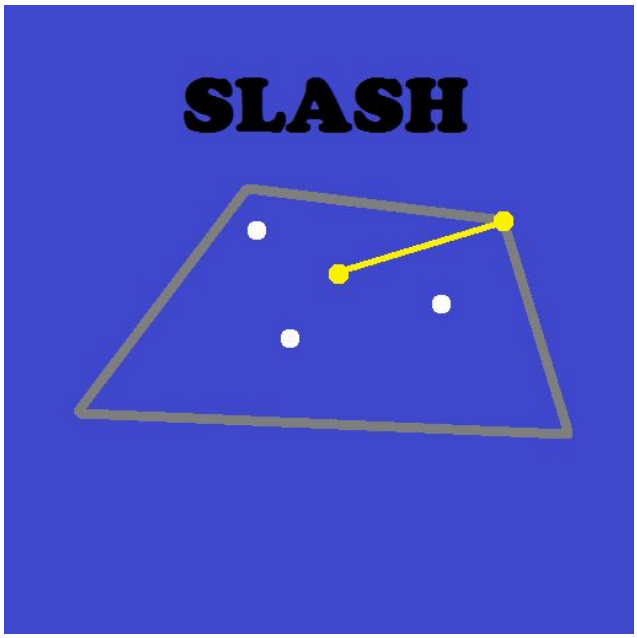
U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Laboratório de Programação Orientada por Objetos

MIEIC



07/06/2016

Turma: 2MIEIC03

Autores:
Diogo Serra Duque (up201406274)
João Pedro Antunes Pereira Gomes (up201403275)

Índice

Índice 1

Introdução 2

2. Manual de Utilização 3

- 2.1 Funcionalidades suportadas 3
- 2.2 Instalação e arranque do programa 3
- 2.3 Modo de utilização 3
- 2.4 Ficheiros 3

3. Conceção, Implementação e Teste 4

- 3.1 Estrutura de packages 4
- 3.2 Estrutura de classes 4
- 3.3 Padrões de desenho utilizados 4
- 3.4 Mecanismos e comportamentos importantes 4
- 3.5 Ferramentas, bibliotecas e tecnologias utilizadas 4
- 3.6 Dificuldades encontradas e sua resolução 4
- 3.7 Lista de testes realizados 4

4. Conclusões 5

5. Referências 6

1. Introdução

Este relatório pretende mostrar e explicar o que é o Slash, projeto realizado no contexto da UC de Laboratório de Programação Orientada por Objetos.

O Slash é um jogo para 1 jogador (há a possibilidade de 2 jogadores, mas só é possível no computador) com o objetivo de mostrar os resultados da aprendizagem dos seus autores à UC de LPOO.

O relatório está estruturado de forma simples e concreta para que a informação seja fácil de encontrar e esclarecedora, contendo várias imagens com este mesmo propósito. Este relatório contém um manual de utilização com screenshots de forma a tirar quaisquer dúvidas, para além da estrutura geral do código com recurso a diagramas UML.

NOTA: Tendo em conta que a estrutura de código e do jogo da versão android e desktop são semelhantes, a maior parte do relatório irá mostrar informação apenas relativamente à app. No entanto, em alguns casos especiais, irá mencionar-se a versão desktop. Ainda assim, é de salientar que são **projetos diferentes** tendo em vista **plataformas diferentes**.

2. Manual de Utilização

2.1 Funcionalidades suportadas

Este jogo permite 2 modos de jogo: 1 jogador em android, ou 2 jogadores num computador. Apesar de ambos os modos terem jogabilidade semelhante, contêm algumas diferenças em termos de tempo de jogo.

