

Faculdade de Engenharia da Universidade do Porto



Laboratório de Programação Orientado a Objetos

2015/2016 MIEIC

Apocalypse



Turma: 5

Grupo:

Autores:

Catarina A. T. Ramos (up201406219@fe.up.pt)

Pedro M. D. Soares (up201404178@fe.up.pt)

Índice

1. Introdução
2. Manual de Utilização
 - 2.1. Descrição do jogo
 - 2.2. Funcionalidades
 - 2.3. Instalação e Arranque
 - 2.4. Modo de Utilização
 - 2.4.1. Menu
 - 2.4.2. Jogar (Play)
 - 2.4.3. Armas
 - 2.4.4. Zumbis
 - 2.4.5. Caixas de Munições (Ammo Boxes)
 - 2.4.6. Fim de Jogo (Game Over)
 - 2.4.7. Pontuações (HighScores)
 - 2.4.8. Instruções (HowToPlay)
 - 2.5. Ficheiros de entrada/saída
3. Conceção, Implementação e Testes
 - 3.1. Estrutura de Packages
 - 3.2. Estrutura de Classes
 - 3.2.1. Package logic
 - 3.2.2. Package sprites
 - 3.2.3. Package gui
 - 3.2.4. Package states
 - 3.2.5. Package audio
 - 3.2.6. Package tests
 - 3.3. Padrões de Desenho
 - 3.3.1. States
 - 3.3.2. FlyWeight
 - 3.3.3. Factory
 - 3.3.4. Singleton
 - 3.4. Mecanismos e Aspetos importantes

- 3.4.1. Gestão de Jogo
 - 3.4.2. Máquinas de Estados
- 3.5. Ferramentas, Bibliotecas e Tecnologias utilizadas
- 3.6. Dificuldades
- 3.7. Testes realizados
- 4. Conclusões
- 5. Referências

1. Introdução

Este projeto foi proposto no âmbito da cadeira de Laboratório de Programação Orientada a Objetos (LPOO) do curso de Mestrado Integrado de Engenharia Informática e Computação Gráfica (MIEIC) da Universidade do Porto e foi levado a cabo no segundo semestre do ano letivo 2015/2016.

O projeto teve como objetivo principal o desenvolvimento de um jogo para a plataforma *android* dentro de um tema previamente aprovado pelo professor, onde foram aplicados conhecimentos adquiridos da cadeira encarregue. O nome do nosso projeto é “Apocalypse” e este, tem como objetivo, mantermo-nos em jogo o mais tempo possível. Com uma perspetiva *top-down*, estamos limitados a um espaço que é invadido por zumbis a cada nível, e temos em nossa posse armas para os matarmos, entre outras funcionalidades de jogo.

Este relatório tem como objetivo explicar e apresentar a estrutura e organização do nosso jogo, em vários aspetos, e outras funcionalidades e componentes que são importantes para a compreensão do funcionamento e desenvolvimento do mesmo. Para o cumprimento do mesmo, serão apresentamos diagramas em UML e outros métodos para uma melhor compreensão.

2. Manual de Utilização

2.1. Descrição do jogo

Apocalypse é um jogo de sobrevivência com uma perspectiva *top-down* delimitada por uma área quadrada em que o funcionamento do jogo é gerido, principalmente, pelo nível em que o utilizador se encontra. Numa primeira fase, o jogador é colocado no centro da área de jogo e é gerado, a cada nível, uma *bot wave* de zumbis e *ammo boxes*, a funcionalidade dos mesmos será explicada com mais pormenor nos próximos pontos abaixo.

2.2. Funcionalidades

Para o bom funcionamento do nosso jogo, foi necessário a implementação de algumas funcionalidades obrigatórias para o seu funcionamento, tendo, também nós, implementado alguns extras. Foram implementados vários estados para permitir diferentes cenas de interação com o utilizador que serão enumerados de seguida:

- Menu principal
- Jogo
- Resultados
- Como Jogar
- Final de Jogo

O menu principal possui música ambiente e é constituído por 4 botões, possuem som de clique, que permitem o acesso as diferentes cenas.

O primeiro botão dá acesso ao Jogo onde foi implementado uma HUD (*Heads-up display*) como método de contacto entre utilizador e o jogo. Esta contém várias funcionalidades para o controlo sobre o jogo como por exemplo:

- *Joystick* – controlar movimento do jogador;
- Botão *shoot* – lançamento de balas na direção do jogador;
- Botão A - troca de arma;
- Botão B - iniciar a animação de recarga do jogador;
- Botão de som - retirar/iniciar o som;
- Botão de pausa - pausar/retomar jogo;
- Botão de voltar atrás - desistir do jogo e voltar ao menu principal;
- *Label* nível - indica o nível de dificuldade do jogo
- *Label* score – indica a pontuação atual do utilizador
- *Label* ammo – quantidade de munições disponível na arma seleccionada;

No decorrer do jogo é possível ouvir e ver os diferentes sons e animações como por exemplo:

- Som de disparo das armas
- Som de música ambiente
- Animação dos zumbis, jogador e de colisão entre zumbis e as balas
- Mudança de estado dos botões quando pressionados

No final do jogo é mostrado o resultado obtido pelo utilizador e se estiver nos 5 melhores resultados aparecerá uma mensagem a indicar o mesmo.

Este estado é sensível a qualquer toque, qualquer toque provoca a transição para o menu de resultados onde é possível visualizar os 5 melhores resultados e quando foram obtidos, é sensível ao toque.

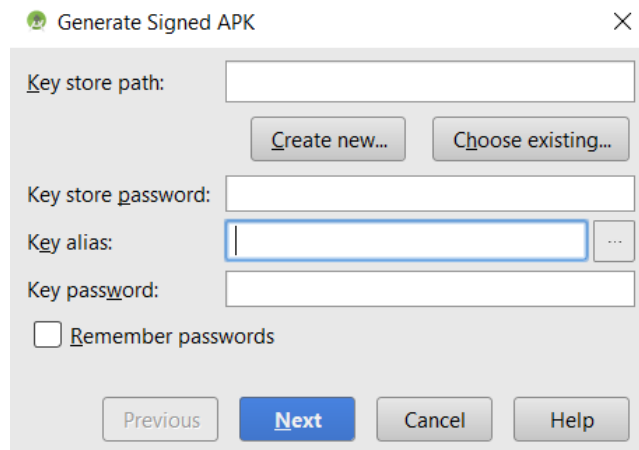
O terceiro botão dá acesso ao “Como jogar” e permite ao utilizador entender melhor o funcionamento do jogo. Esta cena é constituída por uma sequência de imagens informativas, para passar para a imagem seguinte basta clicar no ecrã do telemóvel.

