Projeto Race For Fun - Relatório

Laboratórios de Informática I

2019/2020

Joana Freitas, n.º 55985
Mariana Rodrigues, n.º 93229

Índice

Introdução

No âmbito da unidade curricular Laboratórios de informática I, foi-nos proposto o desenvolvimento de um jogo 2D, de corridas entre motas. Sendo que, os objetivos gerais do jogo são de percorrer a pista, o mais rápido possível; realizar saltos e evitar obstáculos.

O projecto encontra-se dividido em duas fases, tendo em conta que, cada uma é composta por três tarefas distintas; e visa a utilização da linguagem de programação *Haskell*, para a construção de cada tarefa. Onde contempla a aplicação dos conhecimentos adquiridos, tanto nas aulas práticas e de apoio, como também da unidade curricular, Programação Funcional. Além desta linguagem, recorreu-se ao Sistema de Controle de Versões (*SVN*), como também o *Gloss,* para a construção da animação do jogo. Também foi disponibilizada uma plataforma que nos permite a avaliação quantitativa e avaliativa de cada tarefa e do jogo, em comparação ao *oráculo*, equipa do corpo docente.

Assim, presente relatório está composto por cada etapa proposta, onde iniciamos por apresentar os objetivos adjacentes ao projeto e, por fim, como resolvemos os desafios colocados.

PROJECTO

As tarefas solicitadas para o trabalho, apesar de terem objetivos distintos entre si, são dependentes umas das outras, para atingir o objetivo primordial, a execução do jogo, cumprindo as metas gerais, já descritas no início do documento.

Fase I

Tarefa 1

Objetivo:

Construção de um gerador de mapas, sabendo que os inputs são o número de pistas, o comprimento de cada pista e a semente, que é sempre um número inteiro positivo. Salienta-se que, a função geraAleatorios, que recebe um número de elementos a gerar, uma semente e produz uma lista de números pseudo aleatórios, com valores entre 0 e 9, foi fornecida pelos instrutores.

Desenvolvimento da tarefa:

Um mapa corresponderá a uma matriz com um número independente de cada linha (pista), sendo que as colunas têm um número idêntico a cada linha. A matriz será composta exclusivamente por listas de pares de números, em que o primeiro número corresponde ao piso da peça, por exemplo se é Relva ou Terra, enquanto que, o segundo determina o tipo da peça, ou seja, se é Recta ou Rampa. Sendo que, cada célula da matriz é uma peça da pista.

Iniciamos por implementar uma função, em que o mapa tem comprimento de uma unidade, utilizando a informação que todas as pistas iniciam com uma peça do tipo Recta, com piso Terra e com uma altura de 0. Seguidamente, foi necessária a função que gera um mapa com um número aleatório de pistas, comprimento e a semente. Neste caso, retiramos o primeiro elemento de cada lista/pista, tendo em conta que que já se encontravam definidos anteriormente. No entanto, tivemos que adicionar uma função auxiliar que transforma uma lista de pares, numa lista de peças, com o fim de obter uma série de pistas distintas, variando

nos pisos, tipos e alturas. Ou seja, implementamos uma função secundária que verificava se: 1) o primeiro elemento da lista é do piso Terra [0,1], Relva [2,3], Lama [4], Boost [5] ou do mesmo piso da peça anterior [6..9] e 2) o segundo elemento da lista é Rampa a subir [0,1], Rampa a descer [2 a 5] ou Recta, com altura da peça anterior [6...9]. Porém, deparamo-nos com um problema relativo ao cálculo na altura das peças, sendo que a resolução encontrada foi diferenciar as alturas necessárias. Isto é, a primeira função, consoante o segundo valor da lista, devolve-nos a altura final da peça (Rampa ou Recta); enquanto que, a segunda é utilizada para inserir o valor da altura, dada da primeira função, na função auxiliar, que transforma a lista de pares, numa lista de peças.

Resultados:

1) Testes

Foram obtidos resultados positivos, relativamente aos testes definidos no projecto, comparado a solução contra o oráculo ideal, definido pelos instrutores.

2) Cobertura dos Testes (HPC)

module	Top Level Definitions		Alternatives		Expressions	
	%	covered / total	%	covered / total	%	covered / total
module LI11920	3%	2/62	0%	0/12	0%	0/31
module Main	100%	1/1	60%	3/5	73%	19/26
module Tarefa1_2019li1g088	100%	7/7	100%	24/24	100%	190/190
Program Coverage Total	14%	10/70	65%	27/41	84%	209/247

3) Testes (Avaliação)

Argumentos	Resultado
(0, 1, 15)	OK
(1, 1, 20)	OK
(2, 2, 0)	OK
(20, 10, 9223372036854775807)	OK
(10, 20, -9223372036854775808)	OK
(30, 5, 548212)	OK
(15, 15, 8843563)	OK
(8, 5, 645)	OK

Tarefa 2

Objetivo:

Construção das possíveis jogadas a serem efetuadas no jogo, ou seja, dada uma descrição do estado do jogo e uma jogada dos jogadores, conseguir determinar o efeito dessa jogada, no estado do jogo.