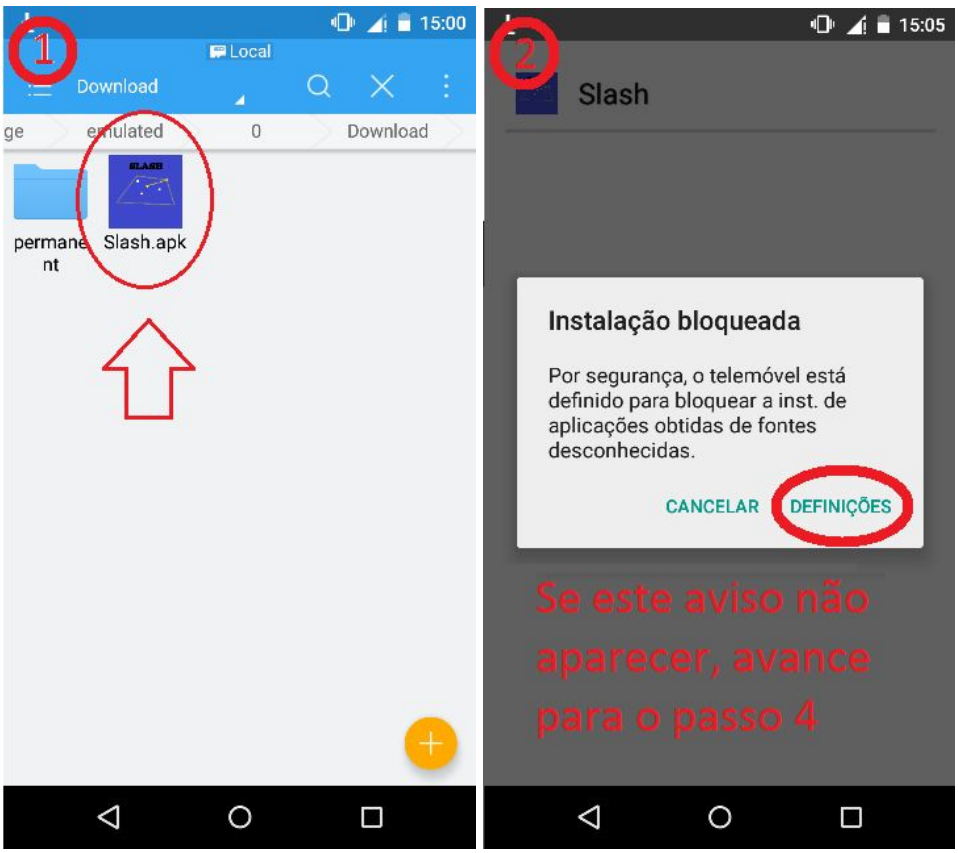
Durante o decorrer do jogo, existe uma musica de fundo.

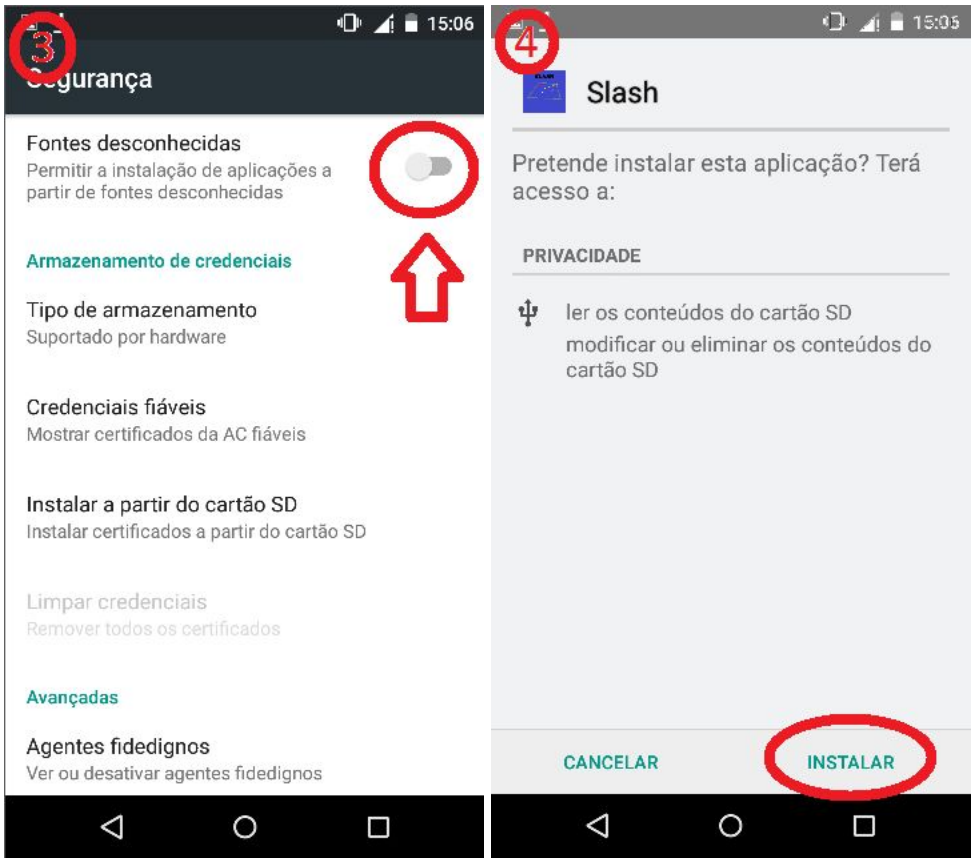
Existe também uma funcionalidade automática que guarda a pontuação mais alta atingida no dispositivo android, pontuação essa que é mostrada quando se inicia a app, enquanto o jogo decorre e também no menu de Game Over. Deste modo, o utilizador pode sempre saber a pontuação mais alta.

E por último, há uma animação escondida no jogo sob a forma de *easter egg*. É mencionado em 2.3 como lá chegar.

