Faculdade de Engenharia da Universidade do Porto

Relatório de projecto

Micro Machines

Miguel António de Oliveira Regala | Nº 090509128 | eo09128@fe.up.pt

Vitor Hugo Gonçalves dos Santos | Nº 090509059 | ei09059@fe.up.pt

Turma 3 Grupo 5

Laboratório de Programação Orientada por Objectos

Mestrado Integrado em Engenharia Informática e Computação

6 de Junho de 2011

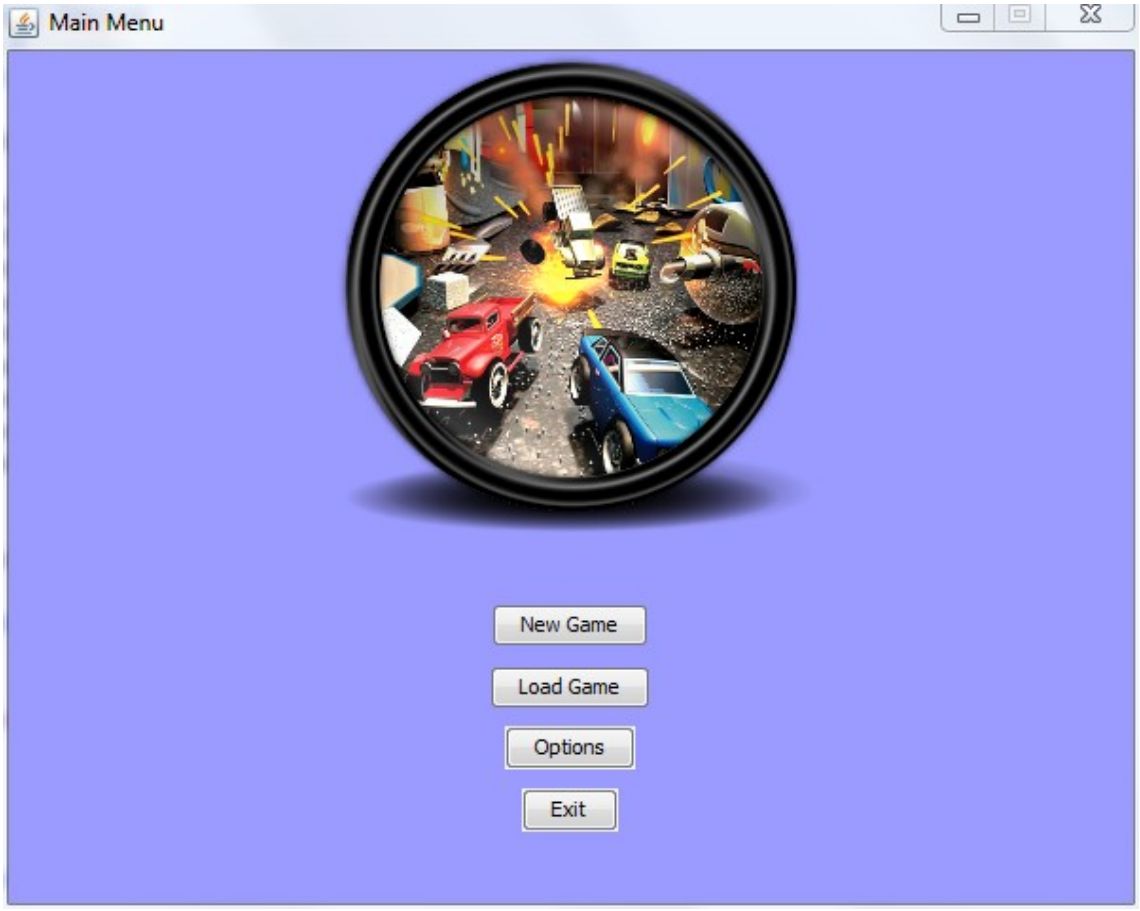
• Objectivos

Os objectivos que se pretendem atingir com este trabalho são os seguintes:

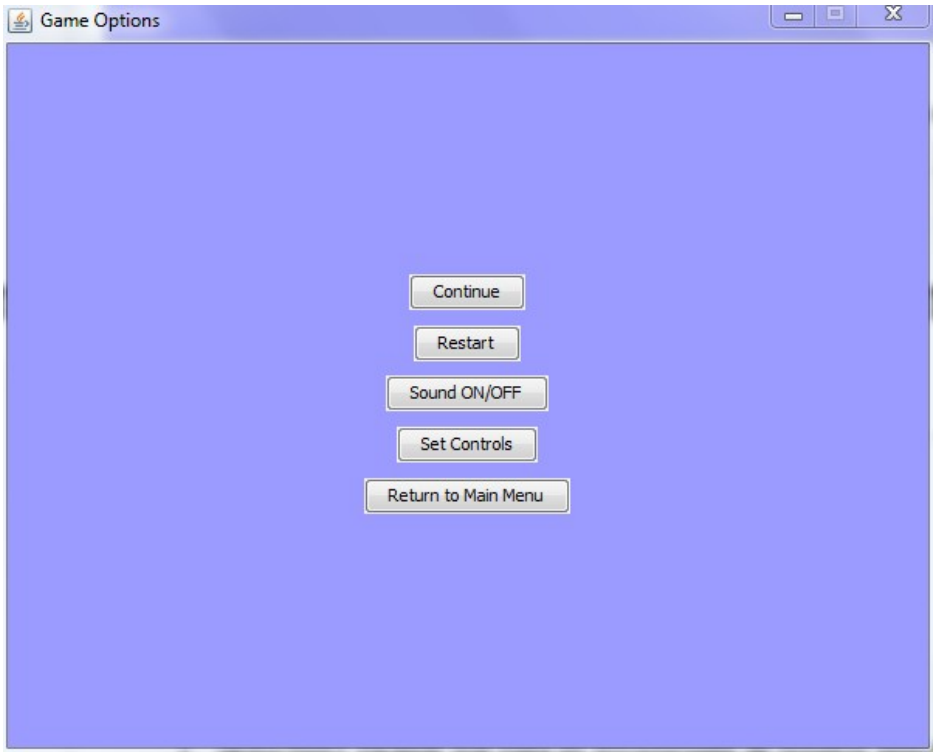
- Aprofundar os conhecimentos, até então adquiridos, em programação em Java.
- Conhecer e utilizar a utilização de padrões de design; de forma a que o projecto final fique o melhor estruturado possível, permitindo a sua melhor manutenção.
- Familiarizar-se com toda a API Swing fornecida com a linguagem, pois mais elementos seus serão utilizados do que no projecto anterior.

Este trabalho consiste na concepção do Micro Machines. É um jogo de corridas, com perspectiva vista de cima, em 2D; do qual os carros terão que correr dentro de uma determinada pista. Estas pistas podem ser carregadas a partir de imagens, sendo que depois disso, é possível definir todas as fronteiras delas, bem como a posição dos carros inicialmente e a meta da corrida.

