**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](#_Toc453100195)

[2. Manual de utilização 2](#_Toc453100196)

[2.1. Funcionalidades suportadas 2](#_Toc453100197)

[2.2. Instalação do Programa 3](#_Toc453100198)

[2.3. Modo de utilização 4](#_Toc453100199)

[Menu 4](#_Toc453100200)

[*Single Player* 5](#_Toc453100201)

[Multiplayer 6](#_Toc453100202)

[Pausa 6](#_Toc453100203)

[Fim de jogo 6](#_Toc453100204)

[2.4. Ficheiros de entrada e saída 7](#_Toc453100205)

[3. Conceção, implementação e teste 7](#_Toc453100206)

[3.1. Estrutura de packages 7](#_Toc453100207)

[3.2. Estrutura de classes 8](#_Toc453100208)

[Logic 8](#_Toc453100209)

[WagonStates 9](#_Toc453100210)

[Network 9](#_Toc453100211)

[Test 10](#_Toc453100212)

[States 10](#_Toc453100213)

[3.3. Padrões de desenho 10](#_Toc453100214)

[3.4. Mecanismos e comportamentos importantes 10](#_Toc453100215)

[3.5. Ferramentas, bibliotecas e tecnologias utilizadas 10](#_Toc453100216)

[3.6. Dificuldades encontradas e sua resolução 11](#_Toc453100217)

[3.7. Listas de testes realizados 11](#_Toc453100218)

[4. Conclusões 12](#_Toc453100219)

[5. Referências 12](#_Toc453100220)

# 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* permite a dois jogadores controlarem um de dois vagões que aparecem no monitor do PC (aplicação servidor) com os seus dispositivos *Android* como se estivesse a jogar em modo *single player*. O jogador que sobreviva mais tempo é o vencedor.

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.

