Projeto de Laboratórios de Informática 1
Grupo 81

Pedro Gonçalves (A82313) Rodolfo Vieira (A81716)

2 de Janeiro de 2017

Resumo

Neste relatório encontra-se descrito o processo de criação do jogo Bomberman, utilizando a linguagem de programação funcional Haskell, desde a parte da criação do jogo em si, até à parte da criação da interface gráfica e da programação de um Bot.

Conteúdo

1 Introdução **3**

2 Descrição do Problema **4**

3 Concepção da Solução **5**

3.1 Estruturas de Dados 5

3.2 Implementação 5

3.3 Testes 7

4 Conclusões **8**

1 Introdução

No âmbito da disciplina de Laboratórios de Informática 1 (LI1), foi-nos proposto a realização de um projeto que consiste na recriação, através da linguagem de programação funcional Haskell, do clássico das consolas Bomberman. Aquando da apresentação do enunciado do projeto, o grupo sentiu-se um pouco assustado, pelo simples facto de não ter experiência em programação, mas sentiu-se igualmente motivado particularmente por dois motivos:

1. A possibilidade de melhoramento das nossas capacidades como programadores;
2. O facto de que no final do projeto seríamos capazes de disfrutar do fruto do nosso trabalho.

2 Descrição do Problema

Os problemas apresentados eram:

1. A criação de um mapa, gerado a partir de uma semente e de uma dimensão. Nesse mesmo mapa deveriam estar listados os *Power-Ups* (!,+), as *Pedras* (#), os *Tijolos* (?), tijolos esses que podem ser destruídos, e também os *Espaços Vazios* ( ), nos quais o jogador se pode movimentar.

2. O movimento do jogador, que envolvia as 5 diferentes ações que este poderia realizar.

   (a) Mover-se para cima (U);

   (b) Mover-se para baixo (D);

   (c) Mover-se para a direita (R);

   (d) Mover-se para a esquerda (L);

   (e) Colocar uma Bomba (B).

   Era também pedido que fosse criado um sistema que ao analizar o mapa, diria se era possível executar um determinado comando.

3. O processo de guardar e continuar um jogo já começado previamente, através da implementação de um sistema de compressão/descompressão desse mesmo estado de jogo ou seja, ao comprimir, retirar certos caractéres que não são vitais ao Estado de Jogo de maneira a que o ficheiro que contenha o Estado de Jogo ocupe um menor espaço (tenha menos caractéres) que o original, em quanto que ao descomprimir, os caractéres previamente retirados são inseridos de novo no Estado de Jogo.

4. A passagem do tempo, que consiste em dois processos:

   (a) Efeito da passagem de tempo no Estado de Jogo, ou seja programar o que acontecia com o passar de tempo (Explosão de Bombas, eliminações de Jogadores e de Power-Ups e destruição de Blocos);

   (b) Criação de uma *Espiral* que elimina do Estado de Jogo tudo o que esteja no seu caminho.

5. A parte gráfica, que consiste na criação de uma interface gráfica de todo os problemas enumerados anteriormente.

   Esta interface permite que o jogo seja jogado, pois representa graficamente todas as possibilidades: (Movimentos dos Jogadores, Colocação de Bombas, Passagem de Tempo, Explosões das Bombas e também a criação da Espiral).

6. O Bot, consiste na criação de uma estratégia de jogo onde através da análise de um Estado de Jogo, é executado um determinado comando, tendo como objetivo a criação de um Jogador autónomo que consiga lutar contra outros Jogadores sem a interferência de um utilizador.

3 Concepção da Solução

Esta secção descreve o trabalho desenvolvido com vista a resolveros problema apresentados na secção anterior.

3.1 Estruturas de Dados

As principais Estruturas de Dados utilizadas durante este a elaboração deste projeto foram:

1. Para representar o Estado de Jogo foram utilizadas [Strings];
2. Para representar o Estado, na execução da Tarefa 5,definiu-se estado como
   ((Float,Float),Float,[String],[Picture],Float)

3.2 Implementação

As soluções encontradas aos problemas a que fomos sujeitos foram:

1. Para a criação do mapa foi usado um gerador aleatório, que através do fornecimento de uma semente gera um número de 0 a 99, correspondendo cada número a um certo tipo de bloco.
   (a) Se o número pertencer de 0 a 1, então é representado um Power-Up Bombs atrás de um Tijolo;
   (b) Se o número pertencer de 2 a 3, então é representado um Power-Up Flames atrás de um Tijolo;
   (c) Se o número pertencer de 4 a 39, então é representado um Tijolo;
   (d) Se o número pertencer de 40 a 99, então é representado um Espaço Vazio.