2.2 Instalação e arranque do programa

Para a instalação da app, faça download do instalador e siga os seguintes passos:





A app está agora instalada!
Para iniciar o Slash, só tem de o procurar na *App Drawer* e clicar no seu simbolo.

Para a instalação da versão desktop, deve fazer download do *Node JS* e instalar. De seguida, deve executar o ficheiro *index.js* presente na pasta *server* com esta mesma ferramenta, onde deverá aparecer uma mensagem a indicar que o servidor está a correr.

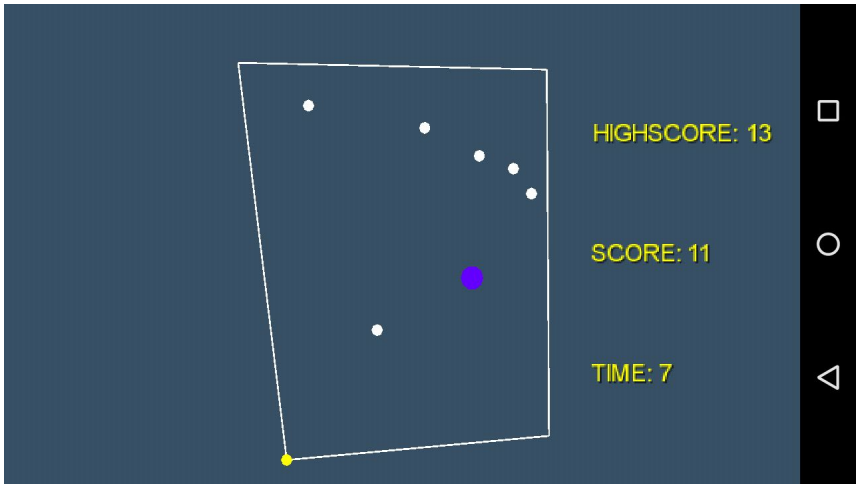
Agora basta abrir o Android Studio e executar a versão desktop do projeto 2 vezes (uma para cada jogador).

2.3 Modo de utilização

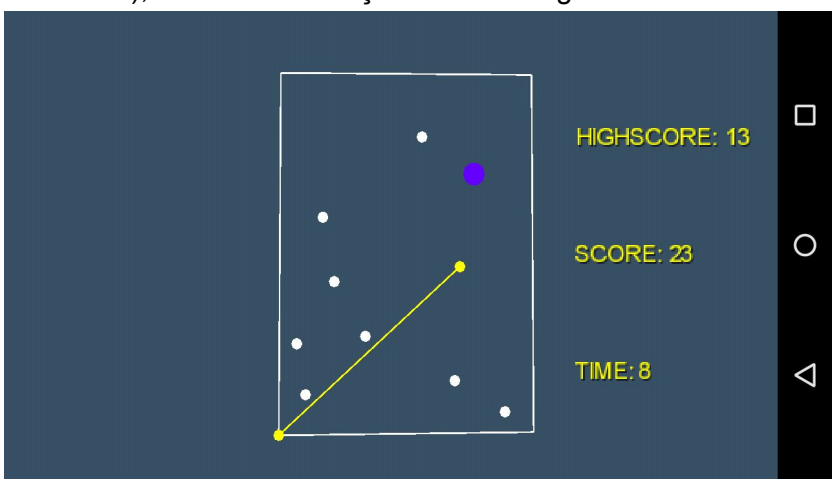
Da primeira vez que iniciar, aparecerá um ecrã semelhante a este:
NOTA: Caso o utilizador carregue em 3 sitios consecutivos (que não incluam o botão de jogar), em que os toques são cada vez mais à direita, aparecerá uma animação como *easter egg*.

Caso já não seja a primeira vez e o utilizador já tenha jogado, a sua melhor pontuação será diferente de 0:

Quando o utilizador carrega no botão “PLAY GAME”, inicia o jogo.
Do lado direito pode ver-se a melhor pontuação, a pontuação atual e o temporizador.



O utilizador deve fazer um corte na caixa antes de o temporizador acabar. Caso isso não aconteça, o jogo acaba. O utilizador sabe que o corte pode ser feito colocando o dedo no ecrã numa posição em que seja desenhada uma linha amarela desde o Slasher (ponto amarelo) até uma parede da GameArea (esta linha tem a direção do Slasher para a localização do dedo). Quando o utilizador retirar o dedo (supondo que a linha amarela estava a ser desenhada), o Slasher começará a cortar a gameArea.