Desenvolvimento da tarefa:

Começamos por incrementar uma função principal, que procura na matriz o jogador que pretendemos alterar, dando as suas coordenadas no mapa; como também o estado atualizado do jogo, caso seja efectuada uma jogada, recorrendo a uma função secundária. Salienta-se que, esta função secundária é a que determinará a direção da jogada [C,D,B,E], a sua velocidade (acelera ou desacelera) e o disparo da cola. Tendo em conta estes três parâmetros, não excluindo as regras impostas para cada jogada, dividimos em partes distintas as jogadas possíveis.

Primeiramente, desenvolvemos funções que envolvessem o disparar da Cola, sendo necessário verificar se o jogador contém Cola para ser usada. Caso seja verdadeiro e, se o jogador disparar: 1) o mesmo ficará com menos uma unidade; 2) há uma alteração da matriz, isto quer dizer, que dada a posição do jogador e do mapa, existirá uma alteração do piso anterior, onde se encontrava o Jogador, para Cola; e 3) mudança no piso da peça em questão, para cola, variando se a mesma é uma Rampa ou Recta.

Relativamente à velocidade da jogada, implementamos uma função que verifica se o estado do Jogador lhe permite efetuar a jogada Acelera ou Desacelera, sabendo que o estado do jogador só se altera, quando este se encontra no Chão.

Por fim, construímos funções diferentes para a direções Cima ou Baixo e Esquerda ou Direita. Assim, implementamos uma condição que verifica se o jogador reúne as condições necessárias para efectuar a jogada de mudança de pista. Caso este cenário seja verdadeiro, a função adjacente faz com que o jogador se mova no mapa. Esta função secundária auxilia a localizar a posição do jogador na matriz; a Peça para onde o jogador pretende efectuar a jogada; verifica se é Rampa ou Recta; e averigua a diferença absoluta de alturas entre a peça, onde o Jogador se encontra, e a peça, para onde o jogador pretende deslocar-se, é

>0.2 ou =<0.2. Enquanto que, para as direções Esquerda ou Direita utilizamos uma função que analisa se o Jogador se encontra no Ar, e isto acontecendo, vai alterar a sua rotação. Para que esta alteração aconteça, recorremos duas funções, uma que altera a inclinação mediante se o movimento é D ou E, e outra que corrige a rotação, ou seja, caso o Jogador tenha rodado em demasia, esta função tem como objetivo normalizar o ângulo de rotação.

Resultados:

1) Testes

Foram obtidos resultados positivos, relativamente aos testes definidos no projecto, comparado a solução contra o oráculo ideal, definido pelos instrutores.

2) Cobertura dos Testes (HPC)

module	Top Level Definitions		Alternatives		Expressions	
	%	covered / total	%	covered / total	%	covered / total
module LI11920	29%	18/62	0%	0/12	0%	0/31
module Main	100%	1/1	60%	3/5	70%	19/27
module Tarefa0_2019li1g088	14%	9/64	6%	8/124	71%	2235/3115
module Tarefa1_2019li1g088	85%	6/7	87%	21/24	84%	161/190
module Tarefa2_2019li1g088	100%	3/3	93%	56/60	97%	492/505
Program Coverage Total	27%	37/137	39%	88/225	75%	2907/3868

3) Testes (Avaliação)

Os resultados obtidos foram positivos, exceptuando nesta situação:

```
( 0
, Movimenta B
, Estado {mapaEstado = [[Recta Terra 0,Recta Terra 0,Rampa Lama 0 2,Recta Lama 2,Rampa
Relva 2 0,Recta Relva 0,Rampa Boost 0 1,Rampa Terra 1 0,Recta Terra 0,Recta Boost
0],[Recta Terra 0,Rampa Terra 0 1,Recta Boost 1,Recta Terra 1,Recta Terra 1,Recta Relva
1,Recta Terra 1,Rampa Relva 1 0,Recta Boost 0,Recta Terra 0]], jogadoresEstado =
[Jogador {pistaJogador = 0, distanciaJogador = 1.1, velocidadeJogador = 0.0,
colaJogador = 0, estadoJogador = Chao {aceleraJogador = True}},Jogador {pistaJogador =
0, distanciaJogador = 0.0, velocidadeJogador = 0.0, colaJogador = 5, estadoJogador =
Chao {aceleraJogador = False}}]} )
```

```
Equality assertion failed:
expected: Estado {mapaEstado = [[Recta Terra 0,Recta Terra 0,Rampa Lama 0 2,Recta
Lama 2,Rampa Relva 2 0,Recta Relva 0,Rampa Boost 0 1,Rampa Terra 1 0,Recta Terra
0,Recta Boost 0],[Recta Terra 0,Rampa Terra 0 1,Recta Boost 1,Recta Terra 1,Recta
Terra 1,Recta Relva 1,Recta Terra 1,Rampa Relva 1 0,Recta Boost 0,Recta Terra 0]],
jogadoresEstado = [Jogador {pistaJogador = 1, distanciaJogador = 1.1,
velocidadeJogador = 0.0, colaJogador = 0, estadoJogador = Chao {aceleraJogador =
True}},Jogador {pistaJogador = 0, distanciaJogador = 0.0, velocidadeJogador = 0.0,
colaJogador = 5, estadoJogador = Chao {aceleraJogador = False}}]}
  got: Estado {mapaEstado = [[Recta Terra 0,Recta Terra 0,Rampa Lama 0 2,Recta
Lama 2,Rampa Relva 2 0,Recta Relva 0,Rampa Boost 0 1,Rampa Terra 1 0,Recta Terra
0,Recta Boost 0],[Recta Terra 0,Rampa Terra 0 1,Recta Boost 1,Recta Terra 1,Recta
Terra 1,Recta Relva 1,Recta Terra 1,Rampa Relva 1 0,Recta Boost 0,Recta Terra 0]],
jogadoresEstado = [Jogador {pistaJogador = 0, distanciaJogador = 1.1,
velocidadeJogador = 0.0, colaJogador = 0, estadoJogador = Morto {timeoutJogador =
1.0}},Jogador {pistaJogador = 0, distanciaJogador = 0.0, velocidadeJogador = 0.0,
colaJogador = 5, estadoJogador = Chao {aceleraJogador = False}}]}
```

