Jogo de Tabuleiro: Quinamid

Relatório Intercalar

Universidade do Porto

Faculdade de Engenharia

FEUP

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo 29:
Luis Guilherme Ribeiro de Castro Silva Martins - ei10105
Rui Tiago Bugalho Monteiro - ei10086

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

5 de Novembro de 2012

Resumo

Neste projecto, será desenvolvido o jogo de tabuleiro *Quinamid*.

Prolog será a linguagem de implementação utilizada para o seu desenvolvimento e, na sua build final, a aplicação deverá permitir ao utilizador escolher entre jogar contra outro **utilizador humano**, enfrentar o **computador** ou assistir a uma partida entre dois jogadores dotados de **inteligência artificial**.

O programa deverá ainda ser dotado de um módulo de visualização gráfica a 3 dimensões, que será, no final, implementada com o recurso à linguagem de programação *c++*.

Conteúdo

1 Introdução

Este trabalho apresenta vários desafios para o grupo.

A sua implementação é feita com uma linguagem de programação com diferenças significativas em relação à maioria das linguagens mais conhecidas. *Prolog* apresenta uma sintaxe fora do comum e, por ser uma **linguagem declarativa**, ao invés de se determinar a solução para um problema seguindo os passos definidos, numa folha de código, pelo programador (como acontece na maioria das linguagens de programação), as soluções são obtidas através de **deduções e de relações lógicas** entre factos e relações declarados, *à priori*, numa base de dados. Por essa razão, programar em *Prolog* é diferente, em todos os aspectos fundamentais, de programar em *c++*, ou em *java*, ou em qualquer outra linguagem de **programação prodecimental**.

Por outro lado, o *Quinamid* é também uma novidade para ambos os elementos do grupo. Fazer a sua implementação implica conhecer, compreender e ganhar uma substancial familiaridade com o jogo.

Dos desafios surgem por necessidade os objectivos e as motivações.

Para este trabalho, **aprofundar conhecimentos** sobre a linguagem de programação *Prolog*, ganhar sensibilidade para a resolução de problemas recorrendo a esta como ferramenta de trabalho e aprender mais sobre os **mecanismos e as estratégias** de jogo do *Quinamid*, acabam por ser os **principais objectivos**.

As **motivações** aliam-se ao facto de tudo acima referido ser ainda uma novidade e de, em acréscimo, o *Quinamid* ser um jogo de tabuleiro com algumas características bastante interessantes, que merecem ser destacadas:

- é um jogo de tabuleiro constituído, curiosamente, por **5 tabuleiros**;
- 4 dos 5 dos seus tabuleiros podem ser **deslocados** ou **rodados** em relação ao tabuleiro maior;
- é possível que **peças** colocadas em jogadas anteriores fiquem **sobrepostas**, durante algumas jogadas, pelos tabuleiros que lhes são sobrejacentes, obrigando os jogadores a memorizar as suas posições.

2 Descrição do Problema

O *Quinamid* foi criado por *Antony Brown* e começou a ser comercializado pela *Third Dynasty Games*, em 2007. Foi considerado um dos **melhores jogos** de tabuleiro, criados nesse ano, por *Tom Vasel*, um avaliador de nome conhecido na indústria dos jogos de tabuleiro [1,3].

Figura 1: Tabuleiro de *Quinamid*.

O *Quinamid* distingue-se dos demais jogos de tabuleiro por possuir **5 tabuleiros**.

O objectivo do jogo é conseguir fazer uma **linha de 5 peças** da mesma cor. Para isso, cada jogador pode, em cada jogada, colocar uma nova peça, rodar ou deslocar um dos 4 tabuleiros que se encontram dispostos em forma de **pirâmide** sobre o maior, com 6 casas de lado. A linha pode ser feita na horizontal, na verticar ou na diagonal.

Regras do Jogo

Apresentação e Preparação

- 5 tabuleiros com 6, 5, 4, 3 e 2 espaços de lado;
- 60 peças de duas cores, 30 de cada uma.

Desenvolvimento de uma Partida

1. os tabuleiros são dispostos uns sobre os outros, do maior até ao de menor lado;
2. na primeira jogada, é aleatoriamente escolhido um jogador para jogar;
3. o jogador a jogar escolhe um dos seguintes movimentos a fazer:
   - jogar uma peça num buraco vazio e exposto de um dos tabuleiros;
   - mover um dos tabuleiros ortogonalmente (cima, baixo, direita, esquerda);
   - rodar um dos tabuleiros 90^{o}, para um lado à escolha.

   Nota: nenhum dos movimentos escolhidos pode resultar numa jogada que anule a imediatamente anterior do jogador oponente.
4. no caso de ainda ser a primeira jogada, com base no movimento tomado, o jogador oponente pode escolher trocar de peças, por motivos estratégicos; neste caso, o jogador que jogou primeiro volta a jogar, repetindo o passo anterior;
5. repetir o passo 3, alternando de jogador a cada turno, até que um dos jogadores faça uma linha.

Notas adicionais:

- quando um tabuleiro é sobreposto, todas as suas peças mantém as posições, de maneira a que o jogo continue igual se, num outro momento do jogo, volte a ser descoberto;
- após cada rotação, o tabuleiro rodado deve sobrepor exatamente os mesmos buracos que sobrepunha antes do movimento;
- um tabuleiro que se encontre numa camada superior deve estar sempre em contacto directo com o imediatamente subjacente.