2.3. Instalação e Arranque

O nosso jogo pode ser exportado para uma aplicação com extensão *‘.apk’* e pode ser instalado em qualquer dispositivo *android* com uma *framework* acima ou igual à 4.3. Após a instalação, é só carregar no *icon* das aplicações que diz “*Apocalypse*” e iniciar o jogo.

Instalação num dispositivo *android* através do *Android Studio*:

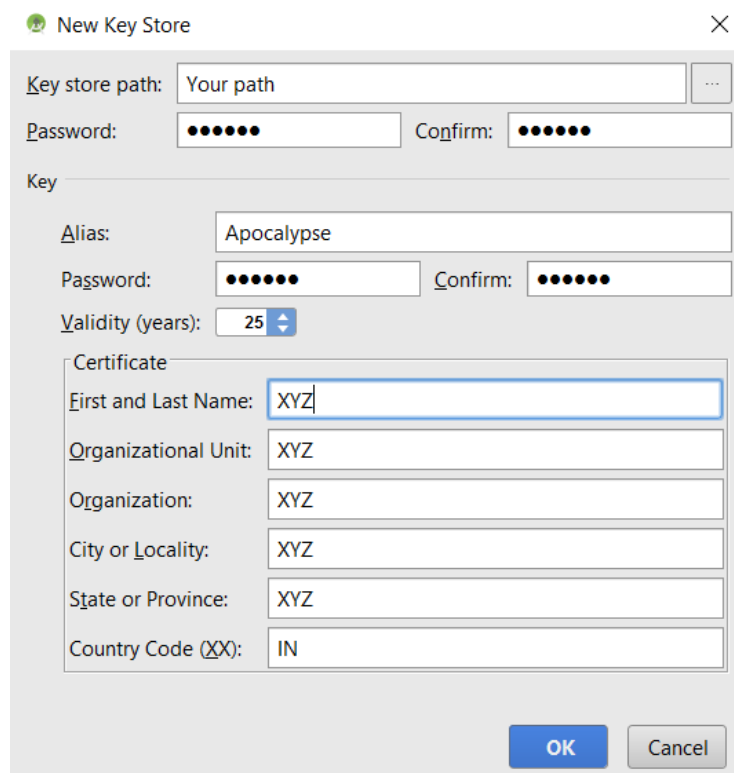
1. Build -> Generate Signed APK...



The screenshot shows the 'Generate Signed APK' dialog box. It has a title bar with a green Android icon and the text 'Generate Signed APK'. The dialog contains the following fields and buttons:

- Key store path:** A text input field with a browse button (three dots) to its right.
- Buttons:** 'Create new...' and 'Choose existing...' buttons are located below the 'Key store path' field.
- Key store password:** A text input field.
- Key alias:** A text input field with a browse button (three dots) to its right.
- Key password:** A text input field.
- Remember passwords:** A checkbox.
- Navigation buttons:** 'Previous', 'Next' (highlighted in blue), 'Cancel', and 'Help' buttons are at the bottom.

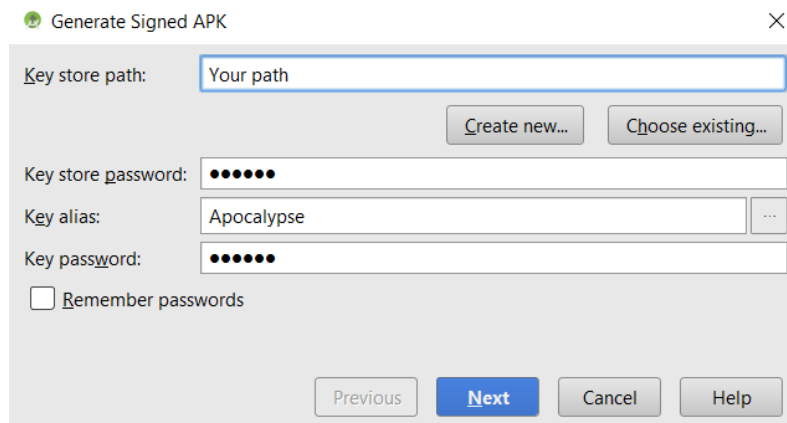
2. Create new... e preencher como na figura abaixo para uma criação rápida, inserindo o *path* pretendido e as *passwords*. Carregar em OK.



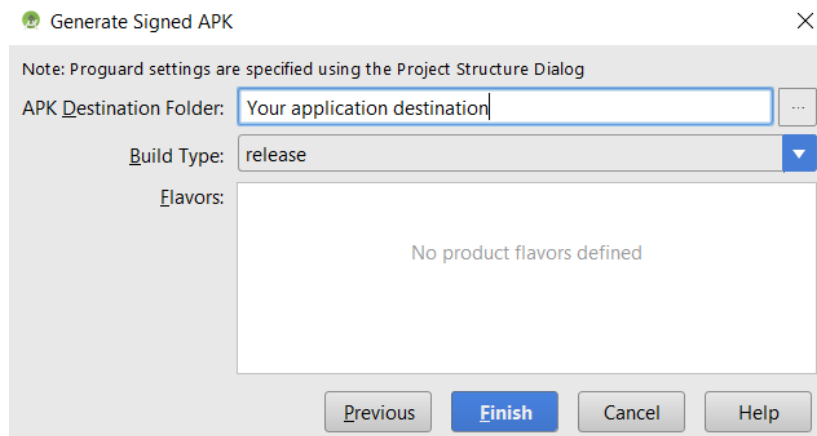
The screenshot shows the 'New Key Store' dialog box. It has a title bar with a green Android icon and the text 'New Key Store'. The dialog contains the following fields and buttons:

- Key store path:** A text input field with the value 'Your path' and a browse button (three dots) to its right.
- Password:** A text input field with masked characters (dots).
- Confirm:** A text input field with masked characters (dots).
- Key section:**
 - Alias:** A text input field with the value 'Apocalypse'.
 - Password:** A text input field with masked characters (dots).
 - Confirm:** A text input field with masked characters (dots).
 - Validity (years):** A spinner box showing the value '25'.
- Certificate section:**
 - First and Last Name:** A text input field with the value 'XYZ'.
 - Organizational Unit:** A text input field with the value 'XYZ'.
 - Organization:** A text input field with the value 'XYZ'.
 - City or Locality:** A text input field with the value 'XYZ'.
 - State or Province:** A text input field with the value 'XYZ'.
 - Country Code (XX):** A text input field with the value 'IN'.
- Buttons:** 'OK' (highlighted in blue) and 'Cancel' buttons are at the bottom right.

