Relatório Trabalho Prático LI1

Grupo 125

José Nuno Martins da Costa e Gonçalo Rodrigues Pinto

31 de Dezembro de 2017

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Contextualização

No 1º ano do Curso de Engenharia Informática da Universidade do Minho, existe uma Unidade Curricular denominada por "Laboratórios de Informática I", que tem como objetivo ensinar aos estudantes a desenvolver programas na linguagem de programação Haskell, documentar código que pode ser muito útil para facilitar a compreensão do programa por parte de quem lê o código fonte, trabalhar em equipa e assim promover o uso de sistemas de controlo de versões e utilizar uma biblioteca Haskell minimalista denominada de Gloss para a criação de gráficos e animações 2D, como também, promover o contacto com uma plataforma - Latex, para a produção de documentos de elevada qualidade cuja ideia central é possibilitar a geração de documentos estruturados de uma forma simples e consistente.

1.2 Motivação

A principal motivação deste trabalho é construir o clássico jogo de corridas "Micro Machines"em linguagem Haskell, proporcionando a aprendizagem desta linguagem de programação funcional, permitir soluções para problemas matemáticos, o facto desta linguagem ser bastante poderosa em termos de expressividade e também o facto de as funções poderem expressar coisas de forma sucinta mas legível, tornando mais fácil a representação de um problema.

1.3 Objectivos

O objetivo do projeto de Laboratórios de Informática I 2017/2018 é fazer o clássico jogo de corridas "Micro Machines". O jogo irá consistir de vários programas em Haskell divido em 2 fases avaliadas separadamente, em que cada fase é composta por 3 tarefas.

Capítulo 2

Análise de Requisitos

2.1 Fase 1

A 1ª Fase do projeto é dividida em três tarefas:
- Na 1ª Tarefa o objetivo é construir mapas, ou seja, implementar funções para que qualquer que seja o caminho que damos à função como input recebemos um mapa como output;
- Na 2ª Tarefa o objetivo é validar mapas, de modo a que as regras do enunciado sejam verificadas;
- Na 3ª Tarefa o objetivo é implementar um parte das mecânicas do jogo, como as movimentações do carro no mapa e as colisões com obstáculos.

2.2 Fase 2

A 2ª Fase do projeto também é dividida em três tarefas:
- Na 4ª Tarefa o objetivo é atualizar o estado do jogo consoante as ações efetuadas por um jogador num determinado período de tempo;
- Na 5ª Tarefa o objetivo é implementar todas as tarefas anteriores, ou seja, o jogo, na biblioteca Gloss;
- Na 6ª Tarefa o objetivo é implementar um bot que jogue o jogo automaticamente.

Capítulo 3

A Nossa Solução

3.1 1ª Fase

3.1.1 Tarefa 1

```
{-
Como o ficheiro não compilava decidimos encurtar a função
ao substituir com o "atc" a função
"atualizaCaminhoPosicaoemTabuleiro" e por "ppp" a
função "passosemPecaPosicao". O programa não compila com estas
denominações, os nomes foram mudados puramente para o relatório.
-}
constroi :: Caminho -> Mapa
constroi c = Mapa (partida c,dirInit) t
        where t = atc (ppp c) (tudoLava (novaDimensao c))
```

Para a resolução desta tarefa usamos as seguintes funções:
- A função "partida"que dado um caminho qualquer calcula a posição inicial no tabuleiro;
- A função "passosemPecaPosicao"que transforma um caminho numa lista formada por pares, compostos por uma peça e a posição no tabuleiro;
- A função "tudoLava"que cria um tabuleiro com a dimensões fornecidas, apenas com peças do tipo Lava 0;
- A função "novaDimensao"que ajusta a função "dimensao"fornecida pelos docentes;
- A função "atualizaCaminhoPosicaoemTabuleiro"que insere uma lista de peças e posições num determinado tabuleiro.

3.1.2 Tarefa 2

```
{-
Devido às mesma limitações referidas acima, substituímos
o código da tarefa 2, por código equivalente, onde
usamos "where" para escrever as funções
usadas em "valida".
-}
valida :: Mapa -> Bool
valida (Mapa ((x,y),o) t) = a && b && c && d && e && f && g
     where a = pecainicialvalida (Mapa ((x,y),o) t)
           b = validaPosicaoInicial (Mapa ((x,y),o) t)
           c = validaAltura (Mapa ((x,y),o) t)
           d = orientacaoInicialFinal (Mapa ((x,y),o) t)
           e = alturatabuleiroLava t
           f = contapecas (Mapa ((x,y),o) t)
           g = tabuleiroRetangulo t
```