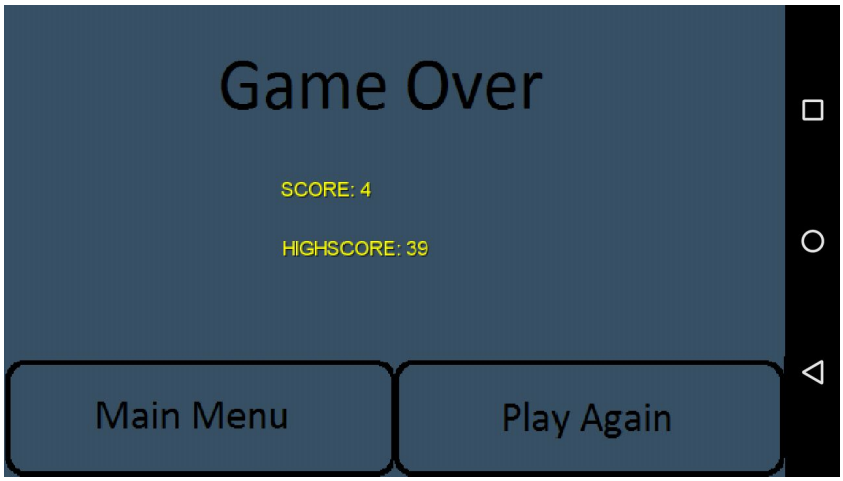
Se o Slasher conseguir atingir o lado oposto da GameArea, esta mesma é recriada (e redimensionada, caso fique muito pequena), o temporizador recomeça, e o score aumenta de acordo com a formula *score += 1 + nrBolasEliminadasComOCorte*. Se antes de atingir o lado oposto da GameArea o Slasher colidir com uma bola, o jogo acaba.

- NOTA: A partir de uma dada pontuação, começa a aparecer um Redirecter (circulo roxo). Caso alguma bola se aproxime deste Redirecter, a sua direção é mudada para um sentido aleatório.

Quando o utilizador finalmente perder ,existem 2 possibilidades:

- O utilizador alcançou uma nova pontuação maxima.

- O utilizador não alcançou uma nova pontuação máxima.

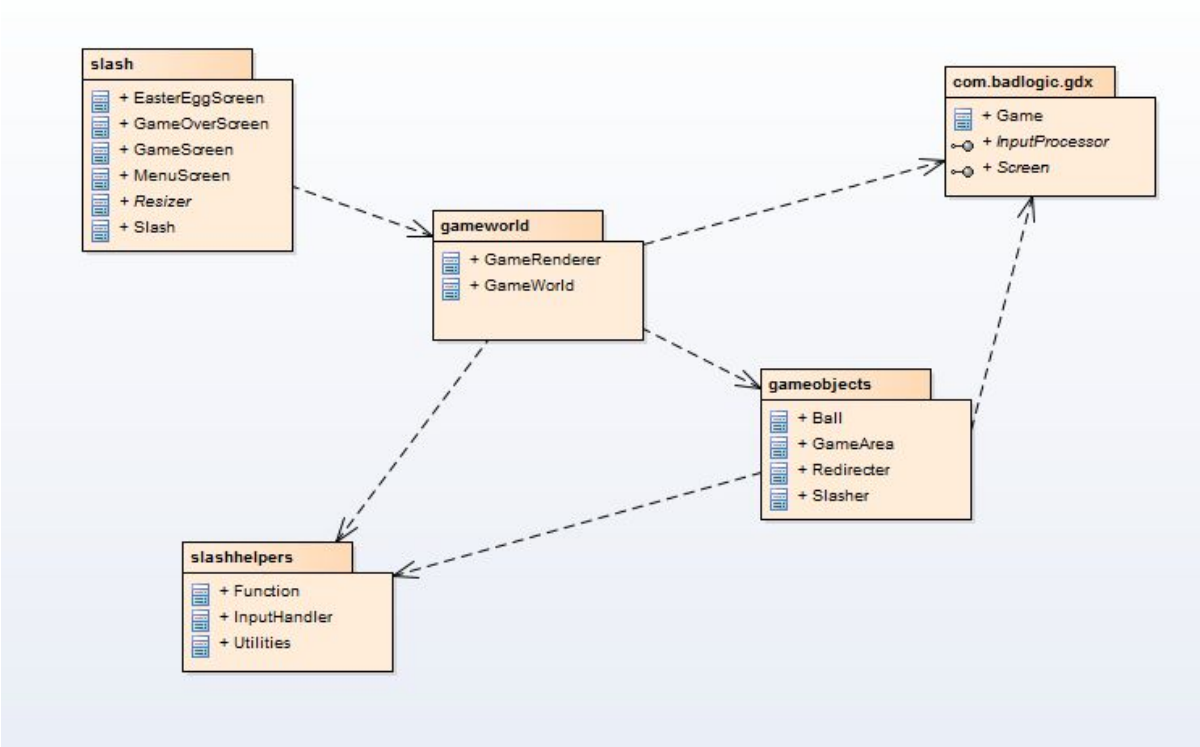


2.4 Ficheiros

Após a primeira vez que o utilizador joga, é criado o ficheiro “highscore.txt” (o utilizador não lhe consegue aceder), onde é guardada a melhor pontuação do utilizador. Este ficheiro vai sendo atualizado conforme a melhor pontuação vai aumentando.