3. Selecionar a KEY anteriormente criada e carregar em NEXT.



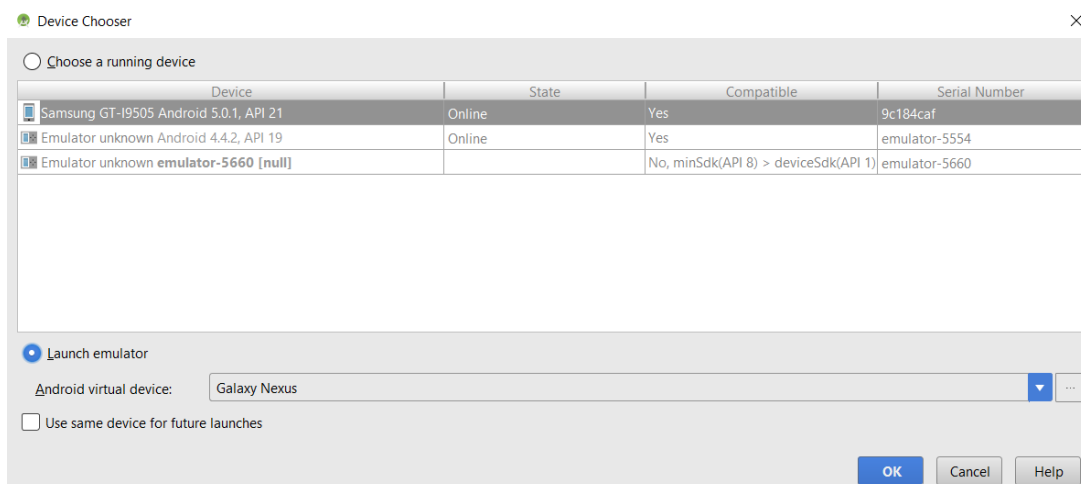
4. Selecionar o destino onde será criado o ficheiro apk e carregar em FINISH.



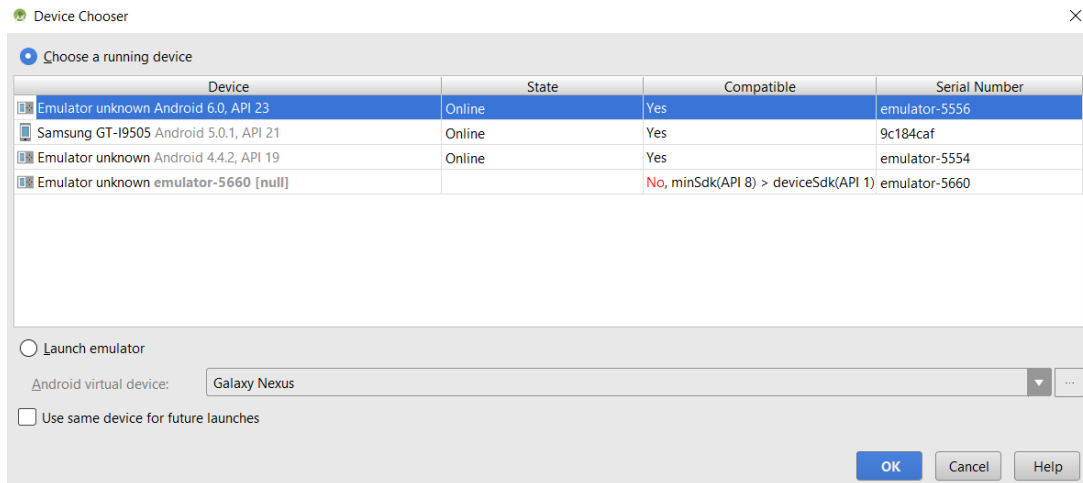
5. Passar o ficheiro para o dispositivo *android* e efetuar a instalação.

Para o uso do nosso jogo através de um emulador, aconselhamos o Galaxy Nexus utilizando a API 23, que foi o utilizado para o desenvolvimento do nosso jogo.

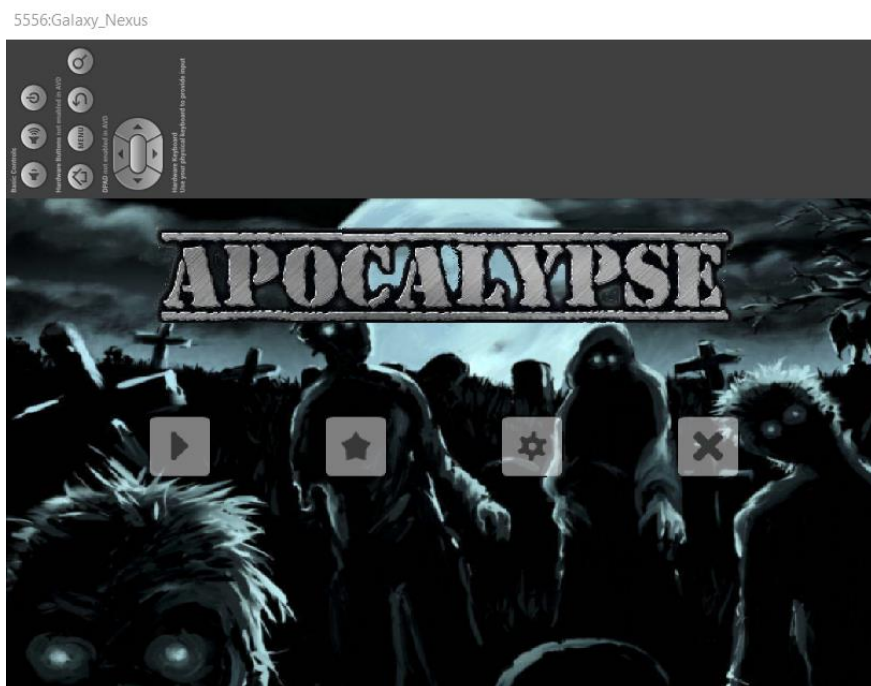
1. Run -> Run Android.
2. Inicializar o emulador, de preferência o Galaxy Nexus.