Para a resolução desta tarefa usamos as seguintes funções:
- A função "pecainicialvalida"que testa se a peça inicial é do tipo Lava;
- A função "validaPosicaoInicial"que testa se o mapa corresponde a uma trajectória, tal que começa na peça de partida com a orientação inicial e volta a chegar à peça de partida com a orientação inicial;
- A função "validaAltura"que testa se as peças do percurso estão ligadas a peças do percurso com alturas compatíveis;
- A função "orientacaoInicialFinal"que testa se a orientação inicial é compatível com a peça de partida;
- A função "alturatabuleiroLava"que testa se as peças do tipo Lava de um tabuleiro têm altura 0;
- A função "contapecas"que testa se o número total de peças no tabuleiro é igual à soma de peças do tipo Lava e sem ser do tipo Lava, recurso à função que define um percurso;
- A função "tabuleiroRetangulo"que testa se o tabuleiro é rectângulo.

3.1.3 Tarefa 3

```
movimenta :: Tabuleiro -> Tempo -> Carro -> Maybe Carro
movimenta m t (Carro p d v) = maybeCarro (dPLava m p v t)
```

Para a resolução desta tarefa usamos as seguintes funções:
- A função "dPLava"que dado um Tabuleiro, um Ponto, o vetor velocidade e tempo irá devolver Nothing caso a peça resultante da translação seja do tipo Lava e Just 'peca ...' caso contrário;
- A função "maybeCarro"que 'converte' um Nothing de uma Maybe Peca num Nothing de um Maybe Carro.

3.2 2ª Fase

3.2.1 Tarefa 4

```
atualiza :: Tempo
         -> Jogo
         -> Int
         -> Acao
         -> Jogo
atualiza t e j a = Jogo a1 a2 a3 a4 a5
        where a1 = mapa e
              a2 = pista e
              a3 = atualizaCarros e t j a
              a4 = atualizaNitro a j t (nitros e)
              a5 = atualizaHistorico e j a
```

Para a resolução desta tarefa usamos as seguintes funções:
- A função "atualizaCarros"que atualiza a direção e velocidade de cada carro;
- A função "atualizaNitro"que atualiza a quantidade de nitro conforme um determinado período de tempo;
- A função "atualizaHistorico"que que atualiza o histórico de posições dos carros;
- Também foi usado "mapa"e "pista", que não são funções mas sim uma forma de manter o mapa e as propriedades do input, no output.

3.2.2 Tarefa 5

```
main :: IO ()
main = do
      inicio <- imgCarro
      joga inicio
joga :: Estado -> IO ()
joga inicio = play
    (InWindow "Micro Machines" (1366, 768) (0,0))
    (greyN 0.5)
    60
    inicio
    desenhaEstado
    reageEvento
    reageTempo
```

Para a resolução desta tarefa usamos as seguintes funções:
- A função "imgCarro"que carrega a imagem do carro e cria o estado inicial, que vai ser usado na função principal;
- A função "desenhaEstado"que desenha o sítio onde o carro vai andar;
- A função "reageEvento"que faz o carro mover conforme uma determinada ação, neste caso, a ação é carregar num determinado botão;
- A função "reageTempo"que altera o estado do jogo consoante um determinado período de tempo;
- A função "joga"que essencialmente vai usar todas as funções definidas e criar um jogo, dendendo de um estado, que será dado na função "main", pela função "imgCarro".

3.2.3 Tarefa 6

Não conseguimos implementar esta tarefa uma vez que esta se encontra condicionada por etapas anteriores para as quais não conseguimos identificar o problema.

Capítulo 4

Validação da Solução

4.1 1ª Fase

4.1.1 Tarefa 1

A nossa estratégia para atingir o primeiro objetivo na fase 1 foi transformar um caminho, uma lista de passos, numa lista com pares sendo o primeiro elemento peças e o segundo a posição dessa peça, de seguida criamos um tabuleiro com lava com uma dimensão e por fim inserimos uma lista de peças com posições associadas no tabuleiro com lava.
Para validar a 1ª tarefa do projeto usamos 6 testes variando de caminhos complexos para caminhos simples

```
*Tarefa1_2017li1g125> constroi [Sobe,Avanca,Sobe,CurvaDir,CurvaDir,Desce,Avanca,De
sce,CurvaDir,CurvaDir]
Mapa ((2,1),Este) [[Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0,Pe
ca Lava 0,Peca Lava 0],[Peca Lava 0,Peca (Curva Norte) 0,Peca (Rampa Este) 0,Peca
Recta 1,Peca (Rampa Este) 1,Peca (Curva Este) 2,Peca Lava 0],[Peca Lava 0,Peca (Cu
rva Oeste) 0,Peca (Rampa Este) 0,Peca Recta 1,Peca (Rampa Este) 1,Peca (Curva Sul)
 2,Peca Lava 0],[Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca
Lava 0,Peca Lava 0]]
```