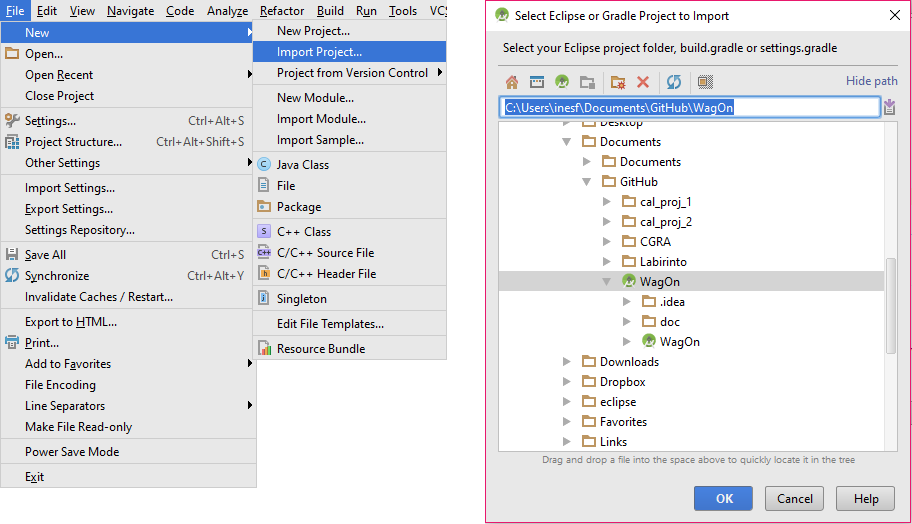


Figura 1: Exemplo de importação do projeto

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

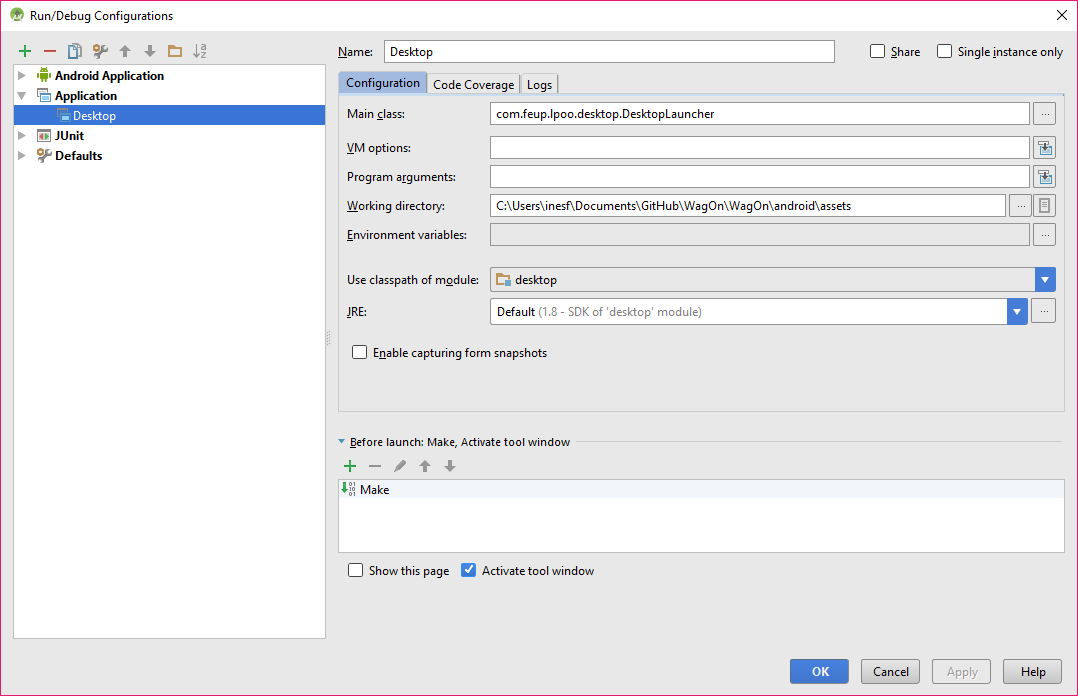


Figura 2: Configurações para correr o projeto em PC

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

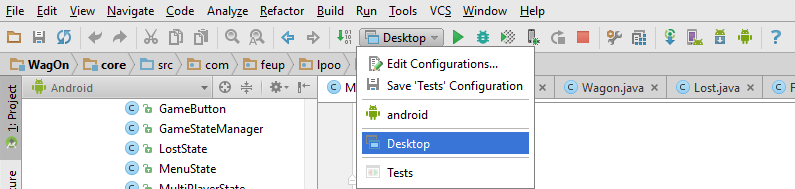


Figura 3: Menu de escolha de configurações e botão Run ao seu lado direito

A ordem de arranque da aplicação é irrelevante para o jogo single *player*. No entanto, 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. Além disso, para jogos *multiplayer* é requisito que o computador servidor e os dispositivos *Android* com as aplicações cliente conectados à mesma rede.

## Modo de utilização

### Menu

Ao iniciar o jogo *WagOn* é apresentado o menu de jogo ilustrado na Figura 4 e que oferece as seguintes três alternativas ao utilizador (da esquerda para a direita): modo single *player*, modo *multiplayer* e sair do jogo. O jogador deve premir o botão correspondente à funcionalidade que pretende.



Figura 4: Ecrã do menu

### *Single Player*

Caso o utilizador escolha o modo *Single* *Player* será reencaminhado para o ecrã de jogo.

A partir de agora pode começar a controlar o vagão que, inicialmente, se encontra, posicionado ao centro. No canto superior direito é apresentada a sua pontuação (número de melancias apanhadas até ao momento).

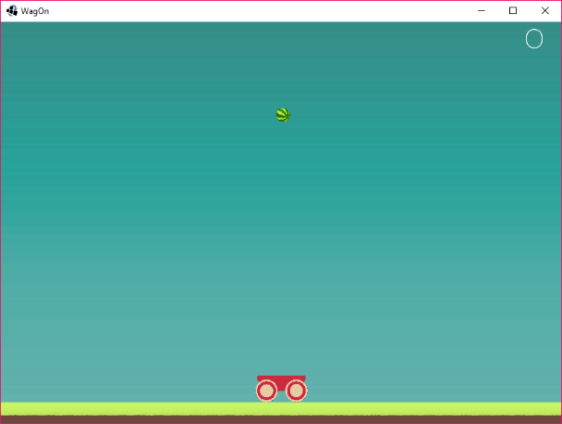


Figura 5: Ecrã inicial do jogo

Para mover o veiculo, o utilizador deverá utilizar as setas do teclado (no caso de estar a correr em PC) ou inclinar o dispositivo *Android*. A Figura 6 ilustra esta situação.