3. Após a inicialização do emulador, Run -> Run Android e selecionar a API 23. Carregar em Ok.



4. O *Android Studio* é encarregue de fazer a instalação no emulador e inicializa-la no emulador.



2.4. Modo de Utilização

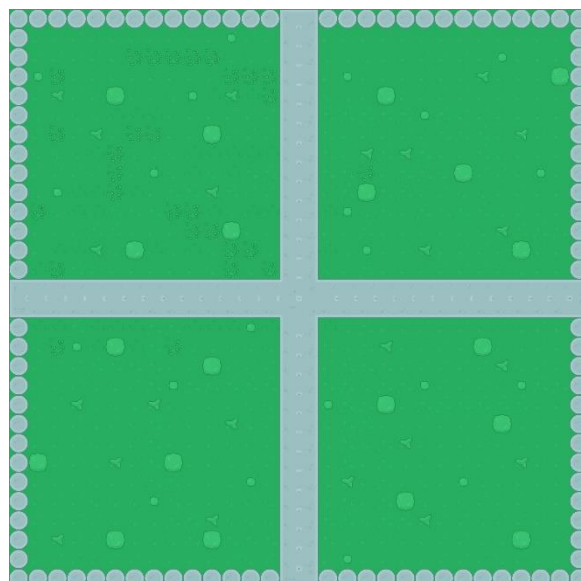
2.4.1. Menu

Ao inicializar o nosso jogo somos direcionados para o meu inicial como está ilustrado na imagem abaixo. Isto é visto sobre uma máquina de estados e o seu funcionamento está explícito no ponto 3.3.1 em *States*.

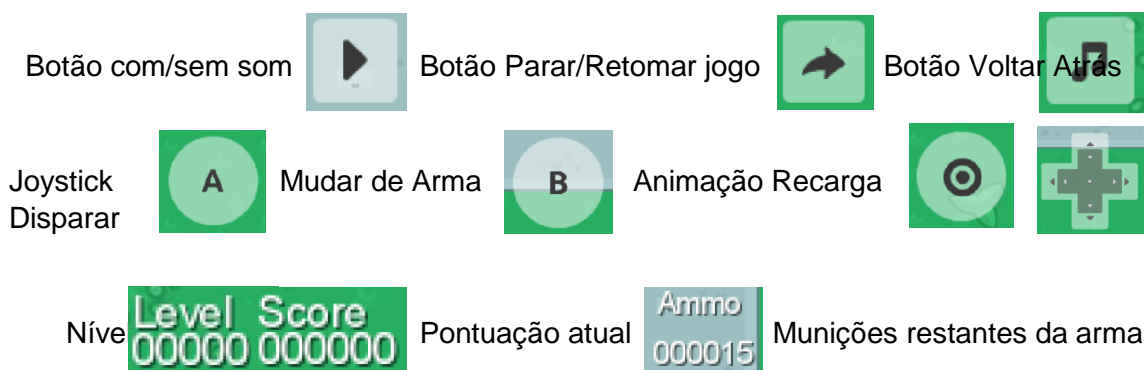
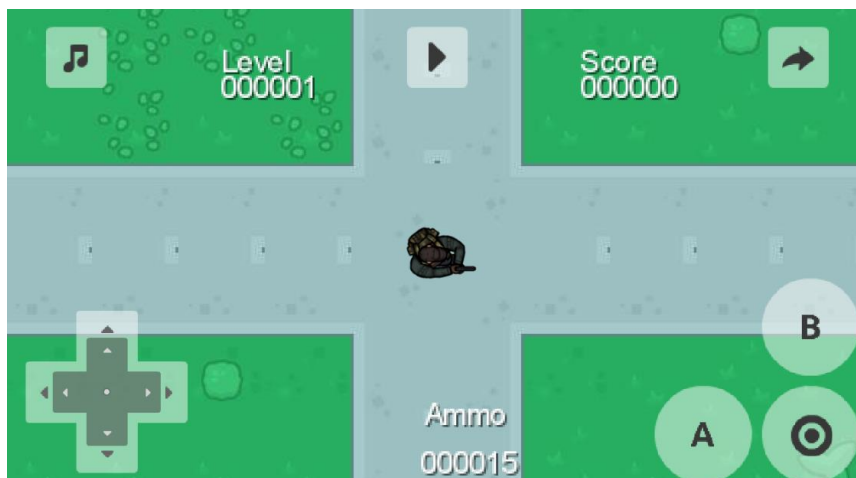


2.4.2. Jogar (Play)

Ao carregar no botão *Play* o nosso jogo começa e somos posicionados no centro de um mapa quadrado como o da imagem abaixo.



São nos disponibilizadas várias funcionalidades para jogar que fazem parte da HUD, referida no ponto 2.3, como as abaixo representadas.



Notas:

- O *joystick* movimenta o jogador conforme o vetor resultante desde o centro do joystick.
- As munições disparadas dependem do tipo de arma selecionada e são disparadas na mesma direção do jogador.
- O botão de voltar atrás não guarda o estado do jogo nem a pontuação.