• Manual de utilização

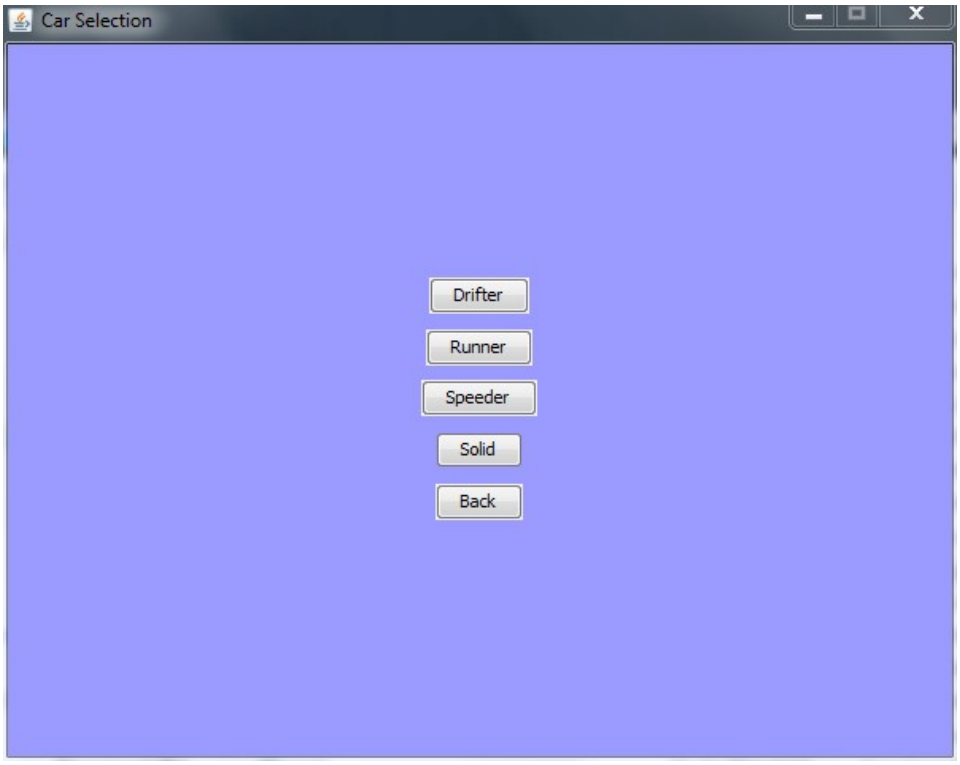


Menu principal: Menu obtido ao iniciar o jogo. É possível iniciar um novo jogo, carregar um anteriormente guardado, aceder ao menu de opções ou sair do jogo.

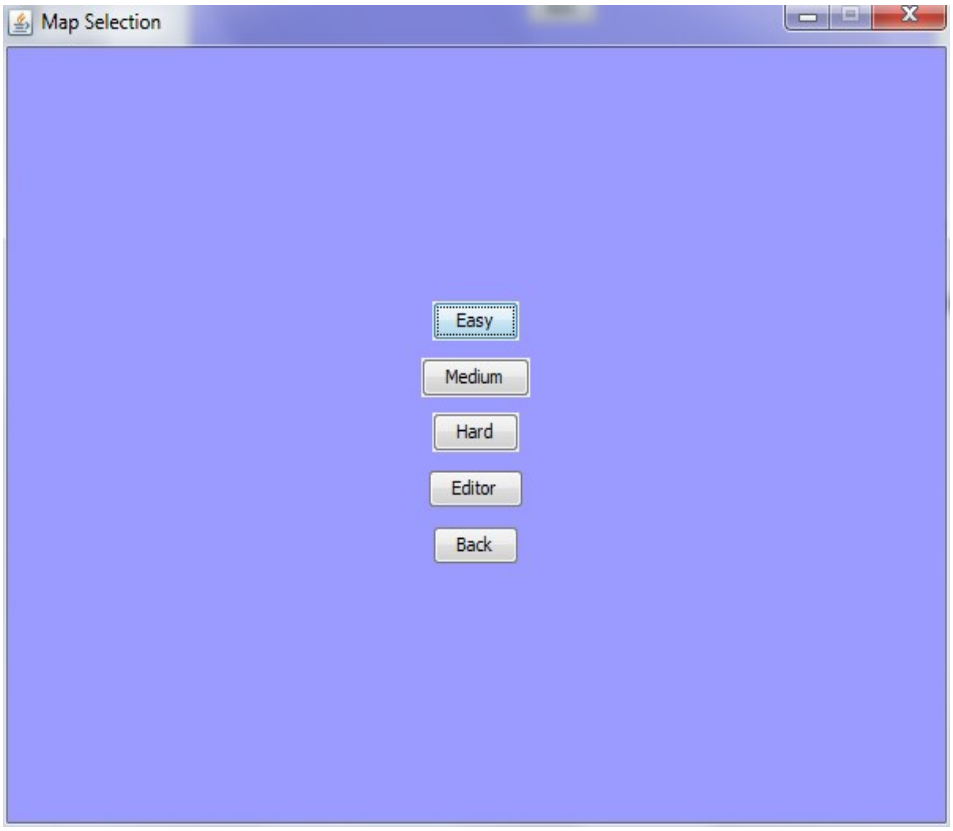


Menu de opções: Ecrã com as opções de jogo existentes. Este ecrã também pode ser acedido a partir de um jogo a decorrer. Permite continuar o jogo ou recomeçar um novo. É possível activar/desactivar o som do jogo, como configurar os controlos. Há a chance, ainda, de regressar ao menu inicial.