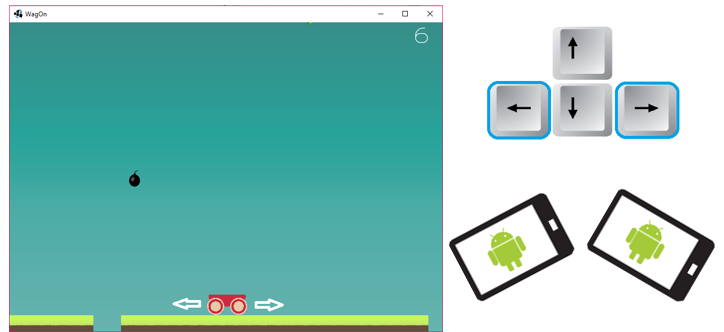


Figura 6: Movimentação do veículo

De modo a evitar que o vagão caia nos buracos, é também possível faze-lo saltar. Para esse propósito o utilizador com dispositivo *Android* deverá clicar em qualquer ponto do ecrã. Os jogadores em ambiente de PC devem clicar no botão esquerdo do rato.



Figura 7: Vagão a saltar

### Multiplayer

Falta fazer isto!!!!!!!!!!!!!!!!!!!

### Pausa

Caso o utilizador tenha de deixar o jogo, para efetuar outras atividades, esta aplicação possui um estado de pausa acedido automaticamente. Para sair deste estado e voltar para o jogo, basta o utilizador clicar no ecrã num local qualquer.



Figura 8: Ecrã de pausa

### Fim de jogo

Quando o jogo termina, é exibido um ecrã com a pontuação obtida e com o *highscore*. Tal como no ecrã de pausa, para sair deste estado e regressar ao menu, é necessário somente clicar na janela.

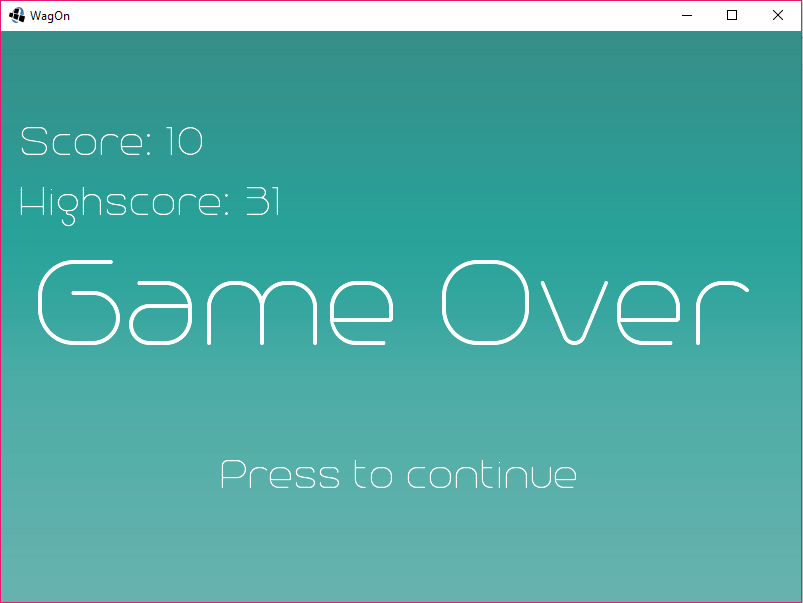


Figura 9: Ecrã de jogo perdido

## Ficheiros de entrada e saída

De forma a garantir a persistência de pontuação máxima, o jogo guarda esta informação num pequeno ficheiro.

Além disso, de forma a adaptar o jogo à linguagem pré-definida no dispositivo, foram criados dois ficheiros de extensão .properties, MyBundle e MyBundle\_pt que armazenam, respetivamente, as mensagens em inglês e português.