2.4.3. Armas

Gun – Mais fraca mas com mais munições



Rifle – Capaz de matar mais do que um zumbi com uma bala mas com poucas munições

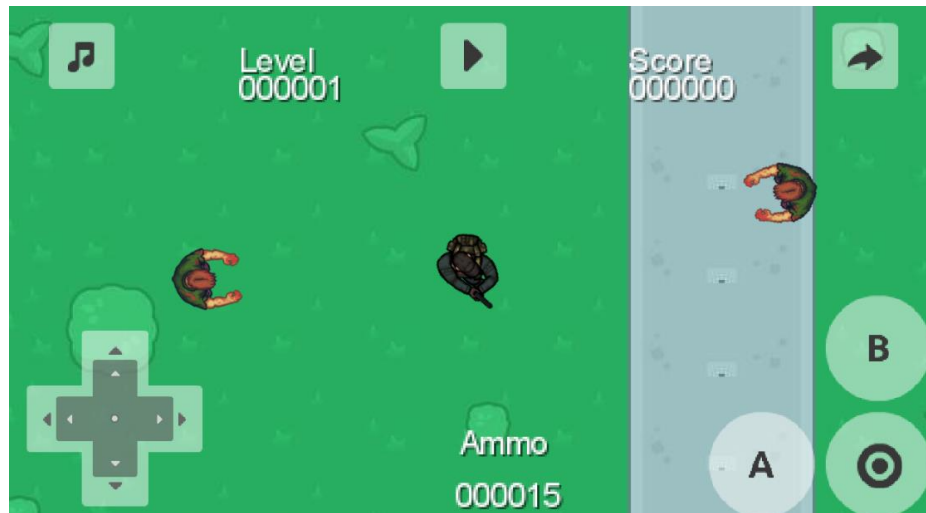


2.4.4. Zumbis

Existem zumbis que entram pelos limites do mapa e que andam em posições aleatórias, caso entrem na zona visível do jogador, virão na sua direção e irão segui-lo até o matar. A única maneira de nos livrarmos deles é disparar uma bala contra eles.

Quando todos os zumbis em campo são mortos subimos de nível e uma nova onda de zumbis é lançada em jogo.

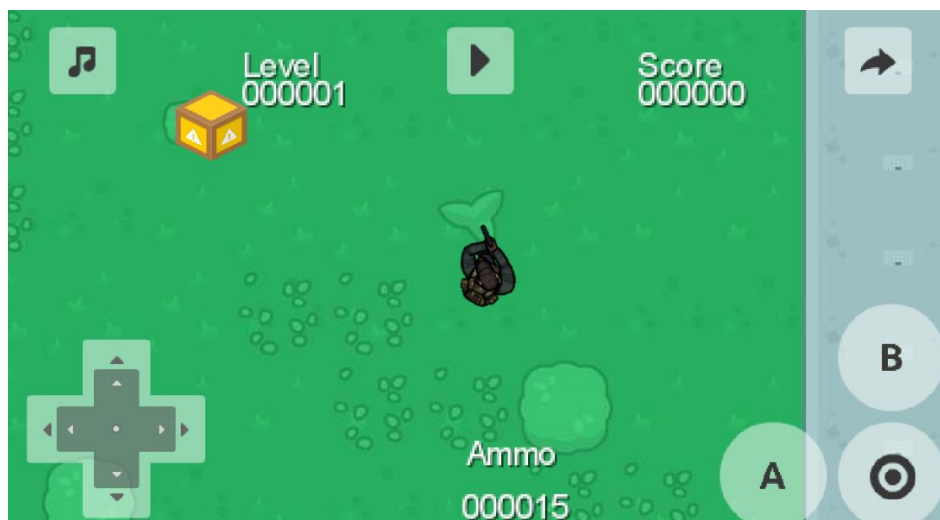
Se chegarem ao alcance do jogador o jogo acaba.



2.4.5. Caixas de munições (Ammo Boxes)

Espalhados pelo mapa também se encontram pequenas caixas amarelas que contêm munições. Ao colidir com elas, é incrementado o número de munições do jogador nas duas armas.

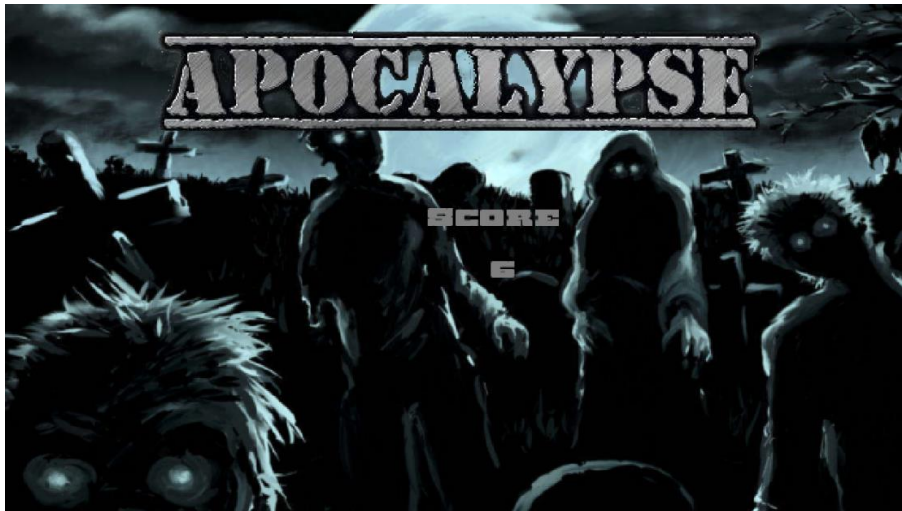
Estas caixas são repostas a cada nível. Caso o jogador não as apanhe e passe de nível, estas deixam de estar disponíveis.



2.4.6. Fim de Jogo (Game Over)

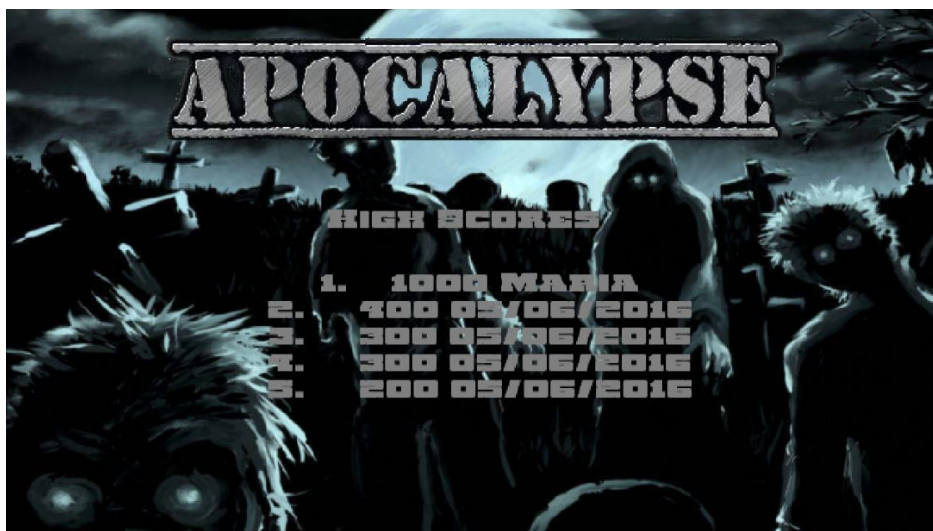
Visto que se trata de um jogo de sobrevivência, o fim do jogo dá-se quando há a colisão de um zumbi com o jogador. Neste seguimento, a pontuação do jogador é apresentada e guardada e depois são exibidos os melhores *HighScores*. Caso haja um novo recorde, somos avisados de tal.