Figura 4.1: Output da função constroi de um caminho dos testes, relativamente simples

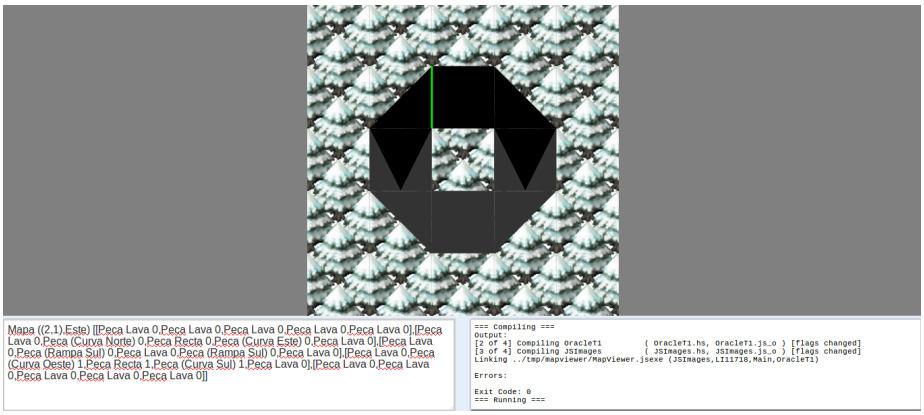

Figura 4.2: Mapa no visualizador

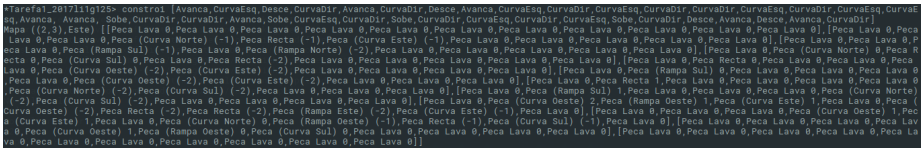

Figura 4.3: Output da função constroi de um caminho dos testes mais complexo

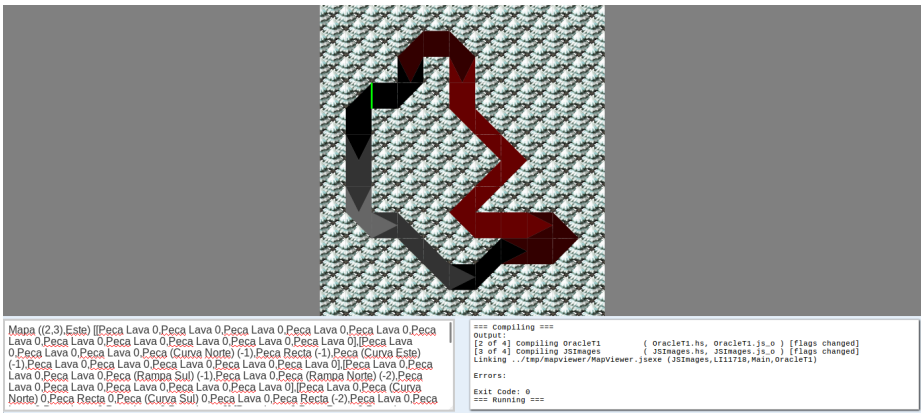

Figura 4.4: Mapa no visualizador

4.1.2 Tarefa 2

A nossa solução para testar se um mapa é válido ou não criámos funções para cada regra definida, como tal, definiu-se uma função que testa se o mapa corresponde a uma trajectória, tal que começa na peça de partida com a orientação inicial e volta a chegar à peça de partida com a orientação inicial; outra para testar se a peça inicial é lava; uma função que testa se as peças do percurso estão ligadas a peças do percurso com alturas compatíveis; de seguida, uma outra que valida se a orientação inicial é compatível com a peça de partida; função para testar se as peças do tipo lava de um tabuleiro têm altura 0; função que testa se o número total de peças no tabuleiro é igual à soma de peças do tipo lava, e sem, ser do tipo lava e finalmente uma função para testar se o tabuleiro é rectângulo e se tem a primeira e a última linha assim como a primeira e a última coluna como uma peça do tipo Lava e altura 0.