Figura 10: Telemóveis com diferentes linguagens

# Conceção, implementação e teste

## Estrutura de packages

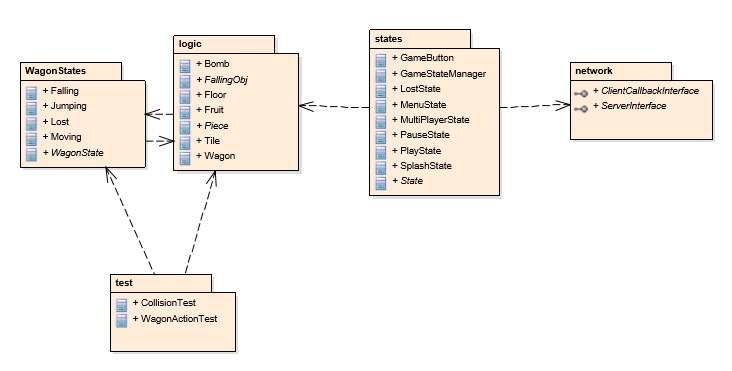


Figura 11: Diagrama de packages

|  |  |
| --- | --- |
| Package | Função |
| WagonStates | Auxilia o package logic que gere os estados do vagão |
| logic | Guarda toda a lógica de jogo |
| states | Realiza a interface com o utilizador e gere os estados do jogo |
| network | Ligar um servidor aos seus clientes |
| test | Realiza os testes unitários |

## Estrutura de classes

### Logic

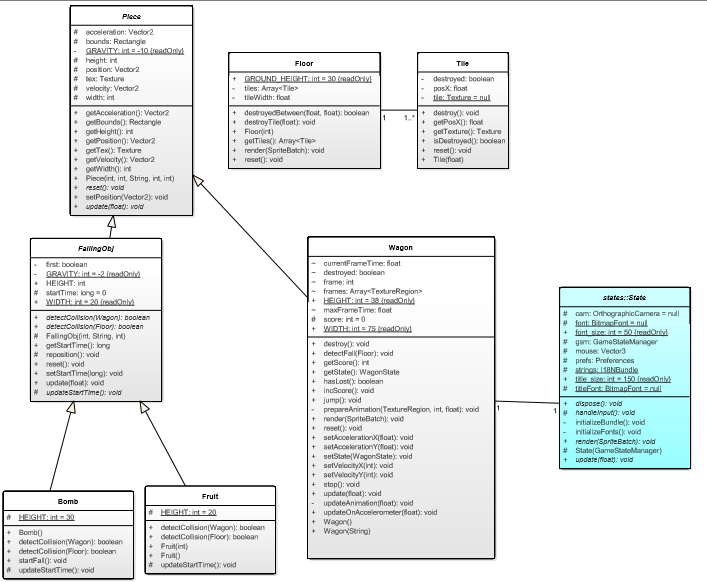


Figura 12: Diagrama de classes do package logic

|  |  |
| --- | --- |
| Classe | Responsabilidade |
| Piece | Generaliza uma peça móvel jogo |
| Wagon | Representa o vagão controlado pelo utilizador |
| FallingObj | Generaliza um objeto que cai do céu |
| Bomb | Representa uma bomba que pode destruir partes do chão ou o veículo |
| Fruit | Representa a fruta que dá pontos ao utilizador |
| Floor | Representa o chão constituído por blocos |
| Tile | Representa um bloco destrutível do chão |