Ao carregar no botão de voltar atrás não estamos a acabar o jogo, portanto, o jogo é perdido e as pontuações não são guardadas.



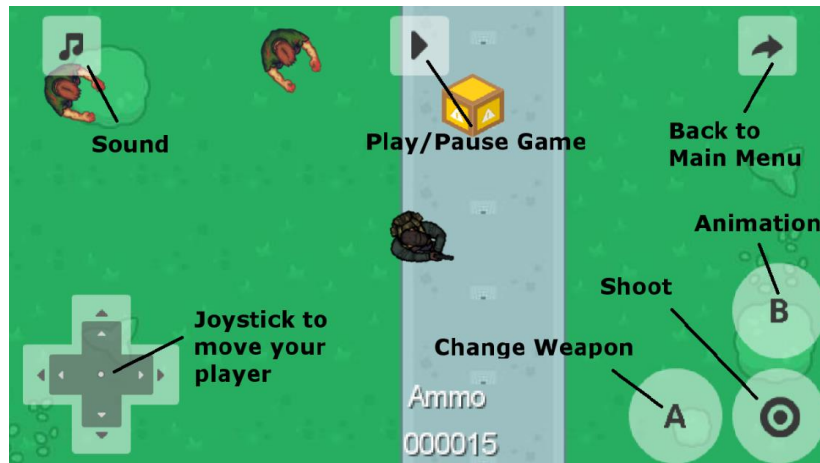
2.4.7. Pontuações (HighScores)

As melhores cinco pontuações são guardadas e podem ser visualizadas acedendo ao às pontuações pelo botão de *HighScores* no menu principal. Para depois voltas atrás basta clicar uma vez no ecrã. As pontuações serão guardadas por data de finalização do jogo.



2.4.8. Instruções (How to Play)

As instruções ou modo de utilização do jogo também pode ser acedido pela aplicação através do botão How To Play no menu inicial. Para passar á próxima página do tutorial basta clicar no ecrã.



2.5. Ficheiros de entrada/saída

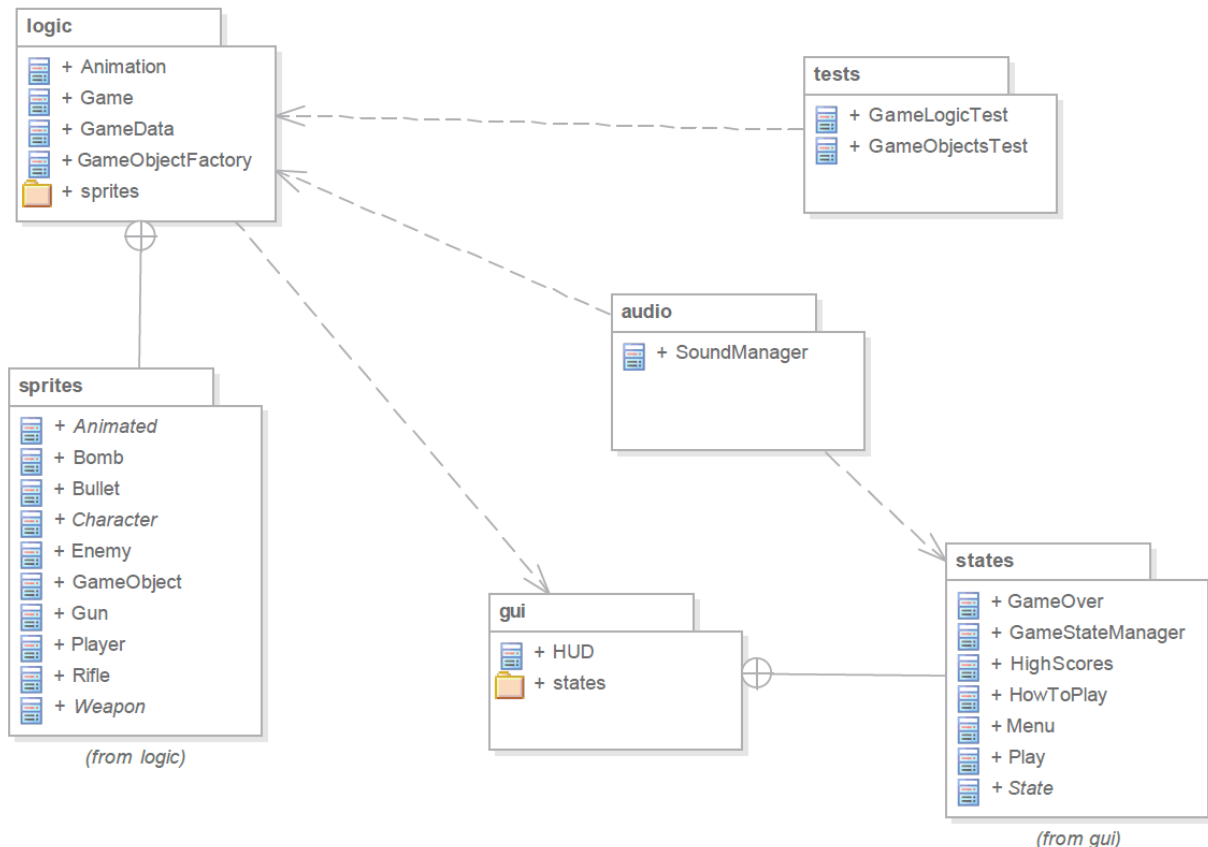
O ficheiro de entrada/saída presente no projeto denomina-se “highscores.txt”. Este tem como função guardar o registo dos cinco melhores resultados e a data a que estes foram obtidos.

O conteúdo do ficheiro é atualizado quando um novo resultado é superior a um ou mais dos resultados guardados no ficheiro. Além disso, o conteúdo é carregado para a classe “*GameData*” quando o jogo se encontra no menu “*Highscores*”.

3. Conceção, Implementação e Testes

3.1. Estruturas de Packages

Para uma boa organização do nosso projeto, procedemos à subdivisão por packages dependendo da funcionalidade do código. Essa organização e dependências podem ser vistas sob a forma de um diagrama como o abaixo apresentado.