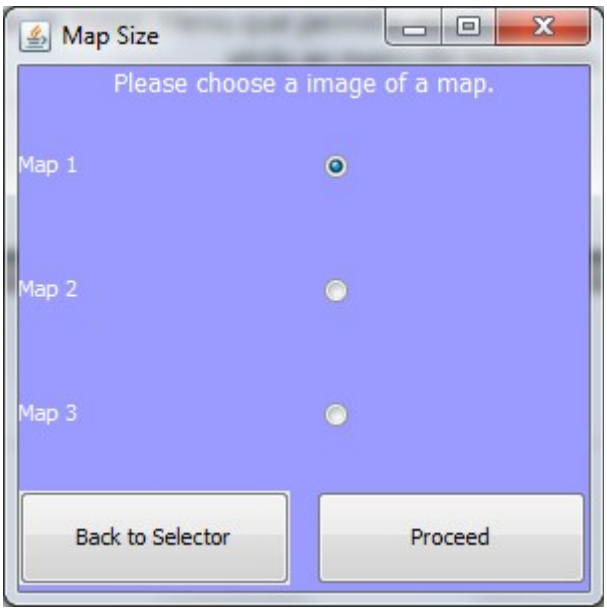
Menu de novo jogo: Este menu permite escolher o carro, o circuito e, no fim, jogar o jogo com essas opções. É possível voltar ao menu inicial.



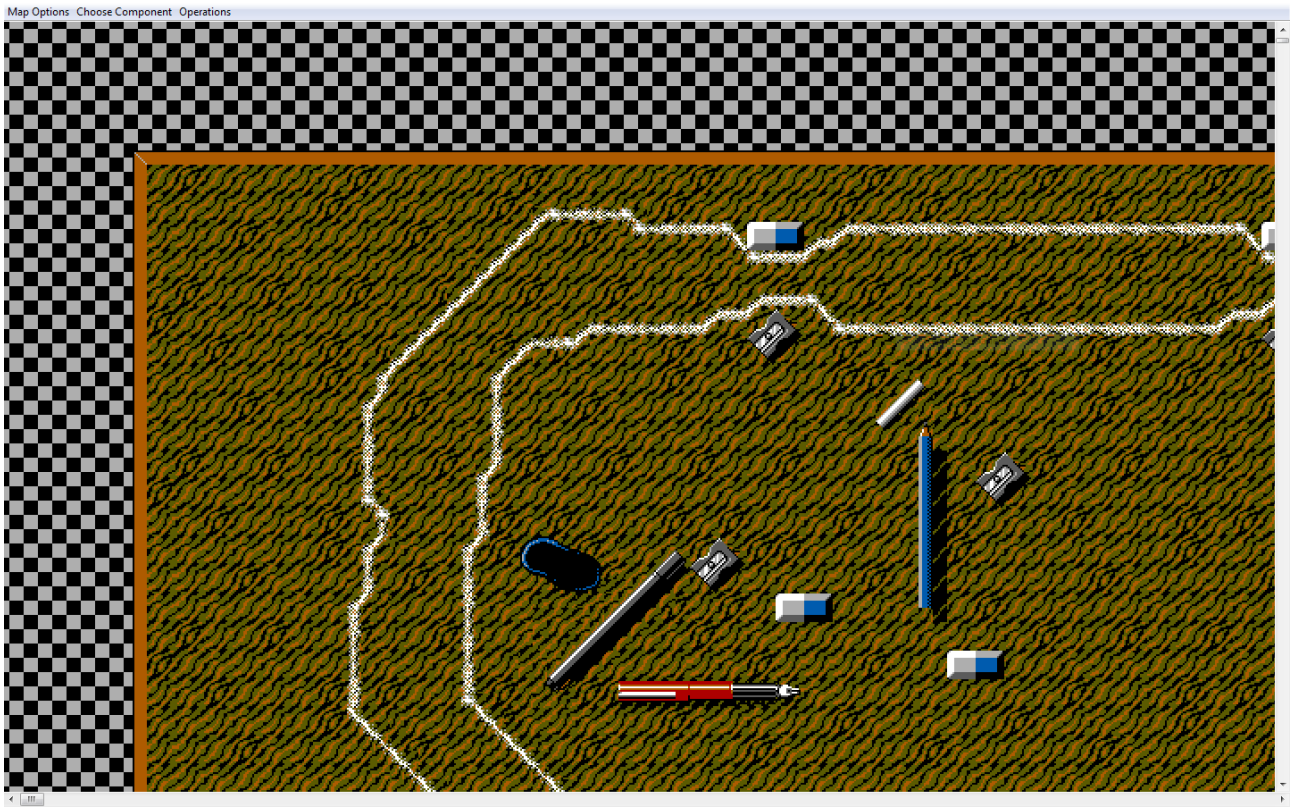
Menu de selecção de carro: Menu que permite escolher o carro pretendido. É possível voltar atrás ao menu de novo jogo.