Tarefa 3

Objetivo:

Converter um mapa, cujo comprimento de cada pista é maior que 1^2, numa sequência de instruções, que serão fornecidas a um grupo de *bulldozers,* que avançam da partida para constuir o mapa.

Desenvolvimento da tarefa:

Iniciamos por converter um Mapa numa sequências de Instruções, tendo sido necessário complementar com a desconstrução de uma Peça para Instruções, devido às particularidades desta poder ser Recta ou Rampa.

Sabendo que os *bulldozzers* movimentam-se em três padrões distintos, de forma horizontal, vertical e verticais desfasados, delineamos duas funções principais. A primeira, constrói um novo mapa numa sequência de Instruções Verticais, enquanto que a segunda função, transforma o mapa em sequências de Instruções Horizontais. Contudo, para que os *bulldozzers* conseguissem executar os padrões verticais e horizontais foram necessárias uma série de funções auxiliares, associadas às funções principais, referidas acima.

Com isto, relativamente à função, que converte o mapa numa sequência de Instruções Verticais, foi necessário verificar, num mapa, quais as posições iguais, no eixo do y. Sendo necessário, saber onde se localizam as: 1) peças iguais, numa determinada peça, tendo em atenção o tipo de piso e 2) as posições iguais, numa certa pista. Depois, ordenamos numa lista as posições que serão para eliminar, posteriormente. Por fim, com o intuito do *bulldozer* conseguir repetir instruções idênticas, aplicamos uma função que transforma uma lista de Instruções iguais, retirando os elementos similares, numa instrução, que irá repetir as sequências homólogas. No entanto, para que este processo fosse aplicado na matriz, elaboramos uma função auxiliar, que utiliza a Instrução que se repete nas restantes directrizes, que compõem o mapa.

No que concerne às iIstruções horizontais, criamos uma função secundária que analisa: 1) dando uma Instrução, se a seguinte é igual – sendo que este processo só se executa na mesma pista. Caso se verifique, devolve a sua posição e 2) quais as posições iguais numa lista de Instruções, relativas ao eixo do x. Assim, estes resultados serão

utilizados na função principal, onde as posições iguais serão retiradas, para que a instrução se possa Repetir, construindo um novo mapa.

Salienta-se que devido ao seu grau de dificuldade, inerente ao movimento com padrões desfasados, não nos foi possível concluir essa parte do problema.

Resultados:

1) Teste

Foram obtidos resultados positivos, relativamente aos testes relacionados com a desconstrução, comparado a solução contra o oráculo ideal, definido pelos instrutores. Contudo, na Taxa de Compreensão não foi atingido a percentagem máxima, em nenhum dos cenários.

2) Cobertura dos Testes (HPC)

module	Top Level Definitions		Alternatives		Expressions	
	%	covered / total	%	covered / total	%	covered / total
module LI11920	17%	11/62	16%	2/12	29%	9/31
module Main	100%	2/2	57%	8/14	71%	43/60
module Oracle	6%	6/93	10%	8/76	11%	55/493
module OracleT3	68%	13/19	53%	16/30	62%	177/281
module Tarefa0_2019li1g088	21%	14/64	28%	35/124	4%	137/3115
module Tarefa1_2019li1g088	85%	6/7	87%	21/24	84%	161/190
module Tarefa2_2019li1g088	0%	0/3	0%	0/60	0%	0/505
module Tarefa3_2019li1g088	100%	15/15	95%	44/46	99%	419/423
Program Coverage Total	25%	67/265	34%	134/386	19%	1001/5098

3) Testes (Avaliação)

Foram obtidos resultados positivos, relativamente aos testes relacionados com a desconstrução, comparado a solução contra o oráculo ideal, definido pelos instrutores. Contudo, na Taxa de Compreensão não foi atingido a percentagem máxima, em nenhum dos cenários.