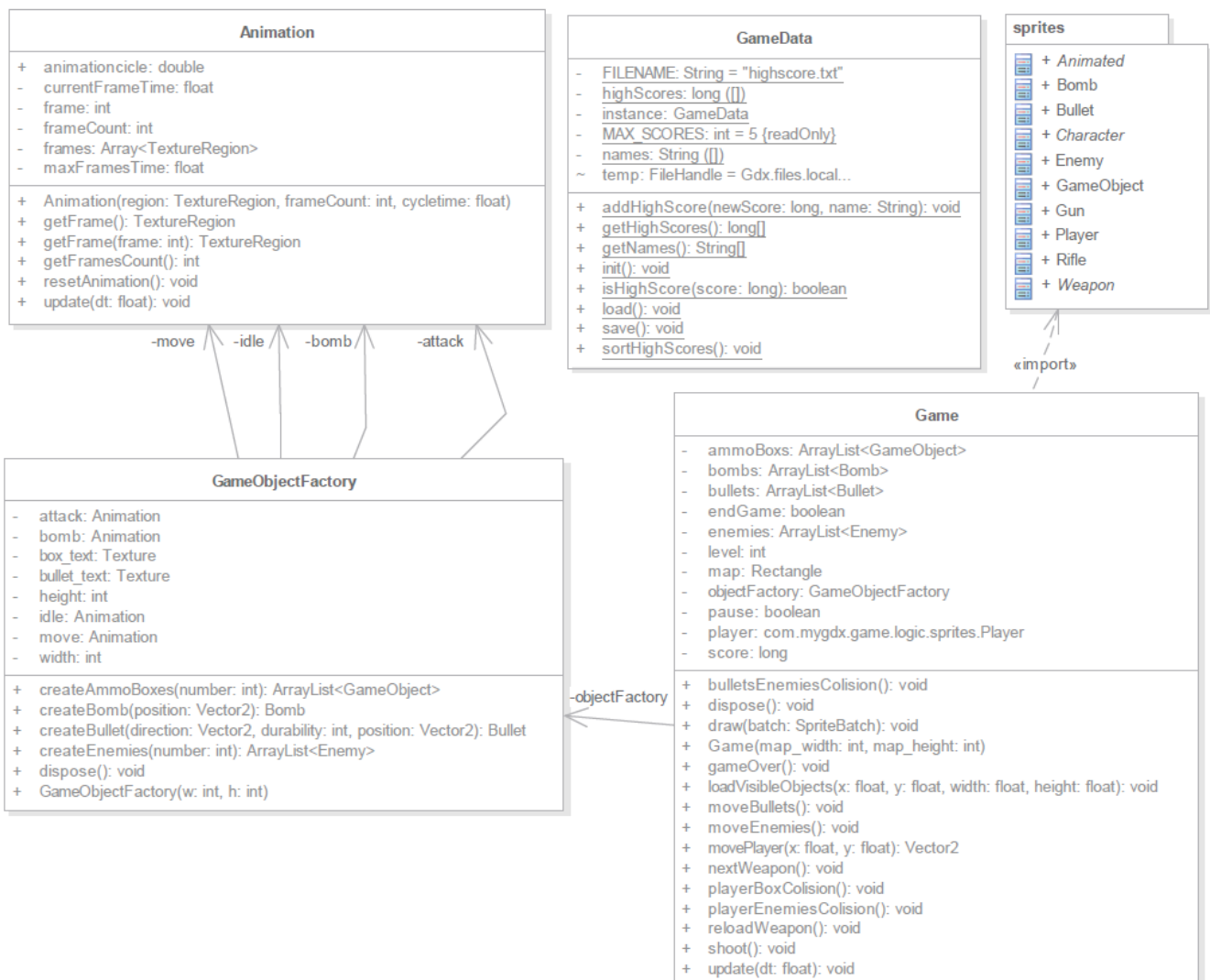
Package	Explicação/Conteúdo
logic	Lógica do Jogo
sprites	Elementos/Objetos presentes em jogo
gui	Interface Gráfica
states	Conjunto de estados e gestão dos mesmos para a navegação da aplicação
audio	Gestão de áudio
tests	Testes da lógica do jogo