3. Conceção, Implementação e Teste

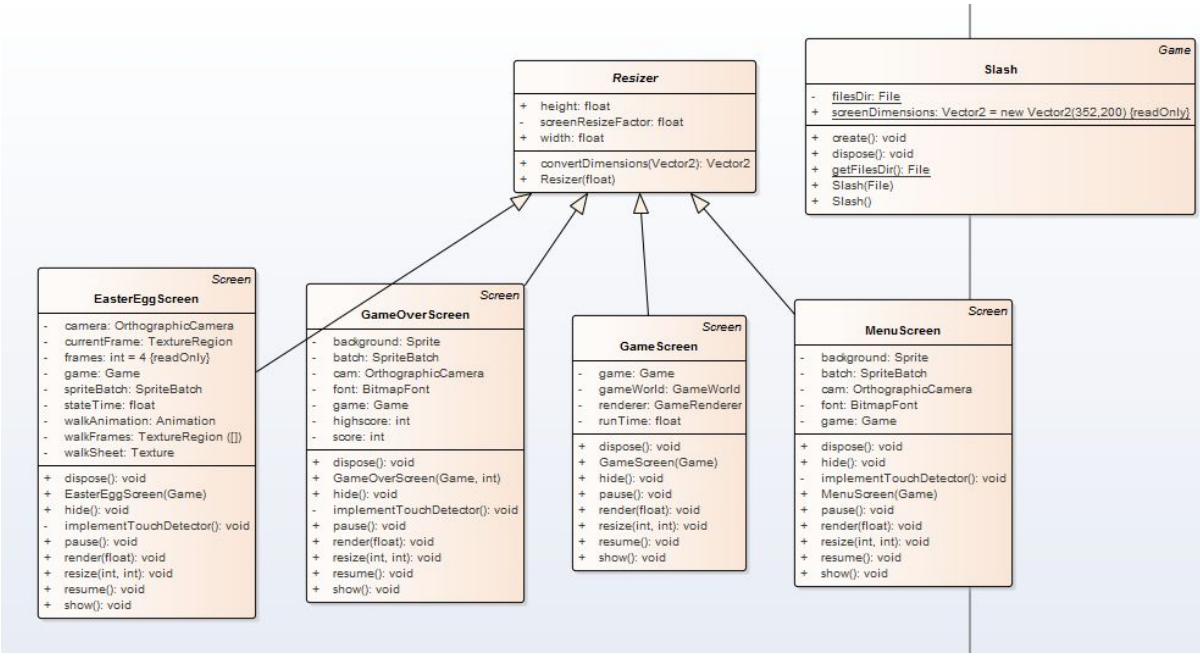
3.1 Estrutura de packages



Package	Descrição
com.lpoo.slash	Contém os diversos ecrãs.
com.lpoo.gameworld	Contém os objetos usados no jogo.
com.lpoo.gameobjects	Gere o jogo no seu total, apesar de a parte gráfica e lógica estarem separadas.
com.lpoo.slashhelpers	Contém diversas “ajudas” para diferentes situações.