   De seguida foi usada uma função que através do fornecimento da dimensão do mapa, gera um mapa que consiste de blocos com a sua posição já predifinida em qualquer tipo de mapa. Sendo depois os caracteres obtidos no gerador integrados na carcaça do mapa, obtendo-se assim um mapa completo, onde so falta a listagem dos power ups, que por sua vez são originários do gerador aleatório previamente mencionado, que vai passar por um conjunto de diversas funções que decifram a sua posição relativa no mapa, e vão ordenalos numa lista que é mais tarde integrada no mapa ja quase completo, obtendo-se assim o produto final.
2. No movimento do Jogador, foi utilizado um conjunto diverso de funções que, após o fornecimento de um *Estado de Jogo*, de um *Comando* que se pretende que seja efetuado e do *Número identificativo do Jogador*, analisar a posição do Jogador no Estado de Jogo, verificando depois a partir do Comando fornecido qual o objeto que se encontra na posição para a qual um Jogador se pretende mover. Se esse objeto for uma *Pedra* (#), ou um *Tijolo* (?), o Jogador é impedido de executar essa ação, mas se no entanto esse objeto se tratar de um *Espaço Vazio* ( ), a acão que o Jogador pretendia relaizar é executada.

   Se o Comando que se pretenda executar for a colocação de uma Bomba, um conjunto de funções vão ser executadas com os seguintes objetivos:

   (a) Verifica se um dado jogador pode colocar Bombas, com base em certos fatores:
      i. Se o número de Bombas que um determinado Jogador possuí for igual ou superior ao Número de *Power-Ups Bomba* (+), ou se já existir uma Bomba na posição onde se pretende colocar a nossa Bomba,o Jogador não vai ser capaz de colocar uma Bomba.
      ii. Se o Jogador puder colocar Bombas,vai ser verificado o Número de *Power-Ups Flame* (!) que o Jogador possui, com o objetivo de determinar o raio da Bomba que será colocada, sendo depois criada uma *String* que possui as informações sobre a Bomba: (Posição onde foi colocada; Identificador do Jogador que a colocou; Raio

de Explosão; Tempo até à Explosão da Bomba), *String* essa que vai ser adicionada à *[String]* do Estado de Jogo, logo após das *String* com as informações sobre os *Power-Ups*.

3. Na Compressão do Estado de Jogo, é utilizada uma função cujo objetivo é eliminar os blocos pré-definidos do Estado de Jogo (criados na Tarefa 1), diminuindo assim o número de caractéres utilizados e consequentemente o tamanho do Estado de Jogo.

   Na Descompressão, é utilizada uma função que adiciona os blocos pré-definidos novamente ao Estado de Jogo.

4. Na passagem do tempo, é fornecida à funçâo principal um Estado de Jogo e também o tempo restante até ao final do jogo. De seguida, esta função analisa o Estado de Jogo e faz com que ocorra a passagem de uma Unidade de Tempo. Caso esta Passagem de Tempo faça com que uma bomba exploda, a função analisa as alterações que a explosão vai gerar no Estado de Jogo, ou seja, vai analisar todos os objetos que se encontram dentro do raio de explosão da bomba e vai verificar se estes serão destruidos ou não:

   (a) Se o objeto for uma Pedra, a função faz com que o Raio de Explosão da Bomba não ultrapasse a Pedra e faz também com que a Pedra não seja destruída;

   (b) Se o objeto for um Espaço Vazio a função vai fazer com que o raio da Bomba continue a propagar-se;

   (c) Se o objeto for um Tijolo a função vai destruir esse mesmo Tijolo e substitui-o por um Espaço Vazio, parando assim a propagação da Bomba;

   (d) Se num dos Espaços Vazios em cima mencionados, a Bomba se depare com um Power-Up "destapado", o seu Raio de Explosão pára de se propagar, e consequentemente elimina esse mesmo Power-Up do Estado de Jogo;

   (e) Se exista um Jogador num dos Espaços Vazios afetados pelo raio da bomba, este será eliminado do Estado de Jogo, mas neste caso o raio da Bomba continua a propagar-se;

   (f) Se num dos espaços vazios exista uma Bomba, o tempo até à Explosão desta, vai ser reduzido para 1 Unidade de Tempo.

