**Faculdade de Engenharia da Universidade do Porto**



**Mestrado Integrado em Engenharia Informática e Computação**

**Laboratório de Programação**

**Orientada por Objetos**

Relatório Final de Projeto

WagOn

**6 de junho de 2016**

**Autores:**

|  |  |  |
| --- | --- | --- |
| 201403090 | Carolina Ferreira Gomes Centeio Jorge | up201403090@fe.up.pt |
| 201404228 | Inês Filipa Noronha Meneses Gomes Proença | up201404228@fe.up.pt |

**Índice**

[1. Introdução 1](#_Toc453063990)

[2. Manual de utilização 2](#_Toc453063991)

[2.1. Funcionalidades suportadas 2](#_Toc453063992)

[2.2. Instalação do Programa 2](#_Toc453063993)

[2.3. Modo de utilização 3](#_Toc453063994)

[2.4. Ficheiros de entrada e saída 4](#_Toc453063995)

[3. Conceção, implementação e teste 5](#_Toc453063996)

[4. Conclusões 6](#_Toc453063997)

[5. Referências 6](#_Toc453063998)

# Introdução

Este documento tem como objetivo descrever o trabalho efetuado pelos seus autores no âmbito do projeto integrado da unidade curricular de Laboratório de Programação Orientada por Objetos.

O referido trabalho consiste no jogo multiplataforma designado WagOn, que tem por objetivo apanhar o maior número possível de frutas com um vagão. O jogo termina quando o referido vagão for destruído por uma bomba ou quando este cair numa das crateras, originada por uma bomba, no chão.

Com este contexto, o presente documento relata as fases de estudo das funcionalidades, implementação e teste da aplicação desenvolvida bem como descreve o processo de instalação e execução da mesma. Por último, serão apresentados os modelos de conceção em UML e os padrões de desenho utilizados.

# Manual de utilização

## Funcionalidades suportadas

A aplicação implementa um jogo onde o utilizador controla um vagão que tem por objetivo apanhar o maior número de peças de fruta que caem aleatoriamente ao longo do écran do jogo. No entanto, esporadicamente, caem também bombas que destroem o vagão caso colidam com ele e abrem crateras no chão nos locais onde caem. Deste modo, o utilizador deve fazer com que o vagão se mova horizontalmente no écran e salte por cima dos buracos no chão enquanto tenta cumprir o objetivo de apanhar as peças de fruta.

Este jogo foi desenvolvido para dispositivos *Android* e para PCs. Nos dispositivos Android, o movimento horizontal do vagão é controlado com o acelerómetro (inclinando o dispositivo para onde se pretende que o vagão se mova) e os saltos são realizados tocando no ecrã. No computador, o movimento do veiculo é realizado com as teclas de cursor (seta esquerda e seta direita) e os saltos são executados clicando no botão esquerdo do rato.

O jogo aqui apresentado, permite ao seu utilizador jogar sozinho ou contra outro jogador. A versão multiplayer (falar sobre diferenças para multiplayer quem ganha, como é efetuada a conecção).

Em qualquer um dos modos, a aplicação permite persistência dos dados guardando a melhor pontuação até ao momento.

Por último, o jogo tem suporte multilíngue (língua portuguesa e língua inglesa) reconhecendo as definições do dispositivo onde corre.

## Instalação do Programa

Antes de mais, é necessário importar o projeto para o *AndroidStudio* (File > New > Import Project), selecionando a pasta onde o projeto se encontra.

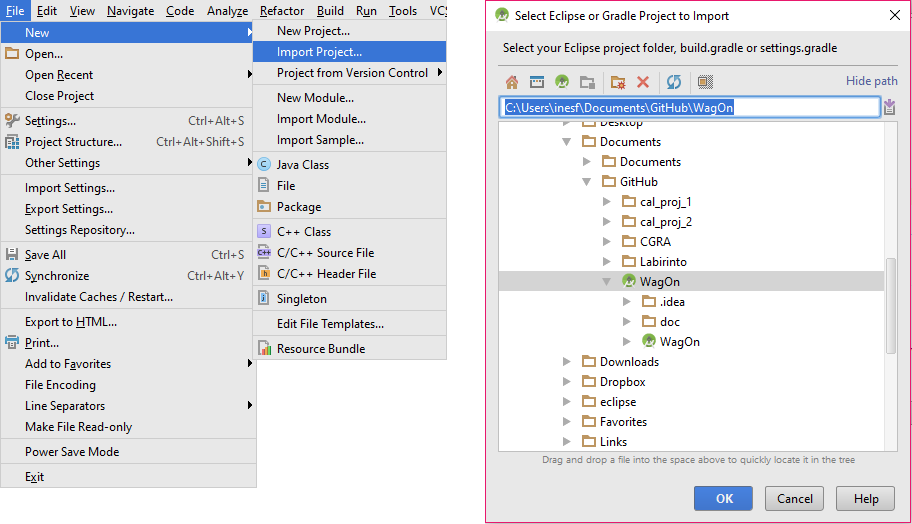


Figura 1: Exemplo de importação do projeto

De seguida, deve-se alterar as configurações de Run/Debug para conter uma configuração do tipo aplicação com as seguintes definições:

* *Main Class*: com.feup.lpoo.desktop.DesktopLauncher
* *Working Directory*: (…)\WagOn\android\assets
* *Use classpath of module:* desktop

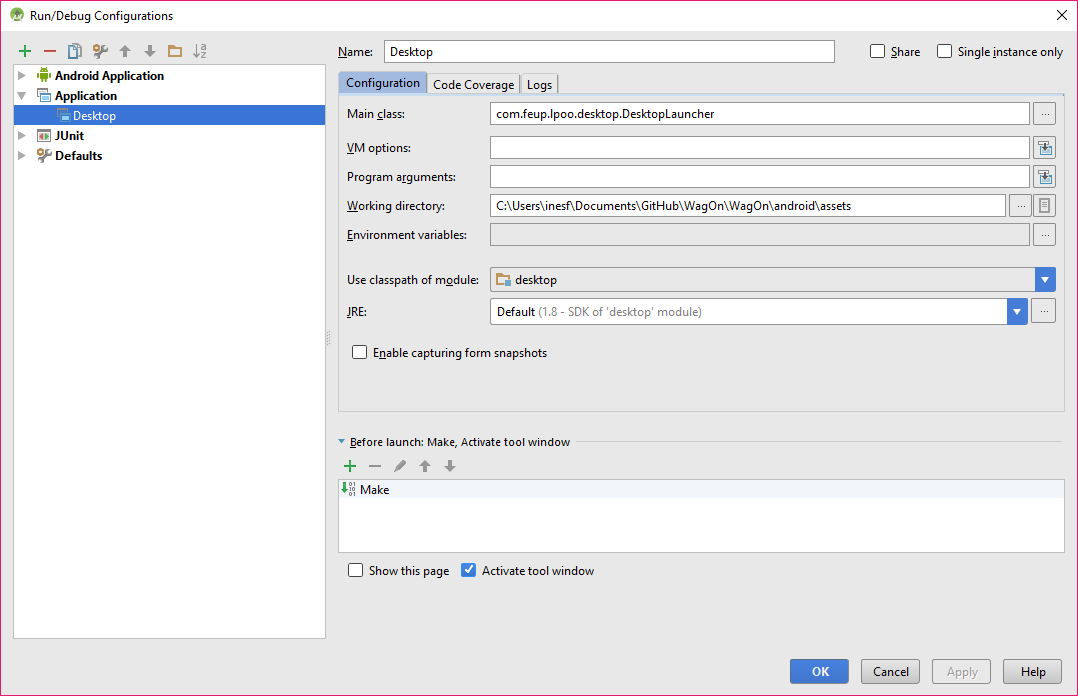


Figura 2: Configurações para correr o projeto em PC

Para executar a aplicação, basta somente selecionar a configuração pretendida (Android ou Desktop) e clicar no botão *Run* na barra de ferramentas do AndroidStudio.

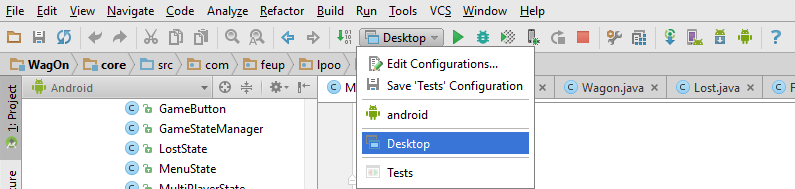


Figura 3: Menu de escolha de configurações e botão Run ao seu lado direito

A ordem de arranque da aplicação é irrelevante para o jogo. No entanto, para jogos *multiplayer* é necessário o PC que correr a aplicação servidor e os dispositivos *Android* onde as duas aplicações cliente irão correr estarem ligados à mesma rede. Além disso, a funcionalidade *multiplayer* deve ser inicializada primeiro na aplicação PC e, posteriormente, nos dispositivos *Android*, de modo a que a conexão entre os dispositivos ocorra normalmente.