Menu de selecção de mapa: Menu que permite escolher o circuito pretendido. Ou então, existe o editor de mapas para importação no jogo. É possível voltar atrás ao menu de novo jogo.

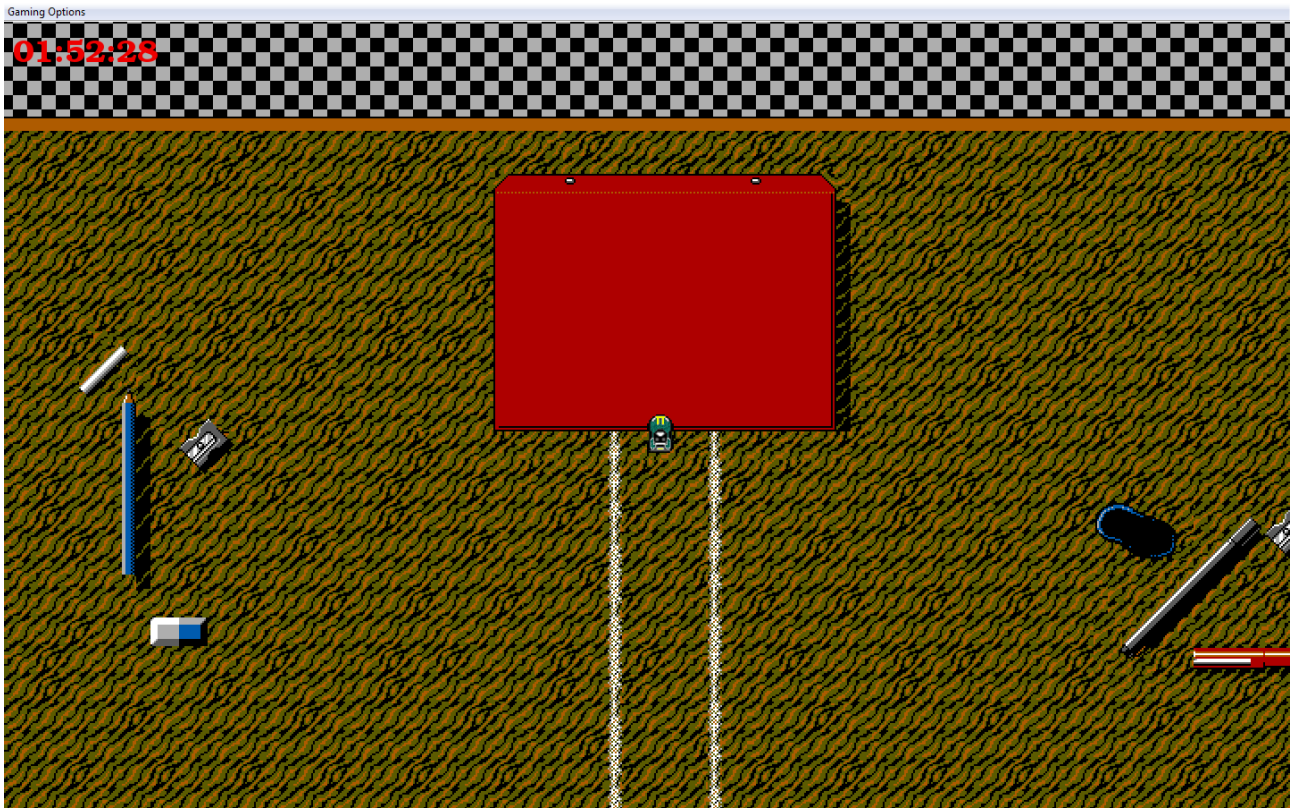


Escolha do editor de mapas: permite escolher uma de 3 imagens representativas de um circuito, e moldá-lo a gosto. Pode ser possível voltar para o menu de escolha de mapas.

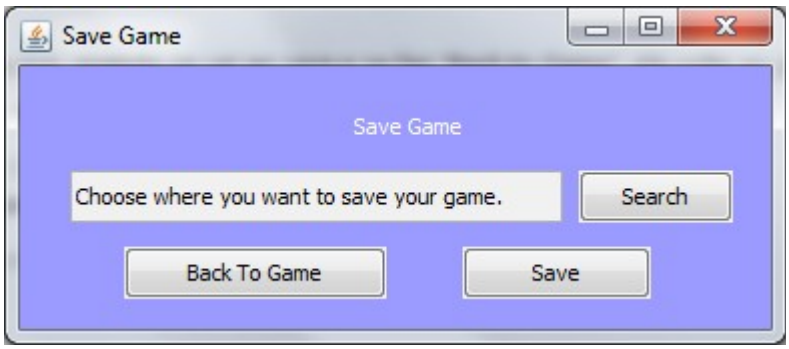