3.2 Estrutura de classes

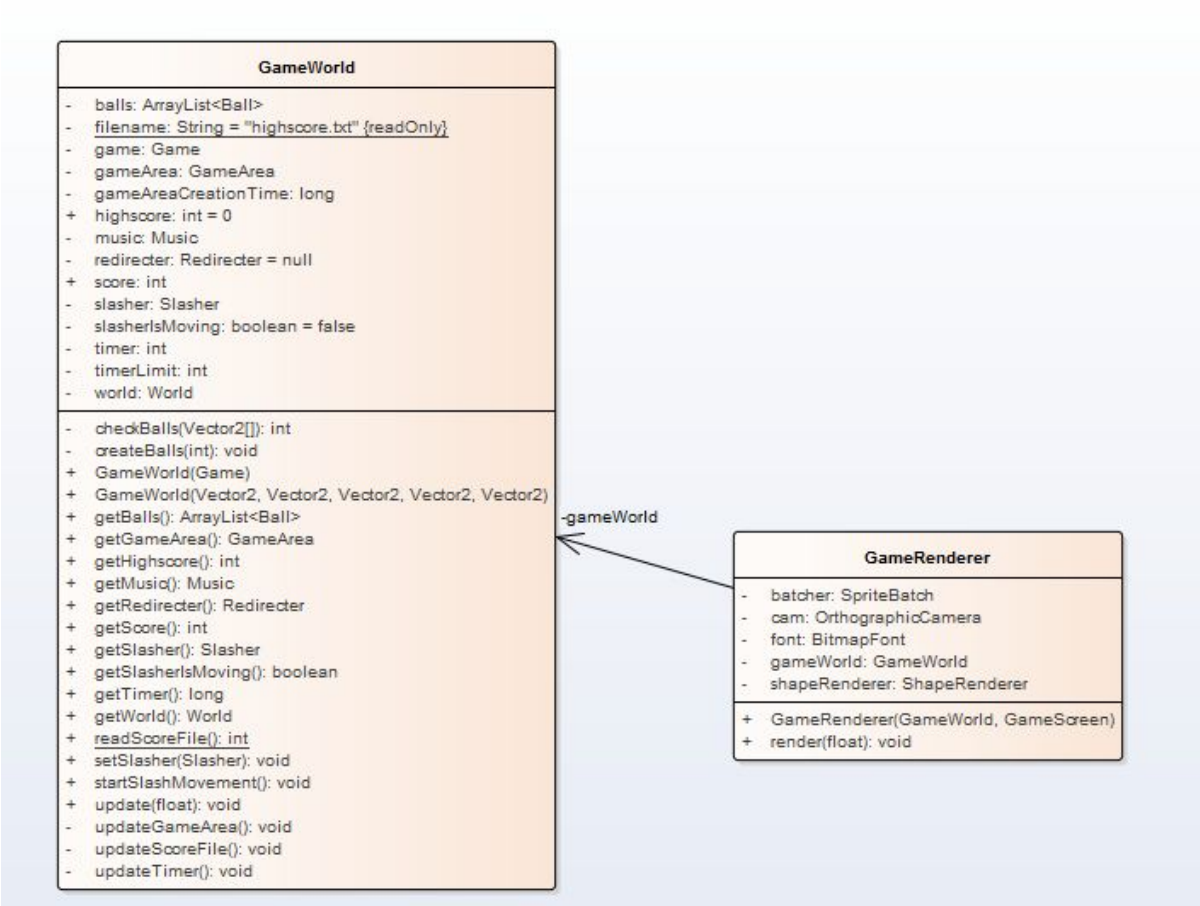
Package **com.lpoo.slash**



Class	Descrição

Slash	Chama o primeiro ecrã que será usado, o MenuScreen
Resizer	Contém alguma informação e ajudas acerca de dimensões do ecrã
MenuScreen	Ecrã onde a app começa
GameScreen	Ecrã onde o jogo acontece
GameOverScreen	Ecrã que aparece quando o jogo acaba
EasterEggScreen	Ecrã com uma animação