5. Na criação da Interface Gráfica, são utilizadas diversas funções com o objetivo de representar gráficamente um determinado Estado (O tipo Estado é definido previamente):

   (a) Uma função que desenha um Estado de Jogo fazendo o Translate de certas imagens para determinadas posições, conforme as informações fornecidas pelo tipo *Estado* (Dimensão e [String] com informações sobre o Estado de Jogo);

   (b) Uma função que faz com que ocorra a passagem de tempo, permitindo assim que ocorram todos os fenómenos que dependam da Passagem de Tempo;

   (c) Uma função que define o Frame Rate do Jogo;

   (d) E por fim uma função que permite que o Jogador execute Comandos, e que provoque as alterações do Estado de Jogo ocorridas devido à execução desse Comando.

6. Na criação do Bot, um conjunto diverso de funções foi criado com o objetivo de analisar o Estado de Jogo. O Bot está programado para perseguir o Tijolo mais próximo e quando chega perto deste, coloca uma Bomba para o destruir, repetindo este processo múltiplas vezes, mas antes de cada movimento ser executado , é analisado o lugar para o qual o bot se pretende mover:

   (a) Se a posição para a qual o Bot se pretende mover estiver no Raio de uma Bomba, o movimento é cancelado e o Bot permanece na mesma posição (Se antes do movimento ser executado o bot já se encontrar no Raio de Explosão de uma Bomba, o Bot irá deslocar-se para uma posição onde não seja afetado por essa mesma Bomba);

(b) Se na posição para a qual o Bot se pretenda mover exista uma pedra, o Bot irá mudar de estratégia com vista a não se tentar deslocar para uma posição que contenha uma pedra.

(c) Se a Espiral já se estiver a formar, o Bot adota uma estratégia defensiva, cujo principal objetivo é dirigir-se para o centro do mapa de maneira a ser o último Bot a morrer (Neste processo, continuam a ser executadas as funções em cima anunciadas).

3.3 Testes

1. (["##########","# #","# #?# # #","# ? #","#?# # #?#","# ? ? #","# #?#?# #","# ?? #","##########"])

   O teste acima representado foi utilizado em funções que apenas necessitavam do mapa em si e não das informações adicionais, tendo sido mais utilizada na criação do Bot, pois a criação precisava constantemente de ser testada, para ver se o Bot reagia da maneira que seria esperado.

2. (["##########","# #","# #?# # #","# ? #","#?# # #?#","# ? ? #","# #?#?# #","# ?? #","##########","+ 3 3","! 5 5","* 7 7 1 1 10","0 4 3 +","1 7 7"])

   Este teste foi o mais utilizad pois tem uma dimensão favorável a alterações, sendo muito útil para testar cenários diferentes mais facilmente.

3. (["################","# ? ?? ? #","# #?# #?#?# # #","# ? ?? ?? ? #","# #?# # #?#?# #","#? ? ??? ? ?#","#?#?#?#?# #?#?#","# ? ???? ?#","# # # #?#?# #","# ?? ? ??#","# # # # #?#?# #","#? ?? ? #","# #?#?# #?# # #","# ? ????? ? #","################","+ 8 3","+ 12 3","+ 7 6","+ 12 9","+ 5 12","! 3 5","! 13 6","! 7 7","* 1 2 0 2 10","* 13 2 1 2 10","* 7 3 0 2 10","* 1 4 2 2 10","* 13 8 1 2 10","* 10 13 0 2 10","0 9 6 +++ !","1 5 2 ++ !","2 1 2 !"])

   Por fim, utilizamos este teste de dimensão 15, com uma complexidade muito superior relativamente aos outros testes previamente utilizados, sendo maioritariamente usado para testar a compressão/descompressão na tarefa3 (Devido à sua elevada complexidade, permite-nos testar muitas mais possibilidades numa só aplicação da função, o que nos permite encontrar possíveis erros nas Tarefas mais facilmente.

4 Conclusões

Com a realização deste trabalho e após a execução de todas as Tarefas propostas no enunciado do projeto com vista à reprodução do jogo Bomberman, consideramos que apesar das adversidades pelas quais tivemos que percorrer, conseguimos melhorar as nossas capacidades de programador e que este trabalho foi uma mais valia para o nosso futuro profissional.