Fase II

Tarefa 4

Objetivo:

Calcular o efeito da passagem de um instante de tempo num estado do jogo. Os

inputs, um número real positivo, que denota o tempo que passou desde a última atualização, o mapa do jogo, e o jogador. Sendo o resultado, o jogador após ter sido atualizado, assumindo que nesse instante momento o mapa não sofre qualquer alteração.

Desenvolvimento da tarefa:

Iniciamos por atualizar a velocidade do Jogador, ou seja, alterar a sua velocidade, durante um período de tempo, consoante onde se encontra localizado. Primeiro, localizamos a posição do Jogador e a Peça, onde o mesmo se encontra. Isto, porque houve a necessidade de perceber que tipo de piso era constituída a Peça, para que fosse atribuído o valor correcto do atrito, por exemplo, no caso da peça conter piso Terra, o seu atrito é de 0.25; porém na Lama é de 1.50.

De seguida, desenvolvemos uma função que atualiza a velocidade do Jogador, enquanto este se encontra no Chão. Obedecendo ao facto que, a velocidade do Jogador, no Chão, não pode ser negativa, devolvendo o valor 0, sempre que a velocidade é calculada, como também a alteração existente, devido ao atrito. Denota-se, que implementamos uma função que corrige a velocidade do Jogador, sempre que o x for >=0.

Em relação à velocidade enquanto o Jogador se encontra no Ar, aplicamos a fórmula matemática para calcular a sua velocidade final, tendo em conta a resistência ao Ar (0.125) e a Gravidade. Sendo que, aceleração oferecida pela gravidade está imposta como 1.

No que toca à movimentação do Jogador, desenvolvemos uma função principal que contempla: 1) a posição e a Peça, onde o Jogador se encontra, tanto para o eixo do x como para o eixo do y; 2) o estado do Jogador, ou seja, se este se encontra Morto, no Chão ou no Ar e 3) o caso em que o estado do jogador é alterado para Chao False, mantendo a velocidade a 0. Quando o estado do Jogador é Morto, aplicamos uma função que quando a diferença entre o timeouJogador e o tempo for > 0, diminuímos pelo tempo em questão. Não se verificando esta condição, o Jogador altera-se para Chao False, mas mantém a velocidade a zero. Se o Jogador se encontrar no Chão, sabendo que a velocidade é constante, abrangeram-se os seguintes casos: 1) se a disância percorrida (velocidade * tempo) mais a distância do Jogador, for menor que a distância do Jogador, há uma correcção para a distância real do kmesmo; 2) se inclinação da Peça seguinte for maior ou igual à inclinação da Peça, onde o Jogador se localiza, este mantém o estado em que se encontra; e 3) se inclinação da próxima Peça for menor que a Peça atual, o Jogador encontra-se no Ar,

sendo a sua inclinação a mesma que a peça atual e a sua gravidade 0. Finalmente, quando o Jogador se encontra no Ar, foram abrangidas as seguintes particularidades: 1) se a diferença entre a inclinação do Jogador e a inclinação do Chão for ≥45º e se houver uma interseção entre a reta da peça atual com a recta do movimento do Jogador, este fica morto durante 1 segundo, caso isto não aconteça o mesmo permanece no Chão; e 2) na possibilidade de as rectas não se intersectam, o Jogador permanece no Ar ou desloca-se para a peça seguinte. No último cenário, foi preciso uma função que verificasse qual a possibilidade do Jogador conseguir transitar de uma peça para a outra.

Resultados:

1) Testes

Os resultados obtidos foram positivos, exceptuando nesta situação, relativo ao passo:

```
( 1.1
, [ [ Recta Terra 0
    , Recta Relva 0
    , Recta Relva 0
    , Recta Terra 0
    , Rampa Terra 0 1
    , Recta Terra 1
    , Rampa Terra 1 0
    , Recta Terra 0
    , Recta Terra 0
    , Recta Lama 0 ]
  , [ Recta Terra 0
    , Recta Terra 0
    , Recta Relva 0
    , Recta Terra 0
    , Recta Relva 0
    , Rampa Boost 0 2
    , Rampa Terra 2 4
    , Recta Lama 4
    , Rampa Relva 4 3
    , Recta Relva 3 ] ]
, Jogador {pistaJogador = 0, distanciaJogador = 2.0,
velocidadeJogador = 2.0, colaJogador = 3,
estadoJogador = Ar {alturaJogador = 4.0,
inclinacaoJogador = -30.0, gravidadeJogador = 5.0}} )
```

```
Equality assertion failed:
expected: Estado {mapaEstado = [[Recta Terra 0,Recta Relva 0,Recta Relva 0,Recta Terra 0,Rampa Terra 0 1,Recta Terra 1,Rampa Terra 1 0,Recta Terra 0,Recta Terra 0,Recta Lama 0],[Recta Terra 0,Recta Terra 0,Recta Relva 0,Recta Terra 0,Recta Relva 0,Rampa Boost 0 2,Rampa Terra 2 4,Recta Lama 4,Rampa Relva 4 3,Recta Relva 3]], jogadoresEstado = [Jogador {pistaJogador = 0, distanciaJogador = 2.858251387592478, velocidadeJogador = 1.4938938215281568, colaJogador = 3, estadoJogador = Chao {aceleraJogador = False}}]}
  got: Estado {mapaEstado = [[Recta Terra 0,Recta Relva 0,Recta Relva 0,Recta Terra 0,Rampa Terra 0 1,Recta Terra 1,Rampa Terra 1 0,Recta Terra 0,Recta Terra 0,Recta Lama 0],[Recta Terra 0,Recta Terra 0,Recta Relva 0,Recta Terra 0,Recta Relva 0,Rampa Boost 0 2,Rampa Terra 2 4,Recta Lama 4,Rampa Relva 4 3,Recta Relva 3]], jogadoresEstado = [Jogador {pistaJogador = 0, distanciaJogador = 2.8582513875924778, velocidadeJogador = 1.725, colaJogador = 3, estadoJogador = Chao {aceleraJogador = False}}]}
```

