.
3. O procedimento de criação do jogador tem de conter definidos internamente todos os nomes que utiliza, à excepção das operações básicas e das operações de alto nível definidas nos enunciados.[6]

6 Classificação

A nota da segunda parte do projecto será baseada nos seguintes aspectos:

1. Execução correcta (40%). Esta parte da avaliação é feita recorrendo a um programa de avaliação automática que não só sugere uma nota face aos vários aspectos considerados, como também decide se o seu jogador automático é apurado ou não para o torneio.
2. Facilidade de leitura, nomeadamente abstracção procedimental, abstracção de dados, nomes bem escolhidos, paragrafação correcta, qualidade (e não quantidade) dos comentários[7] e tamanho dos procedimentos (25%).
3. Relatório (30%).
4. Estilo de programação (5%).

Os pesos das notas das duas partes na nota final do projecto são os seguintes:

1. Primeira parte - 30%.
2. Segunda parte - 70%.

[6] Esta condição pode levar à repetição de código, mas é essencial para participação no torneio.

[7] Tal como na primeira parte do projecto, todos os comentários do seu programa devem ser feitos utilizando a opção “Comment Out with Semicolons". Programas que utilizem a opção ‘Comment Out with a Box” serão penalizados com três valores.

7 Condições de realização e prazos

O projecto deve ser realizado pelos mesmos grupos da primeira parte.

A segunda parte do projecto deve ser entregue até às 15:00 horas do dia **3 de Janeiro de 2011**, na Reprografia do DEI ou na portaria do Tagus (consoante o campus que corresponda ao grupo do projecto), e deverá constar de um relatório, incluindo uma listagem do código. Um modelo de relatório, que podem/devem adaptar de acordo com as vossas necessidades, será disponibilizado no sistema Fénix. Projectos em atraso serão aceites durante a semana de 3 a 7 de Janeiro, sendo penalizados com 0,5 valores por cada dia de atraso. A partir das 15:00 horas do dia 7 de Janeiro de 2011 não se aceitam quaisquer projectos, seja qual for o pretexto. O relatório deve ser entregue dentro de uma capa ou encadernado, apresentando visivelmente o número do grupo e número e nome dos seus autores, na capa. Projectos que não sejam entregues nestas condições serão penalizados com três valores.

Para além disto, a submissão do código por via electrónica, *através do sistema Fénix*, é *obrigatória* e deverá ser feita nos mesmos prazos que a entrega do relatório. Se o código e o relatório forem entregues em dias diferentes, o desconto será o correspondente ao dia da última entrega. O código do projecto deve estar contido num único ficheiro. Se durante o desenvolvimento usaram vários ficheiros devem, no final, antes de submeter o código, colocar todo o código num único ficheiro, chamado `FP1011-parte2-grupo`n`.scm`, em que n é o número do grupo. Por exemplo, o ficheiro do grupo nº 5 deverá chamar-se `FP1011-parte2-grupo5.scm`. O ficheiro de código deve conter em comentário, na primeira linha, os números e os nomes dos alunos do grupo, bem como o número do grupo.

Durante o desenvolvimento do programa é importante não se esquecer da *Lei de Murphy*:

1. Todos os problemas são mais difíceis do que parecem;
2. Tudo demora mais tempo do que nós pensamos;
3. Se alguma coisa puder correr mal, ela vai correr mal, na pior das alturas possíveis.

Pode ou não haver uma discussão oral do trabalho e/ou uma demonstração do funcionamento do programa (será decidido caso a caso).

Projectos iguais, ou muito semelhantes, serão considerados cópias. O corpo docente da cadeira será o único juiz do que se considera ou não copiar no projecto.