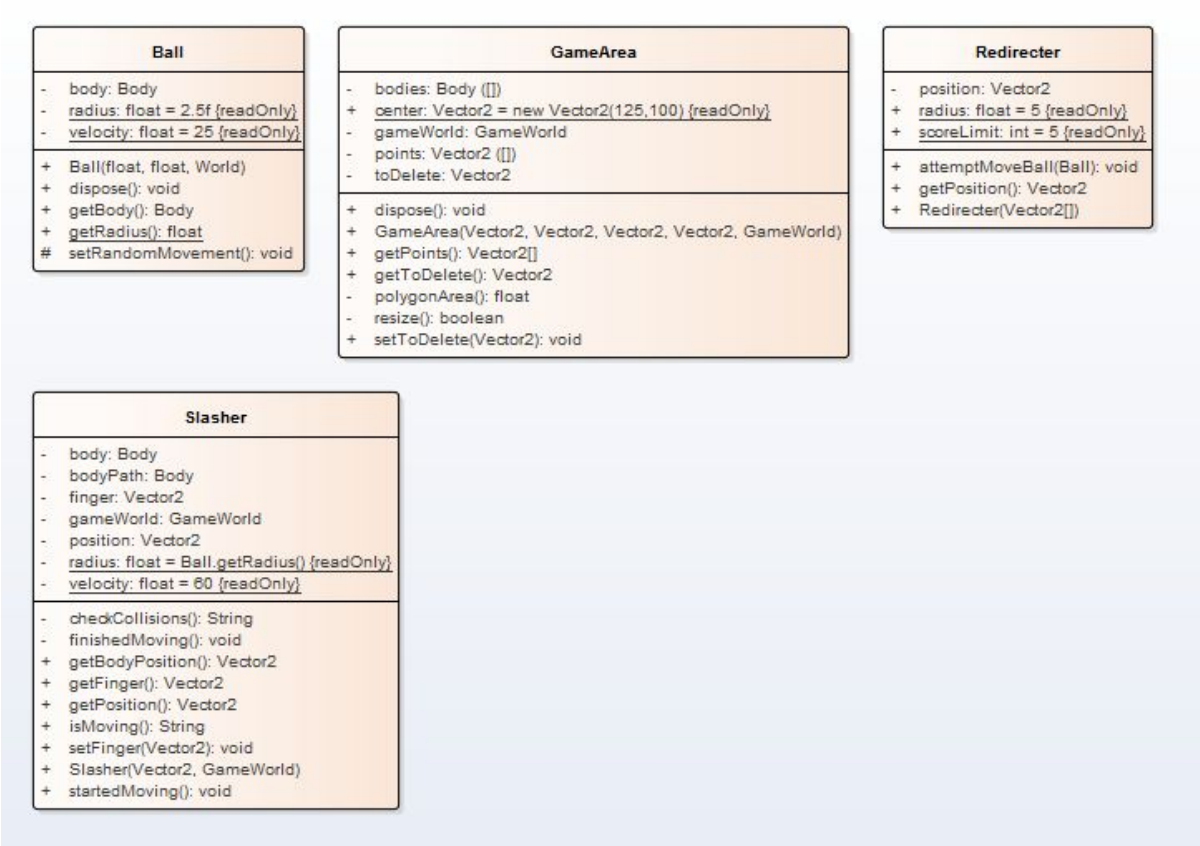
Package **com.lpoo.gameworld**



Class	Descrição

GameWorld	Onde todos os gameobjects são guardados e geridos
GameRenderer	Onde o rendering acontece

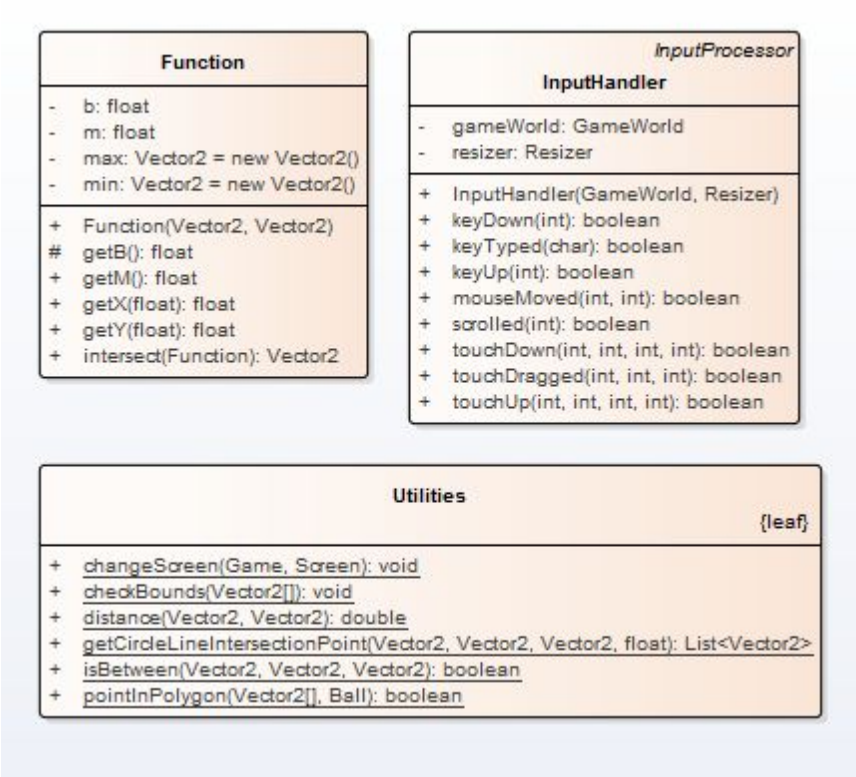
Package **com.lpoo.gameobjects**



Class	Descrição

GameArea	Zona por onde as bolas andam
Ball	Bola que anda na GameArea
Slasher	Responsável por cortar a GameArea
Redirecter	Redireciona bolas