2) Cobertura dos Testes (HPC)

module	Top Level Definitions		Alternatives		Expressions	
	%	covered / total	%	covered / total	%	covered / total
module LI11920	20%	13/62	0%	0/12	12%	4/31
module Main	100%	1/1	63%	7/11	68%	43/63
module Tarefa0_2019li1g088	25%	16/64	11%	14/124	10%	312/3115
module Tarefa1_2019li1g088	85%	6/7	87%	21/24	84%	161/190
module Tarefa2_2019li1g088	0%	0/3	0%	0/60	0%	0/505
module Tarefa3_2019li1g088	0%	0/15	0%	0/46	0%	0/423
module Tarefa4_2019li1g088	100%	10/10	65%	28/43	78%	413/524
Program Coverage Total	28%	46/162	21%	70/320	19%	933/4851

Tarefa 5

Objetivo:

Implementar o jogo completo utilizando a biblioteca Gloss, com o intuito de obter gŕaficos visualmente apelativos.

Desenvolvimento da tarefa:

Começamos por integrar as imagens necessárias ao projecto, reagissem ao tempo, como também os Jogadores (ao todo são 4) e os Robôs. Para o último caso, implementamos funções quando houvesse 1 Jogador e 3 Robôs em jogo, 2 Jogadores e 2 Robôs, e assim consequetivamente, até 4 Jogadores, sem Robôs. Sendo que, a diferença é que o Jogador tem um movimento, designado passo; enquanto que, os Robôs têm a possibilidade de executar qualquer Jogada (Acelera, Desacelera, Cima, Baixo, Direita, Esquerda, Disparar) ou nenhuma.

Sequencialmente, desenvolvemos a parte em que as peças do jogo (Jogadores, Robôs, etc.) interagem com o movimento. Assim, implementamos uma função para os que três cenários possíveis (Menu, Regras e Modo de Jogo) surgissem, dependendo da tecla associada. Por exemplo, se a tecla Home for pressionada, o Menu surgirá no ecrã e no Modo de Jogo é imperativo carregar nas teclas KeyUp, KeyDown, KeyRight, KeyLeft e Enter, consoante da intenção. Também foram criadas funções relativas aos controles de jogo para cada Jogador- sendo que são possíveis 4, estão instituídos controles diferentes, com cada jogada possível. Salienta-se, que à semelhança da reação ao tempo, que também associamos os Robôs, dependendo do número de Jogadores necessários.

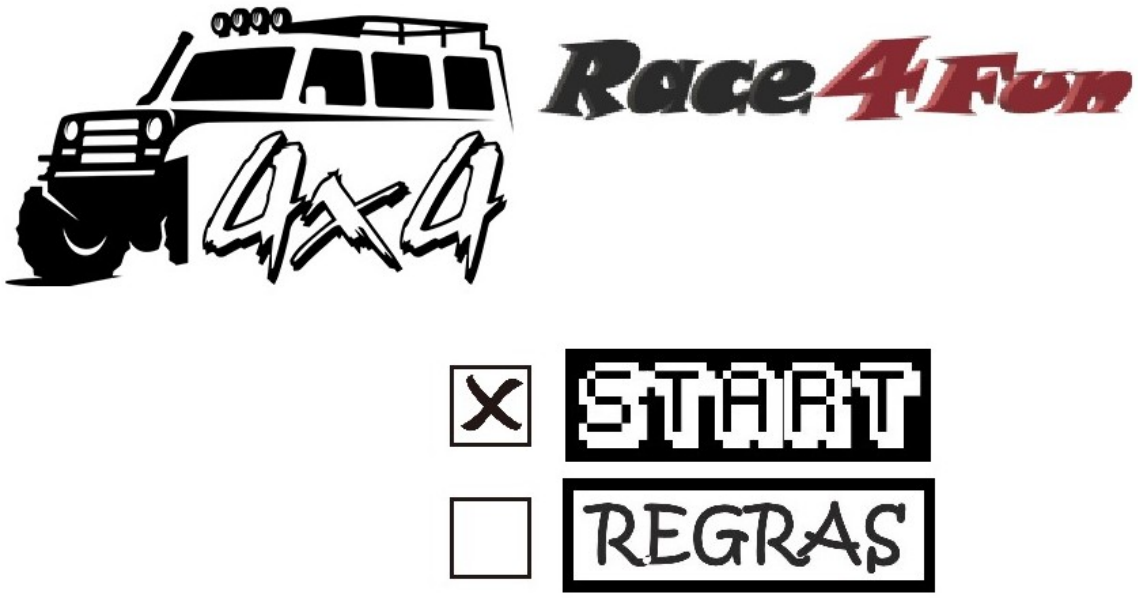
Adicionalmente, com as imagens necessárias, correctamente postas, descrevemos uma série de funções que implementassem o: Menu Inicial; Menu das Regras e a Tabela de Classificações –que contém todas as informações acerca dos Jogadores e muda a sua classificação, dependendo da a posição que se encontra. Relativamente ao mapa, desenvolvemos uma função que tem intenção de desenhar o mapa com os jogadores e outra que cria uma pista com os Jogadores, dessa mesma pista. Também foram necessárias funções que construíssem uma pista, uma rampa e uma recta. Tanto para a Recta e para Rampa, foi tido em atenção o tipo de piso, as formas a implementar e as suas cores. Para a