Editor de mapas: Editor que permite colocar todos os elementos necessários de um jogo numa imagem regular. É possível colocar todos esses elementos, anular ou refazer a última opção cancelada/realizada, bem como guardar este mapa para ser jogado. Também se pode voltar para o menu de novo jogo.

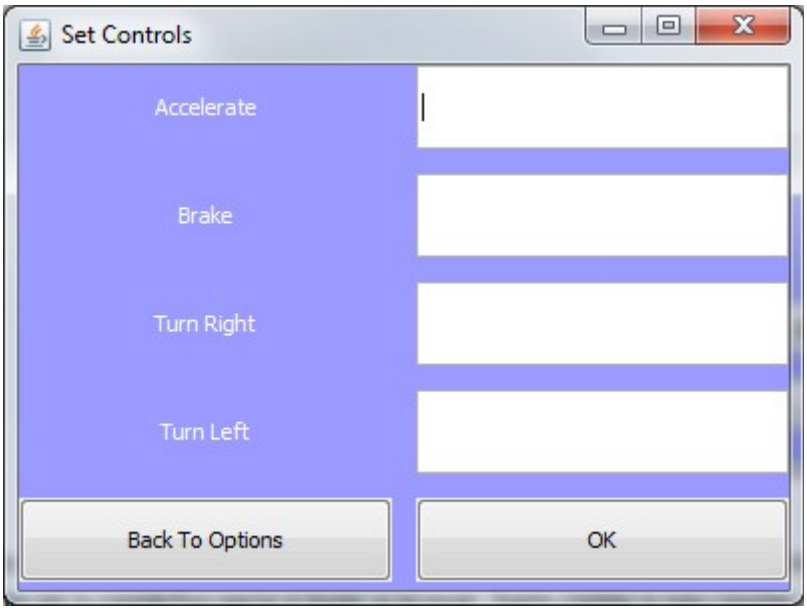
Janela de escrita do nome do jogador: Janela que permite a escrita do nome do jogador, antes de jogar.