### WagonStates

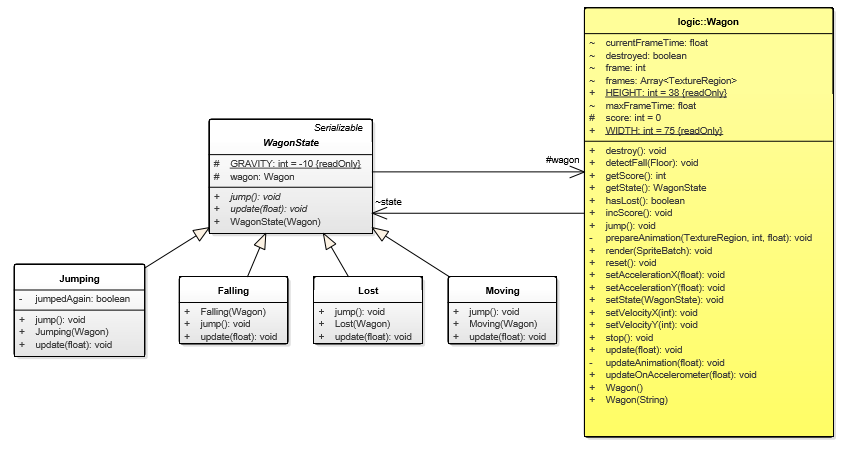


Figura 13: Diagrama de classes do package WagonStates

|  |  |
| --- | --- |
| Classe | Responsabilidade |
| WagonState | Generaliza um estado do veículo |
| Jumping | Gere as movimentações quando o vagão está a saltar |
| Falling | Responsável pela animação quando o veículo está a cair |
| Lost | Bloqueia o controlo do utilizador após o vagão ter sido destruído |
| Moving | Controla os movimentos do veiculo quando rola no chão |

### Network

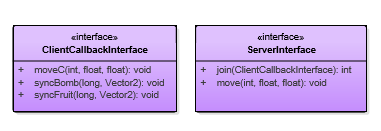


Figura 14: Diagrama de classes do package network

|  |  |
| --- | --- |
| Interface | Responsabilidade |
| ClientCallbackInterface | Responsável por definir os clientes |
| ServerInterface | Define o server e, desta forma, passa informações aos clientes |

### Test

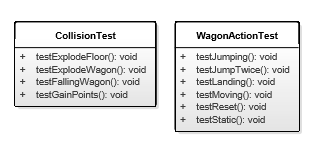


Figura 15: Diagrama de classes do package test

|  |  |
| --- | --- |
| Classe | Responsabilidade |
| ColisionTest | Testa se as colisões estão a funcionar |
| WagonActionTest | Testa se o vagão está a reagir corretamente às instruções |

### States

Falta fazer isto!!!!!

## Padrões de desenho

Neste projeto, fez-se uso do padrão *State* para adequar o comportamento do veículo ao seu estado de acordo com as instruções do jogador.

Foi, ainda, utilizado o padrão *Flyweight* na classe *FallingObj* para gerir eficazmente o processo de alocação de memória e, consequentemente, maximizar a rapidez e eficiência do jogo *WagON*.

## Mecanismos e comportamentos importantes

O diagrama de estados seguinte representa, em alto nível, o mecanismo de transição de entre os diversos estados em que jogo se encontra.

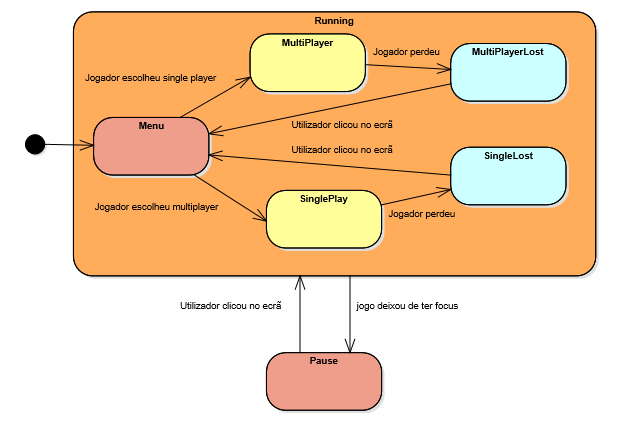


Figura 16: Diagrama de estados do jogo

O diagrama de estados seguinte representa o ciclo de vida do vagão enquanto elemento controlado pelo utilizador.

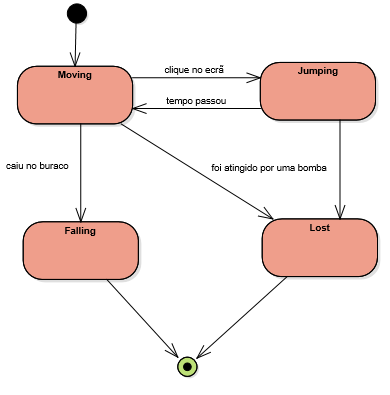


Figura 17: Diagrama de estados da classe Wagon

## Ferramentas, bibliotecas e tecnologias utilizadas

Este projeto foi concebido no IDE *AndroidStudio*, utilizando a biblioteca *libGDX*, para facilitar a implementação da aplicação como sendo multiplataforma e o desenvolvimento do módulo Física do jogo.