Rampa também foi relevante a sua altura.

Contudo, não nos foi possível incrementar o método que relaciona o tempo, logo o Robô não efetua nenhuma movimento, caso nenhum dos restantes Jogadores executarem jogadas.

Resultados:

Menu Inicial:



Menu Regras:

Player 1

Desacelerar	Tecla ' , '
Acelerar	Tecla ' . '
Disparar	Tecla ' - '
Cima	Tecla Cima
Baixo	Tecla Baixo
Esquerda	Tecla Esquerda
Direita	Tecla Direita

Player 2

Desacelerar	Tecla ' 1 '
Acelerar	Tecla ' 2 '
Disparar	Tecla ' 3 '
Cima	Tecla ' W '
Baixo	Tecla ' S '
Esquerda	Tecla ' A '
Direita	Tecla ' D '

Player 3

Desacelerar	Tecla ' 6 '
Acelerar	Tecla ' 7 '
Disparar	Tecla ' 8 '
Cima	Tecla ' I '
Baixo	Tecla ' K '
Esquerda	Tecla ' J '
Direita	Tecla ' L '

Player 4

Desacelerar	Tecla ' Z '
Acelerar	Tecla ' X '
Disparar	Tecla ' C '
Cima	Tecla ' H '
Baixo	Tecla ' B '
Esquerda	Tecla ' V '
Direita	Tecla ' N '

Tenha em atenção ao piso aonde se encontra, cada piso tem um atrito especial.

Piso	Cor do Piso	Atrito
Boost	Azul	-0.5
Terra	Castanho Escuro	0.25
Relva	Verde	0.75
Lama	Castanho Claro	1.50
Cola	Vermelho	3.00

Tecla 'home' -> Retorna para o Menu Inicial

MENU

Menu de Modo de Jogo:

Compita com outros jogadores, seja o melhhor.

1 PLAYER

Tenha cuidado... Não se deixe ficar para trás... O mapa não espera

2 PLAYERS

Tenha cuidado... Não se deixe ficar para trás... O mapa não espera

3 PLAYERS

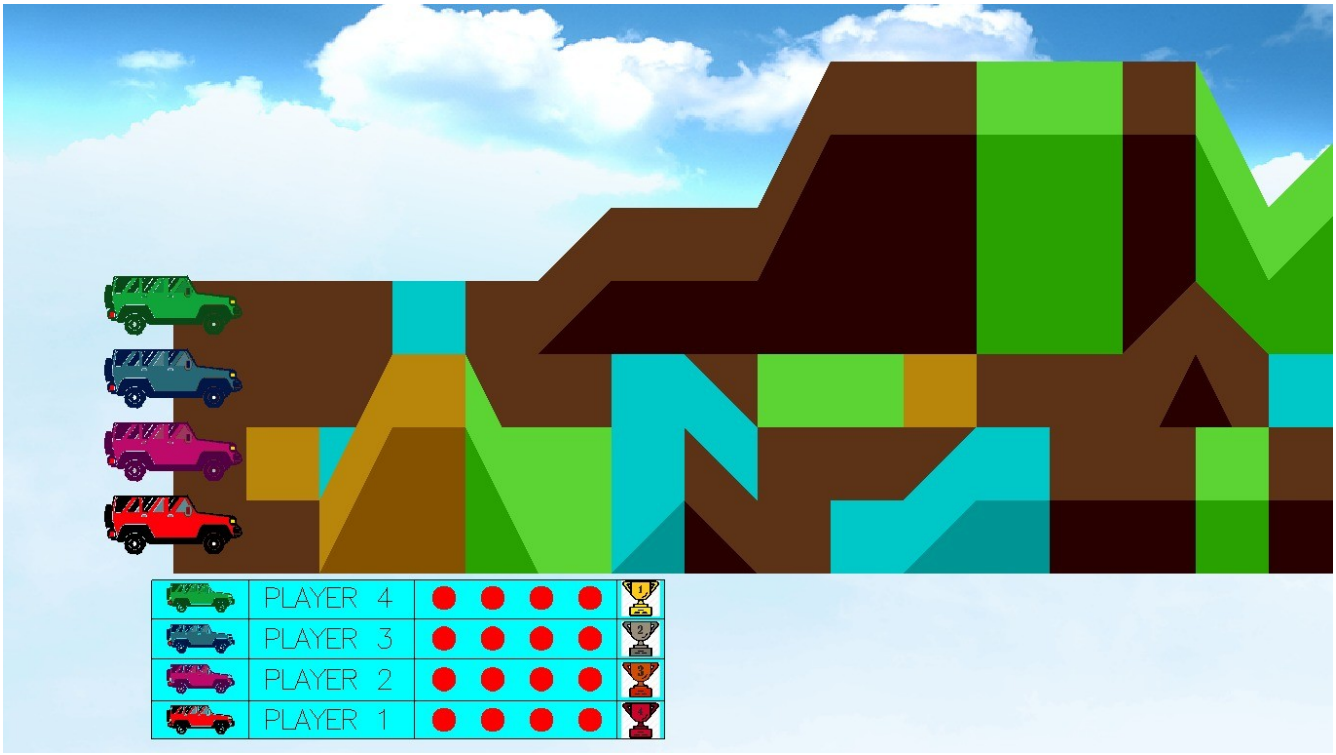
Tenha cuidado... Não se deixe ficar para trás... O mapa não espera