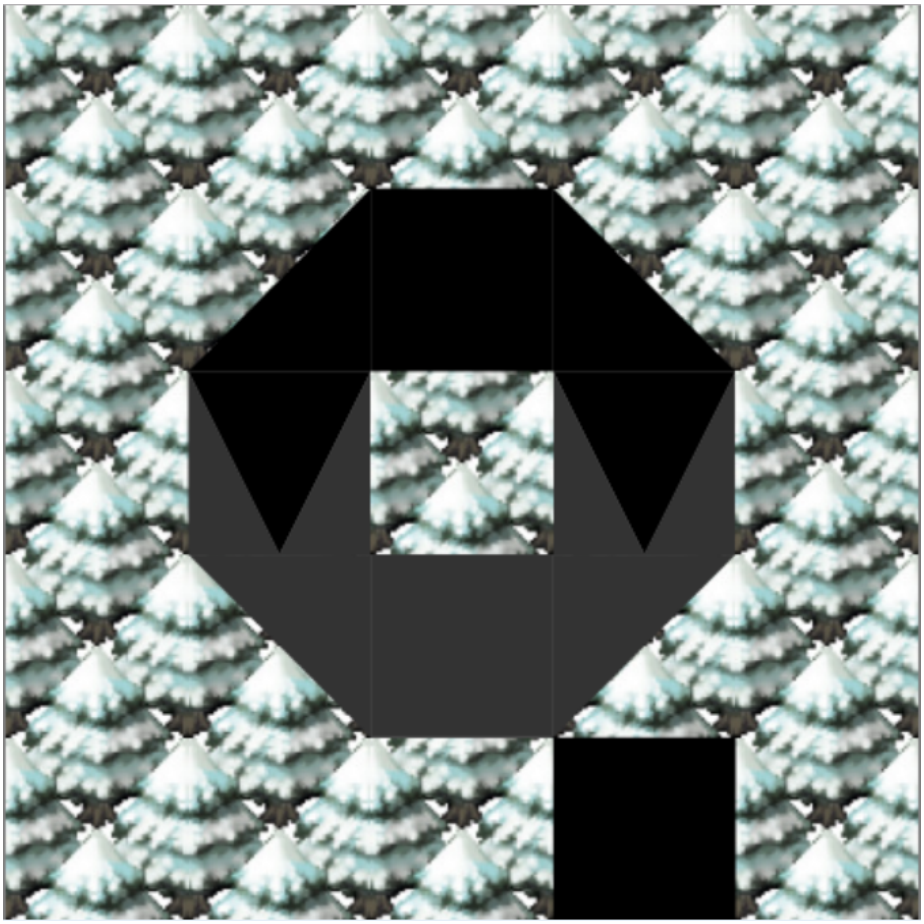

Figura 4.5: Exemplo 1: Teste inválido da tarefa 2

```
?Tarefa2_2017li1g125> valida (Mapa ((1,0),Este) [[Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0],[Peca Lava 0,Peca (Curva Norte) 0,Peca Recta 0,Peca (Curva Es
te) 0,Peca Lava 0],[Peca Lava 0,Peca (Rampa Sul) 0,Peca Lava 0,Peca (Rampa Sul) 0,Peca Lava 0],[Peca Lava 0,Peca (Curva Oeste) 1,Peca Recta 1,Peca (Curva Sul) 1,Peca Lava 0
],[Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Recta 0,Peca Lava 0]])
False
```