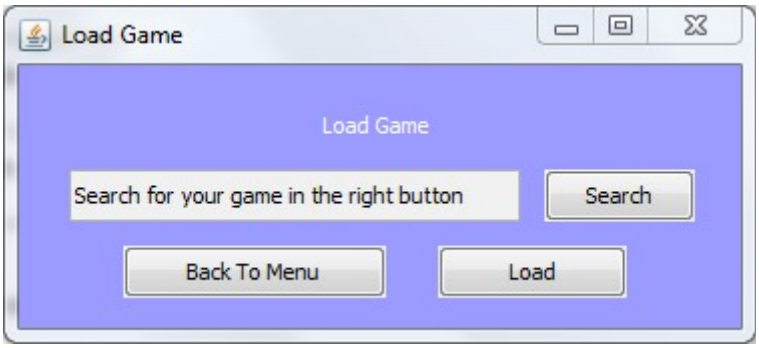
Janela de jogo: Janela do qual o jogo decorre na sua normalidade. É possível pausá-lo, aceder ao menu de opções ou gravá-lo. O jogo contém obstáculos, tem velocidade variável; sendo que quando sai da pista ou cai, o carro é reposto no circuito.



Janela de gravação de jogo: Janela que permite gravar o jogo. Para isso, ao carregar em “Search”, escolhe-se o caminho para o jogo a gravar, bem como o seu nome. No fim, ao carregar em “Save”, aparece uma janela de confirmação, se for gravado com sucesso. Caso contrário, mostra uma mensagem de erro. Contudo, pode-se voltar ao jogo.



Janela de configuração de controlos: Janela que permite configurar os controlos do jogo. Esta opção apenas é possível enquanto decorre o jogo. Pode-se voltar ao menu de opções sem fazer qualquer alteração.



Janela de carregamento de jogo: Janela que permite carregar o jogo. Para isso, ao carregar em "Search", escolhe-se o caminho para o jogo a carregar, ate chegar ao seu nome. No fim, ao carregar em "Load", aparece uma janela de confirmação, se for carregado com sucesso. Caso contrário, mostra uma mensagem de erro. No fim, o jogo é retomado tal como era suposto. Contudo, pode-se voltar ao menu inicial.

• Concepção

• Estrutura de *packages*

Os packages que compõem todo o projecto são os seguintes:

- racing.fileIO: Package que trata da entrada/saída de ficheiros exteriores, para o jogo assim como importação e exportação de mapas.
- racing.gui: Conjunto de ficheiros-fonte responsáveis por toda a interface gráfica do jogo. Assim, ele tratará de todas as funcionalidades do jogo, opções, gravação/carregamento de jogos; bem como o jogo em si.
- racing.gui.frameHandling: Pacote que trata da posição da imagem do jogo (do mapa), conforme um determinado referencial a ser enquadrado.
- racing.gui.imageHandling: Package que trata do carregamento e da utilização de todas as imagens/sprites utilizadas no jogo.
- racing.gui.keyboardHandling: Package responsável pela gravação dos controlos do jogo. Torna-se particularmente útil quando se pretende gravar/carregar jogos, mantendo as configurações.
- racing.logic: Pacote que contém toda a lógica de jogo. Os seus elementos (quer na forma mais genérica ou específica), regras, física de jogo estão presentes.
- racing.physics: Conjunto de código-fonte que traduz a física de jogo. Assim, todos os elementos dinâmicos ficam com um comportamento mais adequado quando existe alguma perturbação do seu movimento.
- racing.sound: Package que inclui sons correspondentes ao que vai acontecendo no

jogo.

- racing.test: Bateria de testes a serem corridos em JUnit.

As bibliotecas usadas para que este programa possa ser corrido com sucesso são:

- java.awt: Bibliotecas AWT, utilizadas como sendo base para a interface gráfica construída com o Swing.
- javax.swing: Biblioteca Swing, do qual a interface gráfica foi criada.
- java.IO: Biblioteca utilizada para gravação/carregamento do jogo, bem como as excepções que possam ocorrer.
- java.util: Inclusões de estruturas de dados básicas como vectores, strings, ...