## Modo de utilização

* Modo de utilização
* imagens dos ecrãs legendadas e explicação da informação e ações disponíveis em cada ecrã, cobrindo as várias funcionalidades

Ao iniciar o jogo WagOn é apresentado um menu de jogo com três alternativas para o utilizador, da esquerda para a direita, single player, multiplayer e sair. Este pode agora escolher premir o botão correspondente à funcionalidade que pretende.



Figura 4: Ecrã do menu

Caso o utilizador escolha a opção single player será reencaminhado para o ecrã de jogo, onde pode começar a controlar o vagão se que encontra, inicialmente, posicionado ao centro. No canto superior direito pode ver também a sua pontuação (número de melancias apanhadas até ao momento).

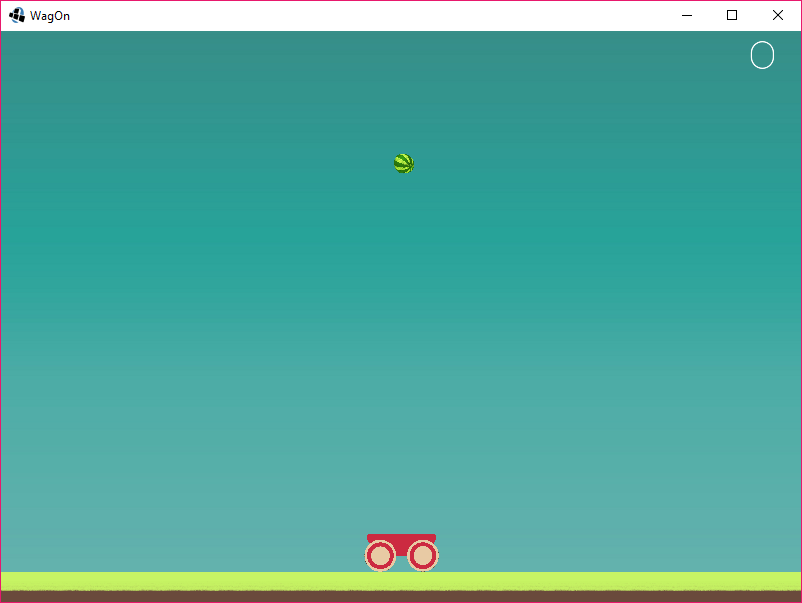


Figura 5: Ecrã inicial do jogo

## Ficheiros de entrada e saída

* Ficheiros de entrada e saída (se aplicável)
* Ficheiros criados / lidos pelo programa e seu formato (se aplicável e relevante para o utilizador)

dddddd

# Conceção, implementação e teste

* Estrutura de packages
* Diagrama de packages UML
* Com packages e dependências
* Será avaliada diretamente a boa utilização de UML para documentação
* Será avaliada indiretamente a boa modularização do programa e separação de camadas
* Tabela com a explicação da responsabilidade/conteúdo de cada package
* Estrutura de classes
* Um diagrama de classes UML por package, incluindo ligações a classes externas de outros packages (escondendo operações e atributos das classes externas)
* Será avaliada diretamente a boa utilização de UML para documentação
* Será avaliada indiretamente a boa utilização de mecanismos de orientação por objetos (abstração, herança, encapsulamento, polimorfismo)
* Tabela com explicação da responsabilidade de cada classe relevante
* Padrões de desenho utilizados
* Indicar que padrões foram usados, onde e porquê
* Inclui também padrão MVC, se usado
* Mecanismos e comportamentos  importantes
* Aproveitar e atualizar o que já produzido para o checkpoint intermédio, no sentido de  explicar um ou mais mecanismos ou aspectos de comportamento importantes do programa com o apoio de diagramas de estado ou de sequência UML
* Exemplos:
* Ciclo de vida da aplicação / diagrama de navegação com diagrama de estados
* Sequência de estabelecimento de ligação em aplicação distribuída
* Basta 1 diagrama dinâmico devidamente explicado
* Ferramentas, bibliotecas e tecnologias utilizadas
* Exemplos: Eclipse ou Android Studio? Usaram libGdx?
* Dificuldades encontradas e sua resolução
* Listas de testes realizados
* Explicar que testes manuais e automáticos foram realizados
* Não são exigidas métricas de cobertura de código ou de mutantes

# Conclusões

* Grau de cumprimento dos objetivos (*achievements*)
* Melhorias possíveis
* Nível de contribuição dos elementos do grupo (percentagem e tipo de contribuição de cada elemento)

# Referências