Figura 4.6: Output da função valida ao Exemplo 1

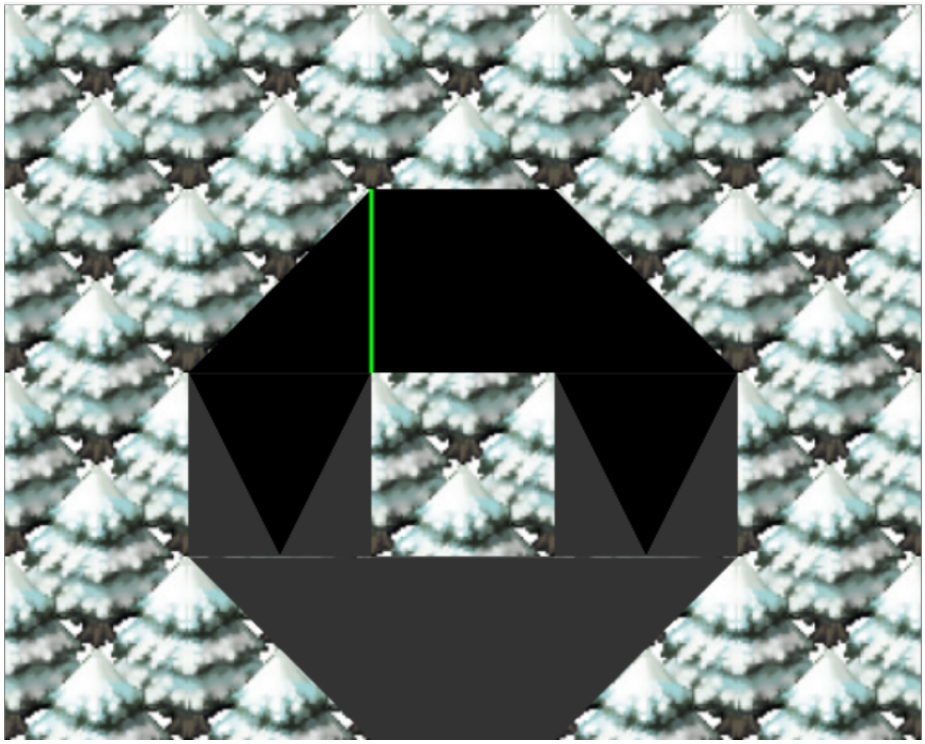

Figura 4.7: Exemplo 2: Teste inválido da tarefa 2

```
?Tarefa2_2017li1g125> valida (Mapa ((2,1), Este) [[Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0],[Peca Lava 0,Peca (Curva Norte) 0,Peca Recta 0,Peca (Curva E
ste) 0,Peca Lava 0],[Peca Lava 0,Peca (Rampa Sul) 0,Peca Lava 0,Peca (Rampa Sul) 0,Peca Lava 0],[Peca Lava 0,Peca (Curva Oeste) 1,Peca Recta 1,Peca (Curva Sul) 1,Peca Lava
0]])
False
```

Figura 4.8: Output da função valida ao Exemplo 2

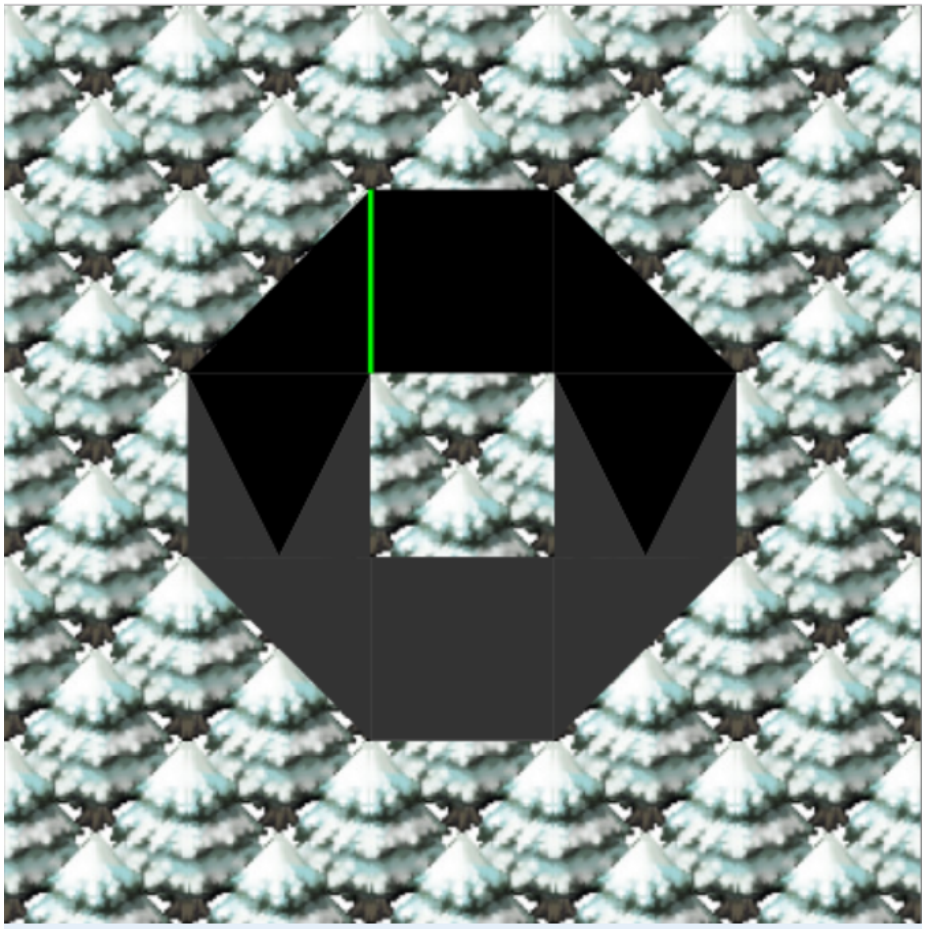

Figura 4.9: Exemplo 3: Teste válido da tarefa 2

```
*Tarefa2_20171i1g125> valida (Mapa ((2,1), Este) [[Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0],[Peca Lava 0,Peca (Curva Norte) 0,Peca Recta 0,Peca (Curva E
ste) 0,Peca Lava 0],[Peca Lava 0,Peca (Rampa Sul) 0,Peca Lava 0,Peca (Rampa Sul) 0,Peca Lava 0],[Peca Lava 0,Peca (Curva Oeste) 1,Peca Recta 1,Peca (Curva Sul) 1,Peca Lava
0],[Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0,Peca Lava 0]])
True
```

Figura 4.10: Output da função valida ao Exemplo 3

4.1.3 Tarefa 3

Para concluir a última tarefa da primeira fase definiu-se uma função que dado um Tabuleiro, um Ponto, um vetor velocidade e um tempo devolve "Nothing"caso a peça resultante da translação seja do tipo lava, e caso contrário devolve o tipo da peça, assim se a peça for do tipo lava o carro é destruído.

