

Para a cadeira de Laboratório de Programação Orientada a Objetos, do curso Mestrado Integrado em Engenharia Informática e Computação pertencente à Faculdade de Engenharia do Porto

José Miguel Botelho Mendes up201304828 up201304828@fc.up.pt

António Pedro Araújo Fraga up201303095 up201303095@fe.up.pt

Índice

1. Introdução - pág. 3

2. Manual de utilização - pág. 4, 5, 6

3. Conceção e Implementação - pág. 7, 8, 9

4. Conclusões - pág. 10

5. Referências - pág. 11

Introdução

O objetivo do relatório é poder explicar e abordar através de um método mais formal o trabalho realizado para a disciplina de LPOO. Assim, esperamos que o projeto seja mais claro de entender e que possamos abordar todos os métodos que utilizamos e todos os problemas e respetivas soluções encontradas.

O objetivo do programa é tentar chegar ao fim de cada nível (existem 5) com os recursos dados ao utilizador. Estes recursos consistem em caixas limitadas em número que é necessário colocar ao longo do nível para poder atingir a saída. Ao longo do jogo é possível ligar ou desligar a música, retroceder para o menu anterior (o menu inicial) e até mesmo reiniciar o nível, caso se tenha perdido a possibilidade da resolução do mesmo. A primeira vez que a aplicação é corrida, é mostrado um pequeno tutorial para que o utilizador possa compreender melhor os controlos do jogo.

O relatório está estruturado da seguinte forma: introdução, manual de utilização, abordagens utilizadas, conclusão.

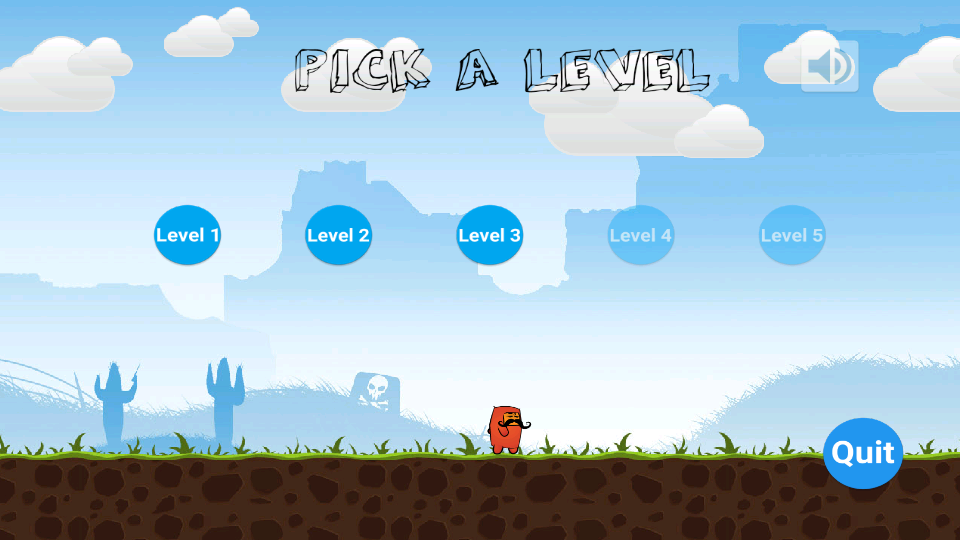
Manual de utilização

O programa apenas necessita de um dispositivo Android para correr, versão 4.2 (Jelly Bean) para cima. Para instalar o programa é apenas necessário ligar o telemóvel ao computador com um cabo USB, ativar a depuração USB no dispositivo, e correr a partir do Eclipse, juntamos também o ficheiro .apk para o caso da aplicação poder ser instalada diretamente no smartphone. Se corrermos a partir do Eclipse, é instalado um ficheiro .apk no telemóvel não sendo a partir desse ponto necessária a utilização do IDE, a não ser para atualizar o código, podendo executar o jogo um número indeterminado de vezes.