Para implementar a comunicação no modo *multiplayer* foi utilizada a biblioteca *LipeRMI*, realizada, deste modo, através da invocação remota de métodos.

## Dificuldades encontradas e sua resolução

Sendo esta a nossa primeira aplicação móvel, aprender como desenvolver um jogo que funcionasse em dispositivos Android e quais eram as ferramentas (IDE’s e bibliotecas) que melhor se aplicavam ao nosso projeto, foi desafiante e trabalhosa.

Além disso, implementação do modo *multiplayer* foi o maior obstáculo encontrado porque as noções que tínhamos sobre comunicações em rede e, particularmente, RMI não era profunda o suficiente para começar a fazer uma aplicação. Posteriormente, observando o exemplo publicado pelo professor e depois de bastante pesquisa, conseguimos implementar uma versão mais simples do que imaginado, mas funcional do jogo.

## Listas de testes realizados

Foram criados testes unitários de modo a testar as consequências das colisões entre os vários intervenientes do jogo e as reações do vagão às instruções do utilizador em diferentes estados.

Os testes de colisões incluem:

* Bomba destrói chão quando colide com este
* Veículo é destruído por colisão com a bomba
* Fruta é reposicionada quando colide com o chão
* Pontuação do vagão é aumentada por consequência da colisão com a fruta

Os testes das ações do veículo são:

* Testar a movimentação horizontal do vagão
* É possível o veículo saltar
* Verificar que o depois de saltar volta ao estado *Moving*
* Testar que só é permitido saltar duas vezes consecutivas.
* Inicialmente o vagão está colocado parado e no centro do ecrã
* Verificar se a reiniciação do veículo está correta

Com estes testes conseguimos atingir a seguinte cobertura de código:

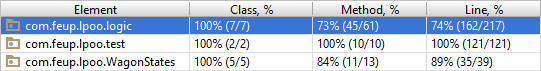


Figura 18: Resultado da ferramenta de cobertura de código

Adicionalmente, também realizamos testes manuais para garantir que a aplicação era robusta e reagia à interação com o utilizador como esperado. Entre estes testes é de se destacar:

* Confirmação da pertinência apresentação de ecrãs de jogo.
* Clicar nos botões do menu e atingir o modo esperado
* Um toque ecrã leva o vagão a saltar
* Clicar nas setas/Inclinar o dispositivo faz o veículo mover-se horizontalmente
* Corroboração da conexão por parte das duas aplicações cliente em dispositivos Android com o servidor em PC.

# Conclusões

Este projeto permitiu a aplicação de competências adquiridas ao longo do semestre da unidade curricular de Laboratório de Programação Orientada a Objetos num contexto real e a melhoria na proficiência de desenvolvimento de aplicações desenhadas para dispositivos móveis, neste caso Android, por parte dos dois elementos do grupo.

Além disso, foi possível cumprir todos os objetivos definidos para este trabalho aquando do checkpoint.

De facto, a sincronização entre o servidor e clientes provou ser o objetivo mais complexo pois depende não só do nosso esforço como também da rede que a suporta e dos dispositivos envolvidos, que pode tornar o tempo de resposta lento, e o jogo pouco interessante. Possíveis melhorias ao jogo poderiam incluir tornar o modo *multiplayer* mais robusto, ou apresentar o ecrã de jogo também dos dispositivos móveis.

Ambos os elementos do grupo tiveram uma participação igual no projeto (50%). A Carolina desenhou a estrutura do jogo, criou as imagens e criou o modo *multiplayer*. A Inês passou adaptou a estrutura do jogo para código que utiliza a biblioteca *libGDX*, implementou as animações, documentou o código e redigiu o presente relatório

# Referências

* ZECHNER, Mario – LibGDX. [Em linha]. 2013. Disponível em: <https://libgdx.badlogicgames.com/> Acesso em: 15 de maio 2016.
* ANDRADE, Felipe Santos – lipermi. [Em linha]. 2006. Disponível em: Acesso em: 5 de junho 2016.
* <https://sourcemaking.com/design_patterns/> <- Melhorar esta referência