O jogo acaba assim que um jogador fizer uma linha, **vencendo o jogo.**

3 Arquitectura do Sistema

Dada a complexidade do projecto, decidiu-se separar o sistema em **vários módulos**, de forma a simplificar a implementação, a legibilidade e a organização do código tornando mais fácil a correcção de possíveis bugs que possam surgir.

Sendo assim, o sistema foi dividido em **3 partes**:

1ª - A parte das regras e validação de jogadas.
Este módulo serve para implementar as **regras do jogo**, verificar se determinado movimento é possível executar naquele momento e ainda verificar se algum jogador vence a partida terminando o jogo.

2ª - A parte da representação e interface com o utilizador.
Esta secção é responsável pela **interacção máquina-utilizador**, recebendo inputs para determinar as jogadas pretendidas pelo visualizador e representa o tabuleiro (resultante da montagem dos vários tabuleiros do jogo) com as peças colocadas e possíveis rotações e translações dos tabuleiros.

3ª - A parte da inteligência artificial.
Nesta parte é implementada a **inteligência artificial** responsável pelas jogadas do computador. Aqui se desenvolvem os algoritmos que permitem ao "computador" escolher a melhor jogada possível de forma a vencer o adversário.

4 Módulo de Lógica do Jogo

Como descrito no ponto anterior deste relatório, a lógica da presente aplicação reside sobre 3 partes fundamentais - a validação das regras do jogo, o suporte à interface tridimensional dos tabuleiros do Quinamid e os mecanismos de inteligência artificial.
De uma forma sucinta, um jogador interage directamente com o sistema de validação, a primeira parte, e essa por sua vez vai interagir com os restantes mecanismos, por via dos predicados explicados a seguir:

4.1 Representação do Estado do Jogo

tabuleiro(-Tabuleiro).
retorna a matriz do tabuleiro inicial de jogo;

4.2 Visualização do Estado do Jogo

mostra_tabuleiro(+Tab, +LimH, +LimV).
recebe o tabuleiro de jogo e as matrizes que guardam os limites horizontais e verticais dos tabuleiros mais pequenos, por forma a desenhar um estado de jogo;

4.3 Validação de Jogadas

a aplicação retorna um erro caso seja seja introduzida uma jogada não válida;

4.4 Execução de Jogadas

jogar_peca(+Linha, +Coluna, +Jogador, +Tabuleiro, -NovoTabuleiro).
retorna um novo estado de jogo, para uma jogada feita por um dado jogador sobre uma dada cordenada do tabuleiro;

4.5 Final do Jogo

verifica_vitoria(+Tabuleiro, +Jogador).
caso haja uma situação de vitória, envia uma mensagem informando qual dos jogadores venceu o jogo e fecha a aplicação;

5 Interface com o Utilizador

O módulo de interface com o utilizador deste projecto requer um trabalho bastante elaborado com listas porque, uma vez que existem **vários tabuleiros** em jogo, existem **várias matrizes** com informação a tratar.

A solução encontrada foi usar **uma matriz para cada tabuleiro** para guardar a informação correspondente e uma matriz "geral" que fosse o resultado do "empilhamento" dos tabuleiros. Esta matriz seria a **matriz de visualização** e seria equivalente à pirâmide de tabuleiros vista de cima, proporcionando uma experiência de jogo semelhante à original e ajudando a discernir qual a posição relativa dos tabuleiros.

Desta forma, o utilizador nunca vai poder alterar directamente essa matriz, a não ser que mova um dos tabuleiros, pois trata-se de uma **junção de várias matrizes**. Se o utilizador pretender adicionar colocar uma peça no tabuleiro ou rodá-lo, vai **alterar a matriz correspondente** ao mesmo, guardando assim toda a informação que a dada altura pode ficar oculta por outro tabuleiro. Uma vez feita a alteração, é **gerada uma nova matriz de visualização** que se mantém actualizada a cada jogada.

6 Conclusões e Perspectivas de Desenvolvimento

Uma vez finalizado o projecto, pode-se concluir que ficou **aquém das expectativas**. Das 3 partes em que se dividia o trabalho, **apenas uma** (regras e validação de jogadas) ficou completamente funcional.

A parte da representação e interface revelou-se bastante **mais difícil do que era esperado**, nomeadamente devido ao uso de **múltiplas matrizes** para armazenar a informação, mas tendo-nos levado o tempo que estava planeado para a inteligência artificial.

Parte do insucesso deve-se à **má gestão do tempo** e à sobrecarga excessiva de projectos a desenvolver no período de entrega deste trabalho. Para além disso, a total **inexperiência na linguagem** de programação utilizada para este projecto foi também um factor decisivo para o mau desempenho do grupo.

7 Bibliografia

[1] Vários autores, Board Game Geek, disponível online a partir de http://boardgamegeek.com/boardgame/29952/quinamid (Consultado em Outubro de 2012).

[2] Vários autores, BoardSpace, disponível online a partir de http://www.boardspace.net/english/about_quinamid.html (Consultado em Outubro de 2012).

[3] Vários autores, Quinamid, disponível online a partir de http://www.quinamid.com (Consultado em Outubro de 2012).