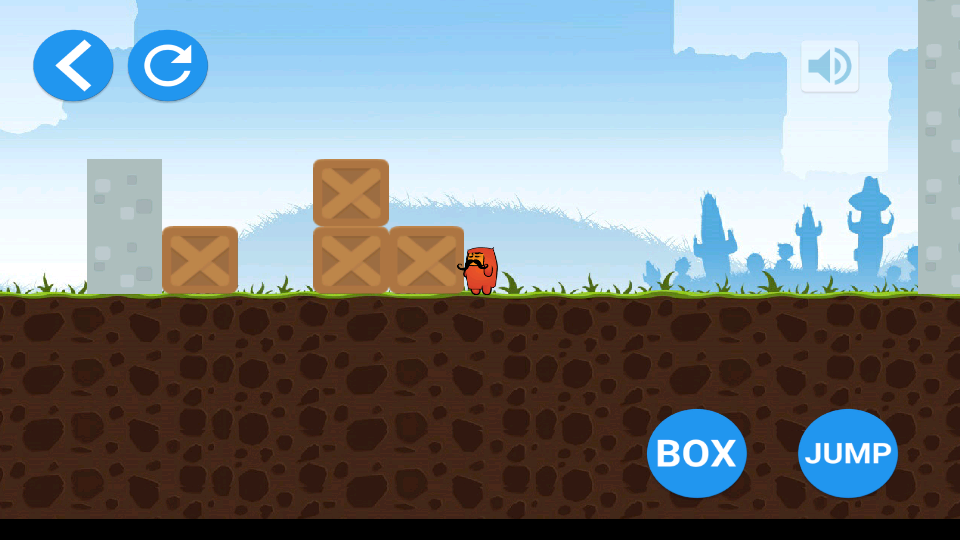
Para arrancar o programa basta apenas selecionar a aplicação que foi instalada e irá surgir um menu como este:



Para utilizar o programa segue-se um breve tutorial em que o utilizador aprende a interagir com o jogo. De seguida, é introduzido um menu com níveis:



É possível saltar, movimentar-se para cada um dos lados apanhar caixas e pousá-las, ligar ou desligar a música, retroceder ou reiniciar o nível, com um breve exemplo a seguir:



No final de cada nível aparece a seguinte mensagem:



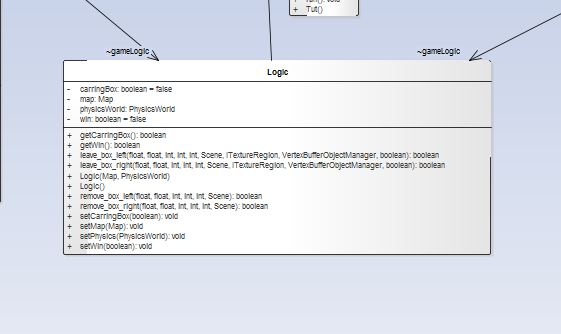
Os ficheiros utilizados são MP3, para a música, JPG e PNG, para as imagens e sprites e TTF para as fontes das letras.

Conceção e Implementação

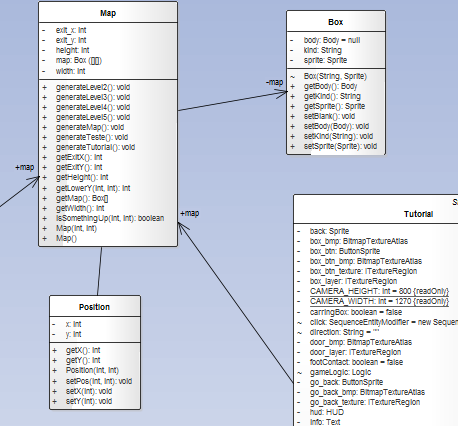
O jogo encontra-se divido da seguinte forma, no que toca a packages:

* Lógica de jogo

Aqui encontra-se tudo o que gere a lógica de jogo, como apanhar e remover caixas. A lógica de jogo garante que não haja erros no que toca a manusear os elementos disponíveis no mapa:

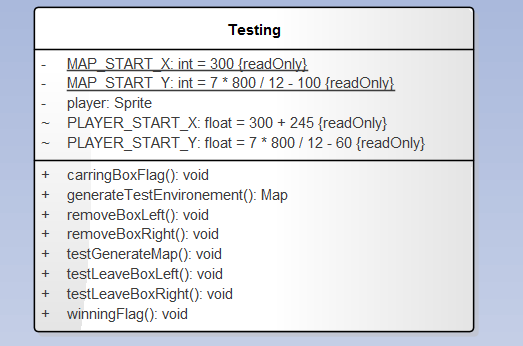


* Elementos do jogo

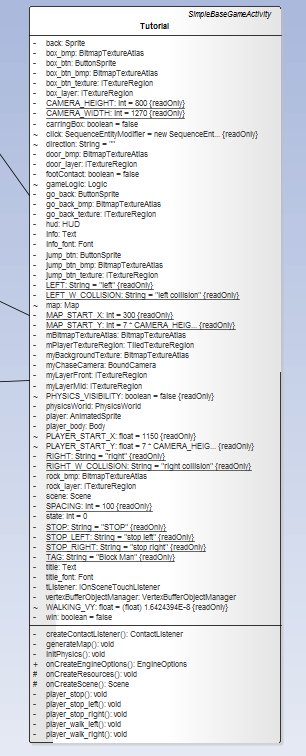
 Neste package encontram-se classes como a classe Box (que cria caixas, paredes e "vazio"), Map que cria o mapa do jogo e a classe Position que cria posições para cada elemento do jogo:

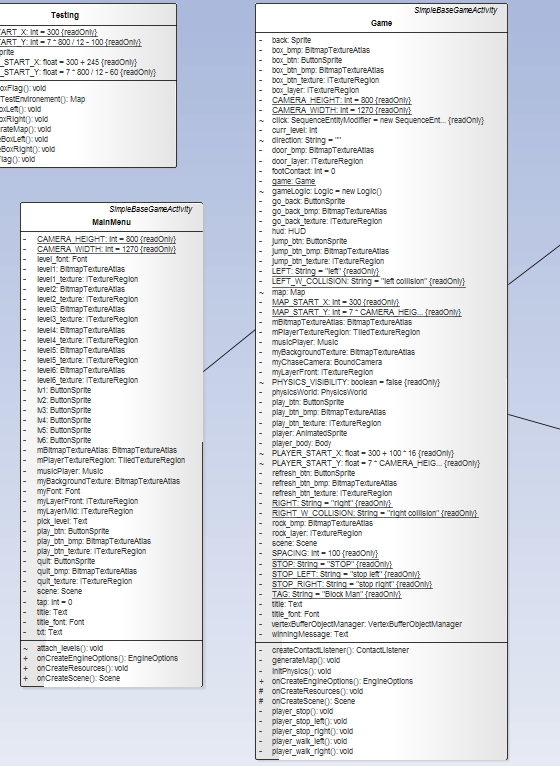
* Testes

Neste package estão todos os testes desenvolvidos com cobertura de 68,8%, de acordo com o JUnit:



* Menus de jogo

 Aqui encontram-se os menus, como MainMenu, que gere o menu inicial e o menu de escolha de níveis, Game, que interage com cada nível que está a ser jogado e possibilita o término de cada um, e, por fim, o Tutorial que inicia apenas uma vez, quando a aplicação é iniciada pela primeira vez num dispositivo novo:



Há quatro tipos de classes, aquelas que extendem uma activity e que são utilizadas para criar uma cena (scene) que vai ser exebida no ecrã do dispositivo Android agrupadas no package com.blockman.game, as classes que guardam dados (que formam objetos) agrupadas em com.blockman.data, são exemplos as classes “Box” e “Map”, a classe de testes uniários em com.blockman.test e a classe de lógica em com.blockman.logic.

Noutro ponto, não utilizamos padrões de desenho ("Design Patterns") pois ao longo do nosso projeto nunca nos deparamos com problemas desses nem vimos a necessidade de os incluir em algum problema.

Um dos mecanismos importantes que utilizamos foi o motor de jogo AndEngine e algumas das suas extensões, AndEngineDebugDrawExtension, para uma mais fácil compreensão de toda a física que estávamos a adicionar no jogo, e o AndEnginePhysicsBox2Dextension, uma extensão que nos permitiu adicionar toda a física ao nosso jogo. Usamos também sprites, música, texturas, uma câmara que segue o jogador e um sensor para detetar colisões, como por exemplo a colisão do jogador com o chão ou com uma caixa, só assim é que era permitido ao utilizador saltar ou apanhar caixas.

Uma das principais dificuldades encontradas foi a implementação de testes unitários, visto que a lógica representa apenas uma pequena porção do código desenvolvido para este jogo, contudo implementámos alguns testes. Não conseguimos também correr o teste de mutações, visto que o *Pit Mutation Test* não corre em código desenvolvido para Android.

No ínico do desenvolvimento deste projeto começámos por usar o *Android Studio*, mas mais tarde, após não conseguirmos adicionar as extensões de física nesse IDE, vimo-nos obrigados a desenvolvê-lo em Eclipse.

Conclusão

Para concluir, temos a dizer que gostámos bastante do primeiro projeto desenvolvido em Android, e sem dúvida que é uma área em que gostaríamos de voltar a trabalhar no futuro. Apesar das dificuldades sentidas, sentimos que conseguimos contorná-las.

Depois de termos começado a habituar-nos à linguagem Java no ínicio deste semestre, chegamos ao final muito mais à vontade em desenvolvê-la, e para concluir achamos que a disciplina em causa (LPOO) tem sido uma mais valia, incluindo a aprendizagem de conceitos importantíssimos na área de Informática e Computação, à espera que estes sejam úteis no futuro.

Referências

AndEngine: https://github.com/nicolasgramlich/AndEngine

AndEngineDebugDrawExtension: https://github.com/nazgee/AndEngineDebugDrawExtension

AndEnginePhysicsBox2DExtension: https://github.com/nicolasgramlich/AndEnginePhysicsBox2DExtension