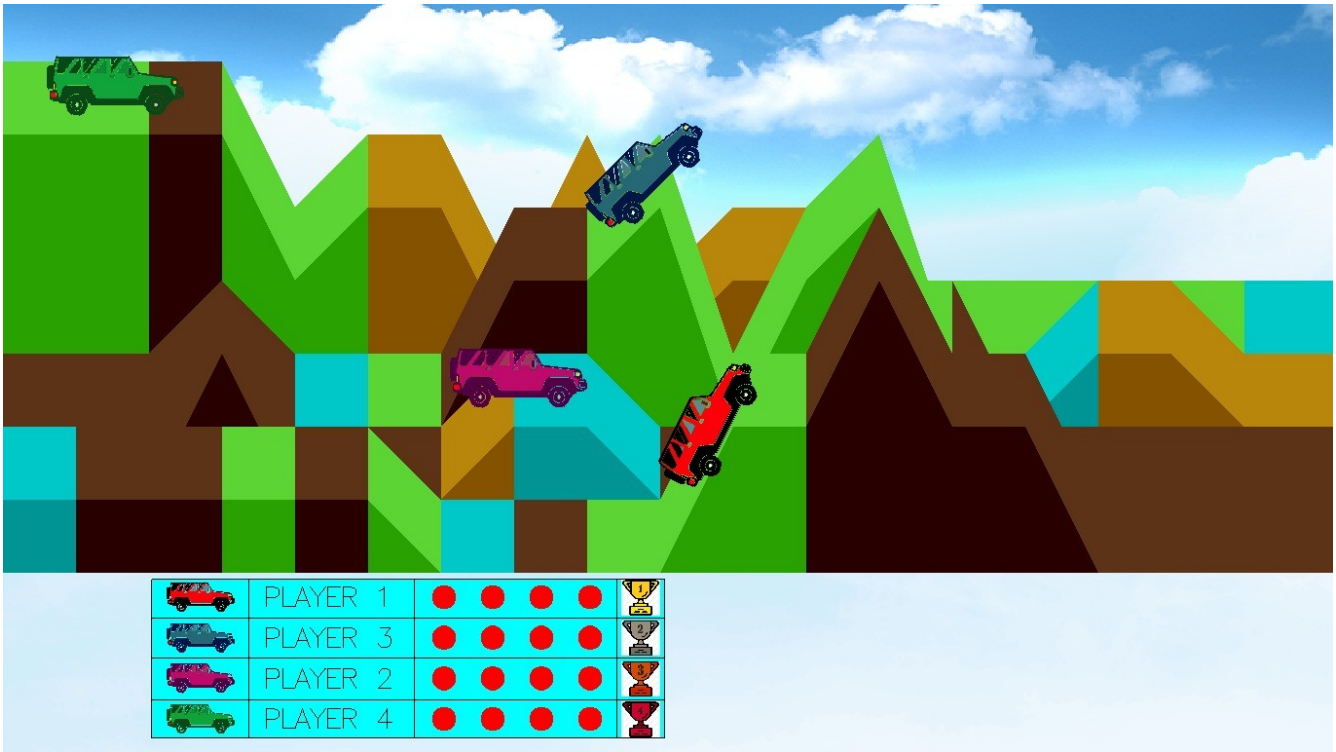
4 PLAYERS

☒ ☐ ☐ ☐

MENU

Race 4 Fun:


PLAYER 4
PLAYER 3
PLAYER 2
PLAYER 1


PLAYER 1
PLAYER 3
PLAYER 2
PLAYER 4

Tarefa 6

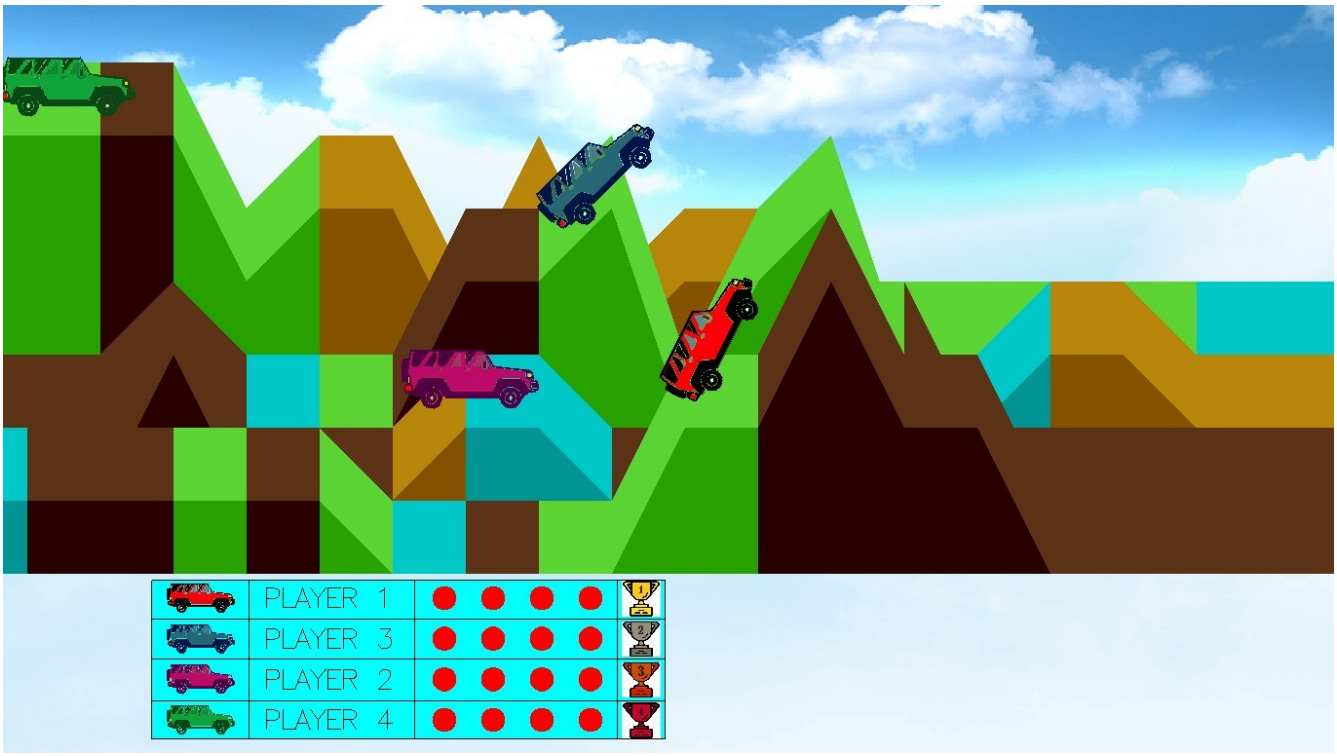
Objetivo:

Construção de robô que joga, de forma autônoma. Sabendo que, em cada momento, o robô tem, unicamente, o conhecimento do estado atual do jogo e pode tomar apenas uma única decisão, utilizando o tipo de jogada, definida anteriormente.

Desenvolvimento da tarefa:

Primeiramente, implementamos uma função em que o robô consegue saber o estado atual do jogo, relacionada com as possíveis jogados que o robô. Ou seja, quando a posição do robô não for válida, este Desacelera; quando se encontra no Chão, Acelera; se estiver Morto, não faz nada;e por último, se o Robô se localizar no Ar, há uma correção da sua inclinação, dependo da da diferença da sua inclinação e da com a inclinação, pois assim garantimos que quando o Robô colidi-se no Chão, não morreria devido à diferença das inclinações.

Resultados:



Conclusão

Após uma reflexão sobre o projeto desenvolvido, concluímos que as metas gerais propostas foram cumpridas. Não obstante ao facto, de alguns objetivos específicos, adjacentes às tarefas, não foram atingidos, como por exemplo na construção de padrões desfazeados, relativos à tarefa 3, apesar de termos recorrido a formas divergentes, para a sua resolução. Como também a falta de perícia na utilização do *Gloss,* onde a maior parte do conhecimento foi adquirido através de informação fornecida pelos tutores e conversas entre pares. Denotou-se alguma dificuldade na execução das tarefas, principalmente aquando do funcionamento anormal do SVN, que acresceu dificuldade na partilha de documentos e na averiguação da qualidade do código.

Porém, devido a originalidade do tema proposto e abrangência de soluções, tornou-se gratificante o desenvolvimento do Projeto, sentindo uma aprendizagem gradual e consolidação dos conhecimentos adquiridos, sobre a linguagem funcional, Haskell. Salienta-se a estimulação sentida, para uma procura ativa e autónoma de novos conhecimentos e ferramentas, com um intuito de potenciar o raciocínio lógico.