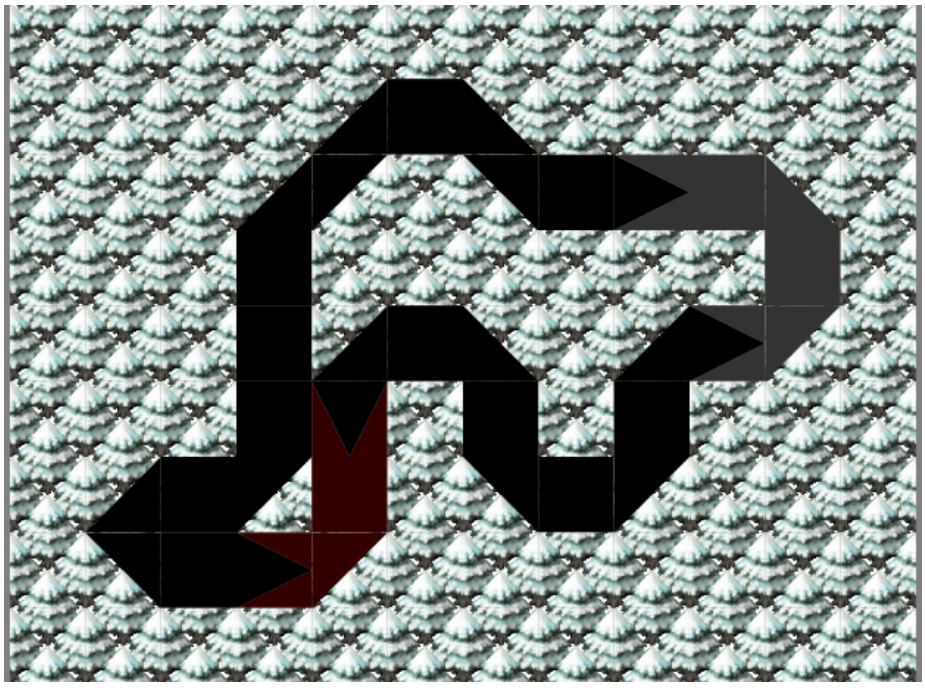

Figura 4.11: Tabuleiro usado para os testes da tarefa 3, denominado por "tab1"

```
*Tarefa3_2017li1g125> movimenta tab1 2 (Carro (2.5,6.5) 45 (0,-1))
Nothing
```

Figura 4.12: Output da função movimenta o 1º teste da tarefa 3

```
*Tarefa3_2017li1g125> movimenta tab1 1 (Carro (6.1,4.8) 45 (-1,-1))
Nothing
```

Figura 4.13: Output da função movimenta o 2º teste da tarefa 3

4.2 2ª Fase

4.2.1 Tarefa 4

A nossa estratégia para cumprir o primeiro objetivo na primeira tarefa da fase 2 foi atualizar a velocidade criando funções individuais para calcular os diferentes vetores e posteriormente perante ações do jogador somar os vetores; para atualizar a direção simplesmente verificámos se o carro se encontrava a rodar para a esquerda ou para a direita; para atualizar a quantidade do nitro, criámos uma função que diminui a quantidade de tempo percorrida; para atualizar o histórico verificámos se mudou de posição desde a última atualização. Criou-se ainda uma função que atualiza a direção e velocidade do carro correspondente a cada jogador.

