Projeto de Laboratórios de Informática 1
Grupo 102

Luís Gonçalo Epifânio Pereira - A74413

26 de Abril de 2016

Resumo

Este relatório visa a documentar quais os métodos utilizados, assim como as estratégias que foram seguidas na realização da 2ª fase do projecto, no âmbito da disciplina de Laboratórios de Informática 1 *(LI1)*, do Mestrado Integrado em Engenharia Informática *(MIEI)*.

No seguimento de três outras tarefas anteriores, aqui será feita uma breve análise acerca de três tarefas relativas ao jogo de tabuleiro *Sokoban*, no qual teríamos de, através de um *input* via terminal, desenhar e testar várias condições que verificassem a validade do programa em questão.

Como tal, abordar-se-ão as questões postas nas três fases propostas para esta segunda fase, assim como o método de resolução escolhido para concretização da realização de cada uma das tarefas.

Conteúdo

Capítulo 1

Introdução

A estrutura deste relatório está subdividida em três tópicos, cada um respectivo à sua tarefa (D, E, F). Em cada uma destas, o método utilizado para resolução de tais problemas baseou-se, maioritariamente, na construção de funções com listas (recorrendo, também, por vezes, a funções de ordem superior).

Na secção Desenvolvimento, será apresentado o problema em questão, assim como a resolução que está por detrás do código apresentado em cada Tarefa.

Na secção Conclusão, será feita uma breve análise crítica de todos os resultados obtidos, não esquecendo também dificuldades passadas na resolução de cada uma das tarefas. Esta secção também remeterá para obstáculos os quais não foram possíveis de ultrapassar.

Capítulo 2

Desenvolvimento

2.1 Tarefa D

2.1.1 Problema

Esta tarefa pretendia um seguimento da tarefa C, na qual calculava a próxima posição através de um comando. Nesta última tarefa, foi-nos pedido que implementássemos um programa que, através de um tabuleiro, posições do boneco (jogador) e posições de caixas, calculássemos não somente a próxima posição, mas sim as próximas posições.

Como tal, o programa devolveria como output uma das duas *strings* seguintes:

1. INCOMPLETO x
2. FIM x

em que x é o número de comandos efectivamente executados (isto é, comandos que alterassem a posição do jogador).

2.1.2 Resolução

De modo a solucionar esta tarefa, foi seguida uma abordagem que recorreria a funções recursivas e a listas temporárias que guardariam as coordenadas das caixas e dos pontos, tal como na tarefa C.

De seguida, atribuí, a cada uma das caixas, um "identificador", de modo a facilitar a modificação de coordenadas após a execução de cada comando (no caso em que houvesse alteração da posição de uma caixa).

```
boxIdentify :: [(Int, Int)] -> Int -> [((Int, Int),Int)]
boxIdentify [] _ = []
boxIdentify (caixa:caixas) identify = (caixa,identify) : boxIdentify caixas (identify+1)
```

Assim, e recorrendo a uma função que processava os comandos, processar-se-iam os movimentos, de acordo com os comandos fornecidos.

Porém, e cada vez que um comando fosse executado, uma outra função auxiliar, de nome *checkEnding*, verificaria se a condição "FIM"teria sido atingida, de modo a forçar um *break* na função de processamento.

```
checkEnding :: [((Int, Int),Int)] -> [(Int, Int)] -> Bool
checkEnding [] _ = True
checkEnding (((x,y),_):resto) a | (elem (x,y) a) = checkEnding resto a
                                | otherwise      = False
```

Deste modo, seria possível um controlo do fluxo de execução de comandos, de modo a limitar os comandos desnecessários na eventualidade da condição de "ganhar o jogo"ter sido atingida.

2.2 Tarefa E

2.2.1 Problema

Esta tarefa visava uma breve introdução à biblioteca *Gloss*, na qual teríamos de, através de tipos fornecidos pela mesma, teríamos de calcular as dimensões do tamanho mínimo de uma figura esboçada via Gloss.

Como tal, recorreríamos a vários tipos de *Pictures*, tais como *Circle*, *Line* e até *Polygon*.

2.2.2 Resolução

A resolução deste problema terá sido feita analisando cada uma das figuras existentes e fornecidas pela Biblioteca, nas quais foram feitas as condições individuais que referiam a cada caso.

```
envolvingFigure :: Picture -> (Float, Float, Float, Float)
envolvingFigure (Pictures x)= finalEnvolvingPicture (map envolvingFigure x) (0.0, 0.0, 0.0, 0.0)
envolvingFigure (Blank) = (a, a, a, a)
                          where a = 0 :: Float
envolvingFigure (Polygon x) = (a, b, c, d) --cima direita baixo esquerda
                          where a = maximum (map (\(px,_) -> px) x)
                                b = maximum (map (\(_,pw) -> pw) x)
                                c = minimum (map (\(px,_) -> px) x)
                                d = minimum (map (\(_,pw) -> pw) x)
envolvingFigure (Line x) = (a, b, c, d) --cima direita baixo esquerda
                          where a = maximum (map (\(px,_) -> px) x)
                                b = maximum (map (\(_,pw) -> pw) x)
                                c = minimum (map (\(px,_) -> px) x)
                                d = minimum (map (\(_,pw) -> pw) x)
envolvingFigure (Circle x)  = (x,x,(-x),(-x))
envolvingFigure (Bitmap x y _ _) = ((b/2), (a/2), -(b/2), -(a/2))
                            where a = fromIntegral x
                                  b = fromIntegral y
envolvingFigure (Color _ _) = (a, a, a, a)
                    where a = 0 :: Float
envolvingFigure (Translate x y pic) = translatePicture (envolvingFigure pic) x y
envolvingFigure Rotate x pic = rotatePicture (envolvingPicture pic) x
envolvingFigure (Scale x y pic) = scalePicture (envolvingFigure pic) x y
```

Desta forma, foi feita uma análise consoante o *input* recebido. Porém, e devido à impossibilidade de descoberta do erro até à data de escrita deste relatório, não me foi possível esboçar *Pictures [Pictures]*, retornando um erro de indentação. Assim, e como este erro é crucial para a receção do *input*, não me foi possível a atribuição de qualquer pontuação na plataforma de submissões *Mooshak*.

2.3 Tarefa F

2.3.1 Problema

Esta tarefa remetia para a realização de uma versão animada da tarefa D através da Biblioteca Gloss. Como tal, esta tarefa não seria avaliada via Mooshak, visto que é algo meramente gráfico e, como tal, não é possível ser testada por qualquer tipo de *input*.

2.3.2 Resolução

Este problema terá sido abrangido de uma forma bastante vaga, sendo que apenas terá sido criada a animação para "empurrar"caixas. Como tal, não terão sido efectuadas as condições de verificação de fim de jogo, resultando num programa interminável.

Capítulo 3

Conclusão

Findando, e na minha opinião, foi-me possível atingir as metas estabelecidas para as tarefas D e E. Porém, na tarefa E, devido a um percalço, não me foi possível concluir com sucesso esta última tarefa. Relativo à tarefa F, poderia ter sido feito algo mais do que o que foi apresentado. Foi feita uma má gestão de tempo ao tentar descobrir qual o erro na tarefa E e, como tal, daí resultou uma tarefa F abaixo das expectativas.

No geral, opino que a resolução das tarefas tenha sido feita de forma satisfatória e, deste modo, concluo também que o tema abordado terá sido interessante, na medida em que me incitou a procurar e investigar mais acerca da animação através de comandos responsivos ao teclado.