Package **com.lpoo.slashhelpers**



Class	Descrição

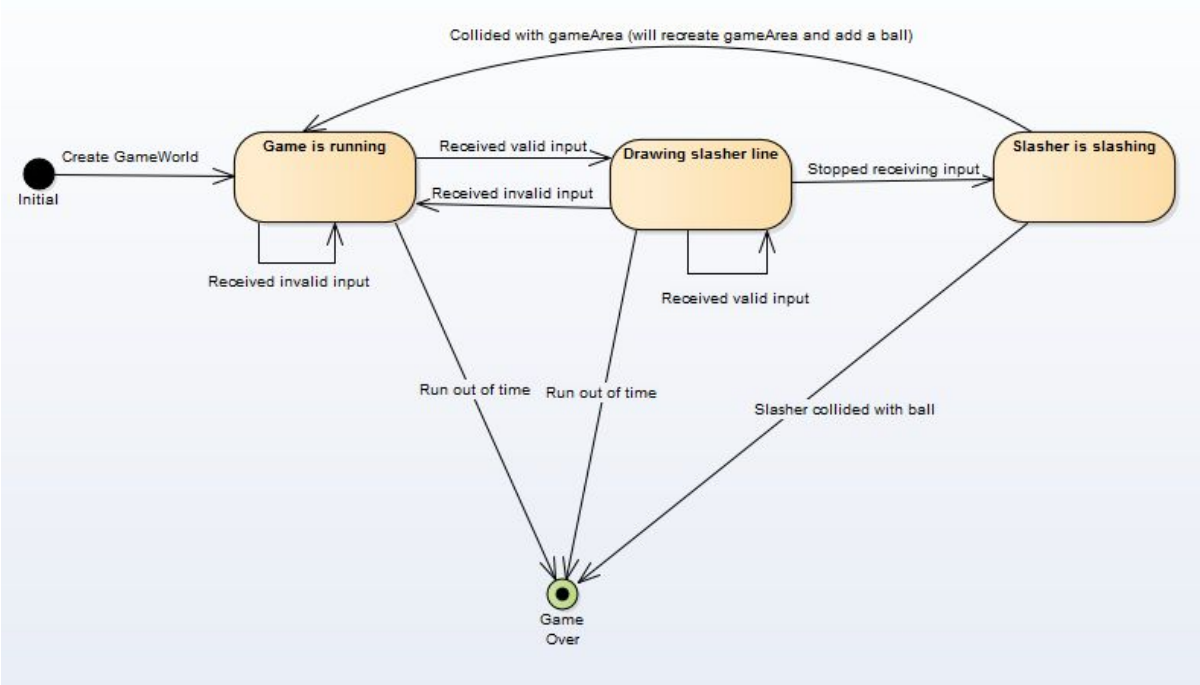
InputHandler	Recebe inputs que depois aplica no GameWorld
Function	Permite usar funções y=m*x+b
Utilities	Contém diversos métodos com variados fins

3.3 Padrões de desenho utilizados

- **MVC** → Usado para separar a parte gráfica (GameRenderer), lógica (GameWorld), e os inputs (InputHandler), de maneira a que sejam mais percetiveis as responsabilidades de certas partes do código.
- **Façade** → Usado para simplificar alguns mecanismos presentes no jogo (package slashhelpers), como por exemplo a mudança de ecrã.

3.4 Mecanismos e comportamentos importantes

Este diagrama de estados representa os mecanismos presentes no ciclo de jogo, incluindo as situações em que o jogo pode acabar.



3.5 Ferramentas, bibliotecas e tecnologias utilizadas

Como IDE, usamos o **Android Studio** tendo em conta que iriamos desenvolver uma app para Android, e portanto achamos o mais adequado logo desde inicio.

Utilizou-se a biblioteca **libGDX** de modo a facilitar o uso da interface gráfica e da aplicação de física, tendo em conta que esta última teve ajuda da extensão **box2d**.

3.6 Dificuldades encontradas e sua resolução

As dificuldades no uso das ferramentas externas foram algumas, sendo que as principais foram: o uso do **libGDX** no inicio do projeto, resolvida com a realização de alguns tutoriais; e o uso de **Socket Programming**, dificuldade que não conseguimos superar na totalidade, tendo em conta que apenas conseguimos uma versão desktop vs desktop.

No entanto, as dificuldades também se manifestaram no desenvolvimento do projeto, sendo um bom exemplo o facto de que não pudemos usar um contactListener para detetar as colisões do Slasher visto que as colisões do mesmo com a GameArea não era detetada em virtude de os BodyType desses 2 objetos não permitirem colisões um com o outro, e desse modo tivemos de detetar colisões manualmente.

Houve também dificuldades em termos de escolhas de como estruturar o projeto. Um exemplo foi a escolha de não usarmos *singletons*, visto que iam contra os princípios do encapsulamento, apesar de termos tido várias oportunidades de implementação.

3.7 Lista de testes realizados

Para que os testes não ocupem demasiado espaço neste relatório, optamos por deixá-los na mesma pasta que o código, */core/src/test/java*. Também podem ser consultados no repositório Git aqui.

Os testes abrangem todas as classes do package gamebbjects e ainda a class Function do package slashhelpers, de maneira a garantir as funcionalidades presentes no Slash.

4. Conclusões

Este relatório procura mostrar os aspetos mais relevantes acerca deste projeto, desde o seu funcionamento à estrutura escolhida em prol desse mesmo funcionamento.

No geral conseguimos concretizar tudo o que pretendiamos, excetuando o facto de não conseguirmos fazer multiplayer android vs android.

As melhorias que poderiamos fazer seriam superar a dificuldade anteriormente mencionada e adicionar mais objetos ao GameWorld, aumentando a complexidade do jogo.

Na globalidade, o trabalho foi maioritariamente feito por Diogo Duque (70%), tendo o restante sido feito por João Gomes (30%).

5. Referências

Moodle de LPOO: https://moodle.up.pt/course/view.php?id=1132

StackOverflow: http://stackoverflow.com/