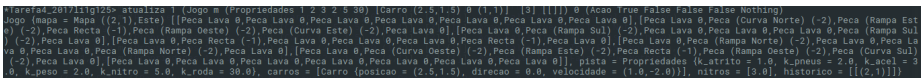

Figura 4.14: Output da função atualiza do teste 1

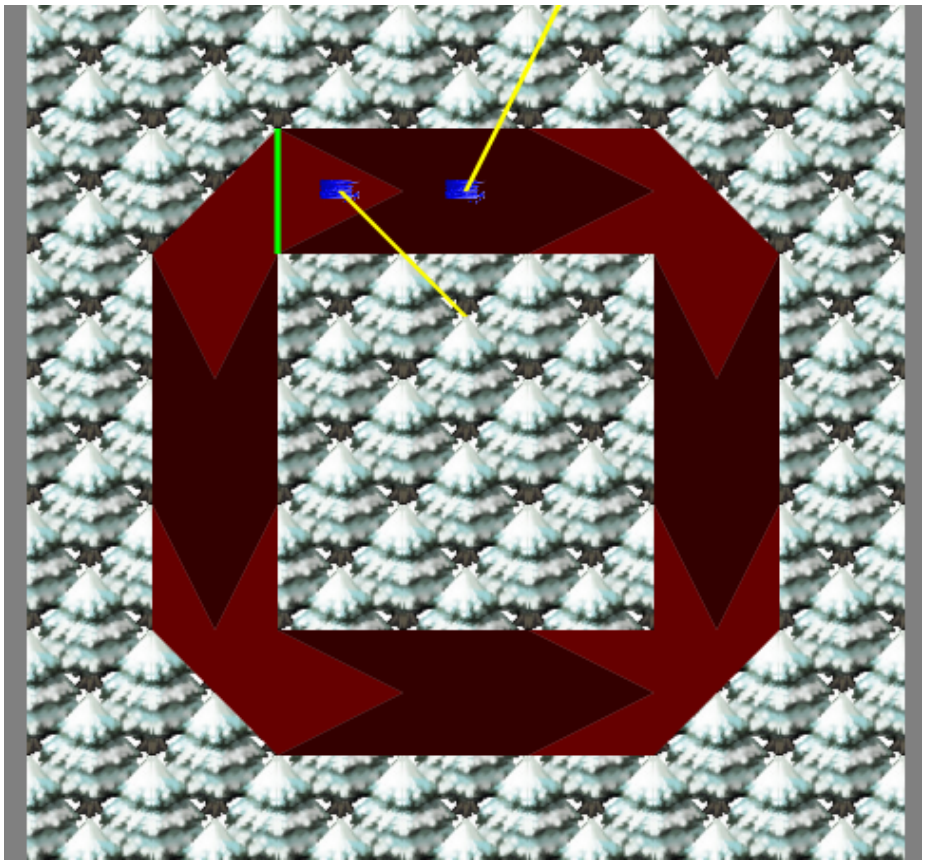

Figura 4.15: Visualização do movimento do carro, do teste 1

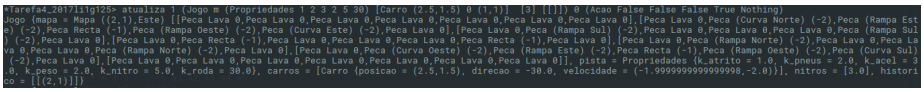

Figura 4.16: Output da função atualiza do teste 6

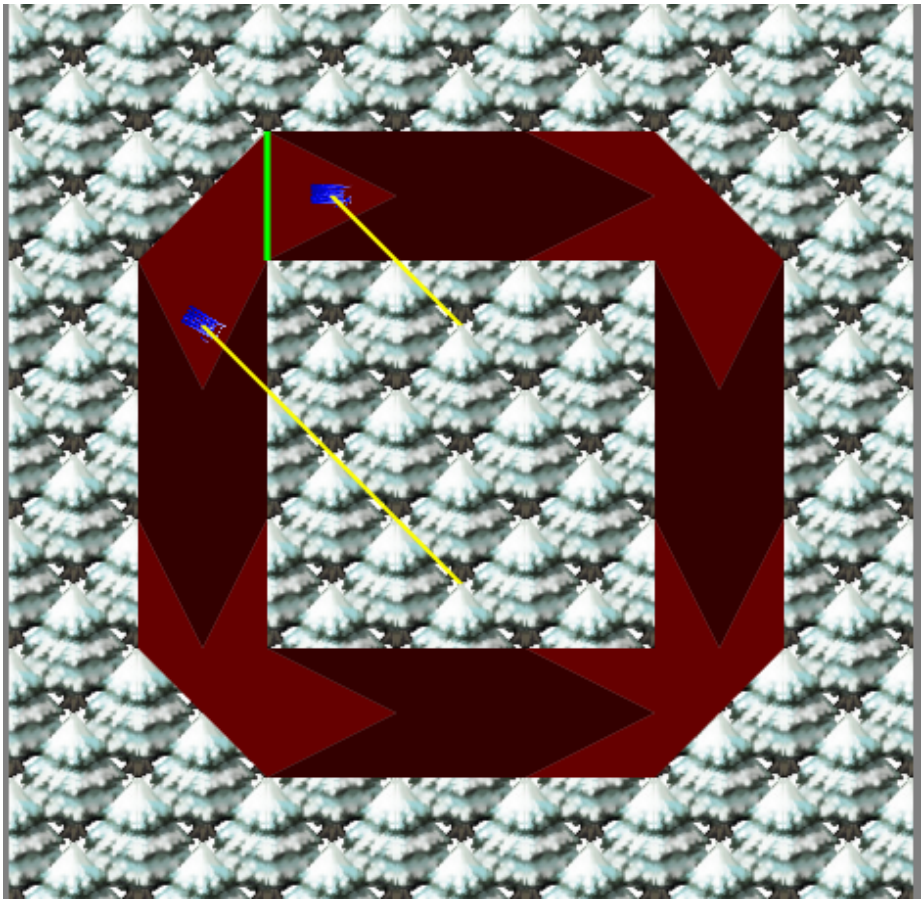

Figura 4.17: Visualização do movimento do carro, do teste 6

4.2.2 Tarefa 5

De modo a completar a 5ª tarefa do projeto decidimos começar por colocar o carro a movimentar-se numa determinada direção, conforme um clique num botão, com isso completo fizemos funções para que o carro andasse sempre numa determinada direção, até que o utilizador largasse o botão Os exemplos seguintes mostram um movimento do carro na diagonal, quando se carrega na teclas seta para baixo e para a direita. Contudo esta tarefa não se encontra totalmente realizada devido às muitas dificuldades sentidas nas funções a utilizar.

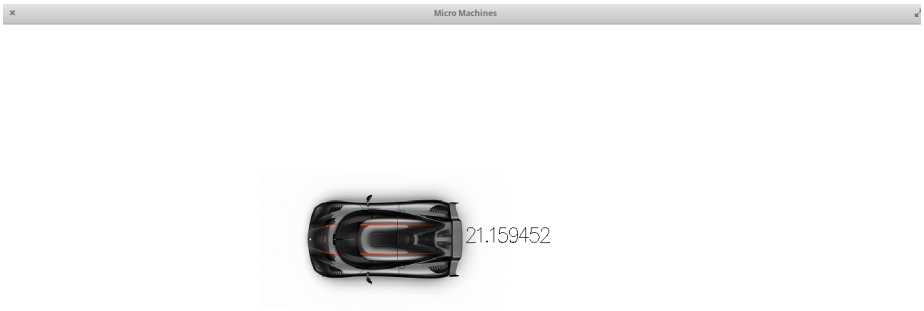

Figura 4.18: Exemplo 1 do carro no jogo

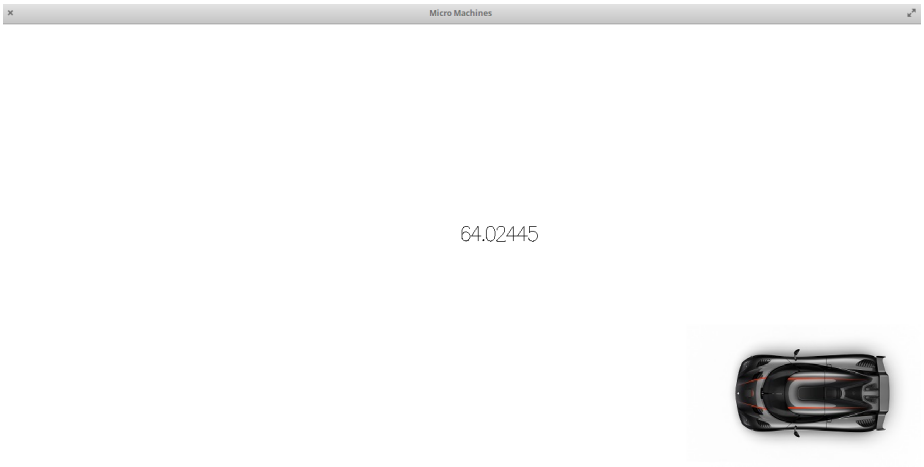

Figura 4.19: Exemplo 2 do carro no jogo

4.2.3 Tarefa 6

Não conseguimos implementar esta tarefa por não termos conseguido identificar as funções a implementar para o funcionamento da mesma.

Capítulo 5

Conclusão

"Strength does not come from winning. Your struggles develop your strengths. When you go through hardships and decide not to surrender. That is strength"

- Mahatma Gandhi

O projeto que nos foi proposto não foi fácil mas nunca desistimos, demos o nosso melhor, no entanto não conseguimos cumprir com sucesso todos os objectivos propostos, deixámos a última tarefa por realizar e além disso algumas ficaram incompletas devido a: falta de tempo, de conhecimentos e de experiência na área de programação. Neste trabalho abordámos as etapas que nos foram propostas apresentando soluções para os problemas e procedendo à validação destes. Apesar das dificuldades sentidas achamos que este projeto contribuiu de forma positiva para a nossa formação pois permitiu-nos: trabalhar em equipa; a desenvolver as nossas competências de investigação; de organização; a desenvolver os nossos conhecimentos e as nossas capacidades de programação ao nível desta linguagem.