Estrutura de classes

Cada pacote contém o seu próprio diagrama UML. Esses diagramas podem ser consulados no seguinte ficheiro: LPOO_Proj2_UML.

As ligações entre pacotes podem ser descritas da seguinte forma:

- A package Racing.GUI está directamente ligada à Racing.Logic, pois a primeira serve como uma interface da segunda.
- A Racing.Logic possui ligações com a Racing.Physics, pois precisa da segunda para fazer os cálculos para as colisões e para o próprio movimento dos carros.
- A Racing.FileIO estará ligada à Racing.GUI e à Racing.Logic, pois ela importará tanto as definições do mapa como também será imprescindível para a Racing.Logic poder trabalhar nela.

Comportamento

O comportamento do programa pode ser descrito no diagrama de sequência que está anexado juntamente com o trabalho. O diagrama situa-se em LPOO_Proj2_StateDiagram.

Principais problemas e soluções (padrões) de desenho

Os padrões de desenho utilizados para este trabalho passaram pelos seguintes:

- Singleton: Este problema resolveu um que acontece várias vezes, o de haver apontadores para objectos nulos. Com o Singleton, é garantido que as classes não

tenham quaisquer objectos nulos.

- Strategy: Padrão de desenho que resolveu a problemática dos objectos com uma classe parente em comum. Assim, é possível adicionar novas funcionalidades sem quaisquer complicações.

- Factory method: Padrão que faz com que um determinado objecto possa ser produzido, conforme uma variável pertencente a uma classe produtora. Assim, a questão da memória utilizada fica melhor controlada, pois assim não existem objectos que se encontram à solta.

• Testes

Os testes implementados em JUnit são estes que passam a ser descritos:

- TestCar.java: Bateria de testes que testa todos os movimentos possíveis do carro. Assim, ele testa as orientações, aceleração, marcha-atrás, travagem, salto, velocidade máxima e travagem até estar parado.

- TestCollisionCarObject.java: Conjunto de testes que verificam a colisão de um carro com qualquer tipo de objecto estático. V se ele é totalmente destruído, se muda a sua direcção ou se alguma das suas características (velocidade, resistência) se alteram.

- TestCollisionCars.java: Testes que verificam colisões entre dois ou vários carros. V se a direcção de um dos carros é alterada (e, por isso, tem a sua velocidade reduzida), se ambos os carros ficaram destruídos, se colidem no ar ou na terra, bem como a sua resistência reduzida.

e

- TestRace.java: V sobre as variáveis existentes na corrida em si. V os vencedores de uma corrida, a quantidade de tempo que cada carro está fora da pista, se passa um checkpoint, aumenta ou diminui o número de carros numa corrida, bem como se um carro a acabou. No fim, verifica os pontos e os tempos acumulados num conjunto de corridas.

- TestMap.java: Testes sobre a movimentação da câmara ao longo de todo o mapa. V se ela se encontra fixa numa determinada coordenada ou se encontra móvel segundo um carro tomado como referencial.

- Test.java: Testes que verificam se as colisões nos saltos ou no ar estão a ser executados correctamente.

• Conclusões

Os objectivos deste projecto foram, na sua maioria, cumpridos. No sentido de que o jogo está jogável, permitindo que o jogador possa controlar o carro sem qualquer sobressalto. Pode escolher 4 tipos de movimentação do seu próprio carro, bem como de um mapa, e de o poder editar com o editor.

No entanto, houve algumas funcionalidades das quais apenas estão parcialmente implementadas, não garantindo o seu pleno funcionamento. Essas funcionalidades prendem-se com a gravação/carregamento de jogo, o estabelecimento de uma carreira de jogo e a implementação de mais carros controlados por inteligência artificial.

Para melhorar o trabalho, incluirão maior refinamento na sincronização de procedimentos entre o jogo e os seus menus, bem como a inclusão de uma inteligência artificial para que possam ser incluídos mais carros adversários, bem como a inclusão de um conjunto de corridas que o jogador poderia fazer.