3.2. Estrutura de Classes

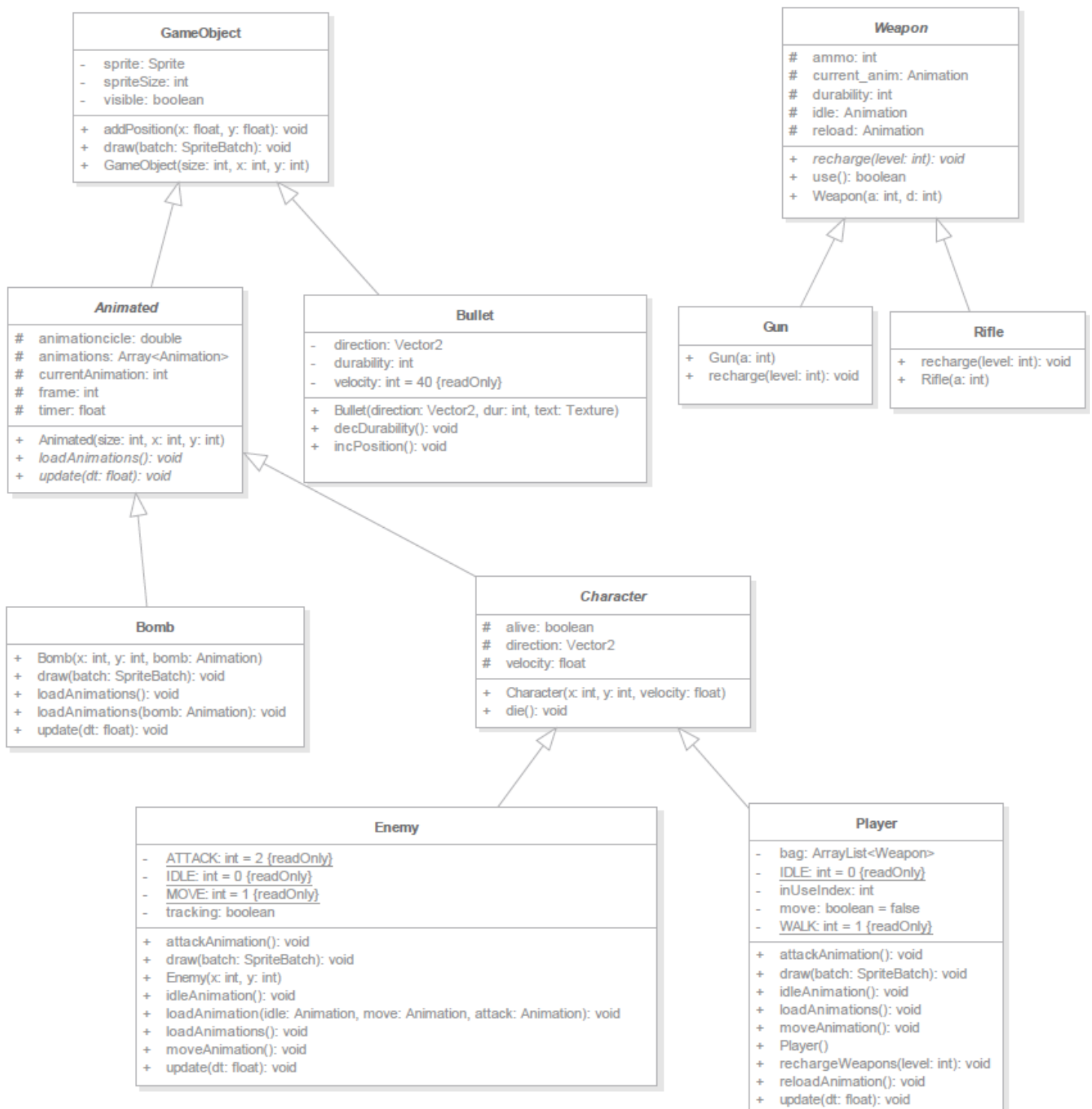
Neste ponto iremos apresentar os diagramas de classes UML por package, incluindo ligações a classes externas de outros externos. Os métodos *sets* e *gets* das classes serão omitidos.

Package	Função
logic	Responsável pela lógica do jogo
sprites	Conjunto de todas as sprites/objetos em jogo
gui	Responsável pela interface gráfica
states	Mecanismo de estados
audio	Responsável pelo Áudio
tests	Testes de qualidade do programa

3.2.1. Package logic



3.2.2. Package Sprites (de logic)

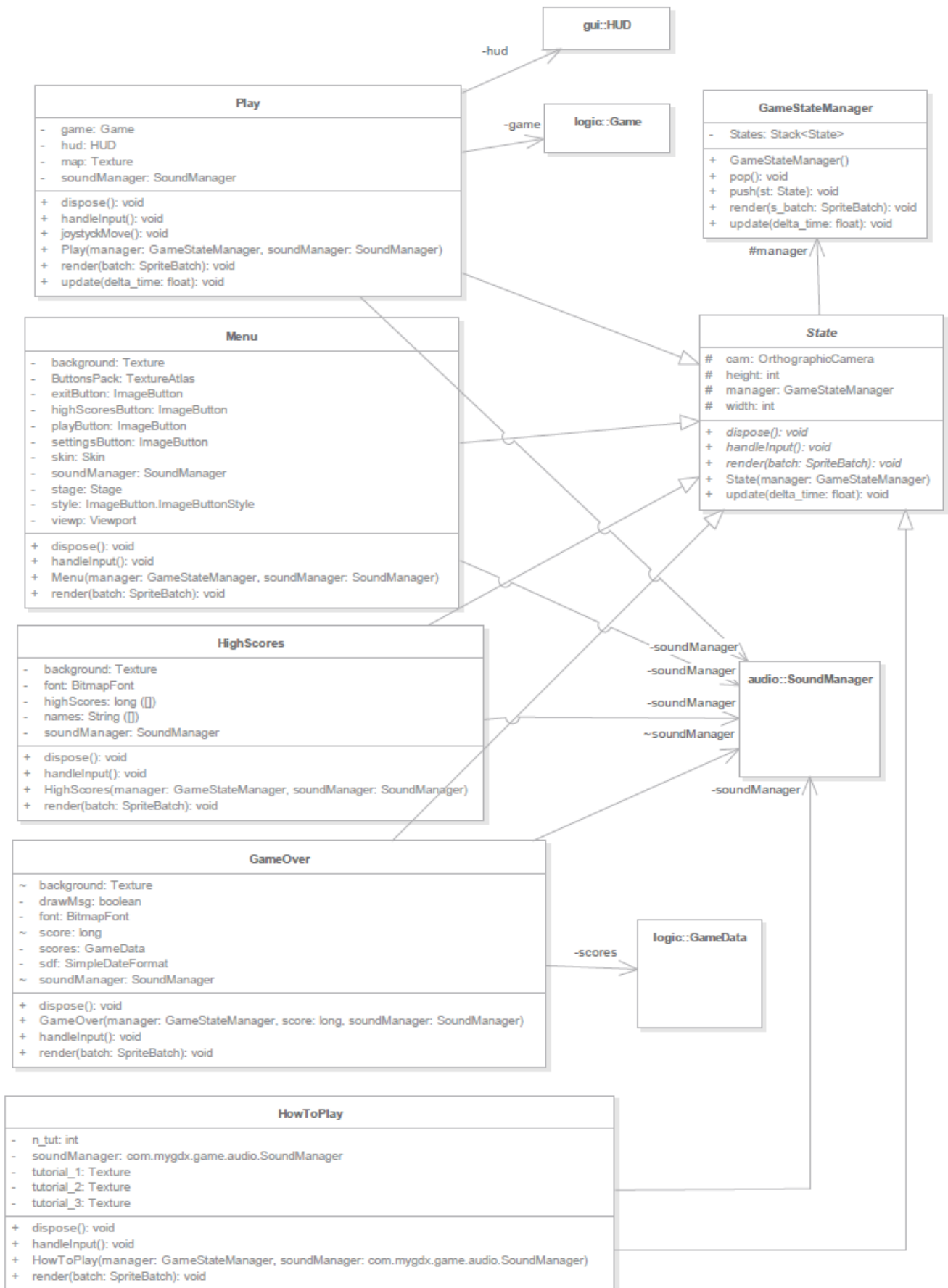


3.2.3. Package gui

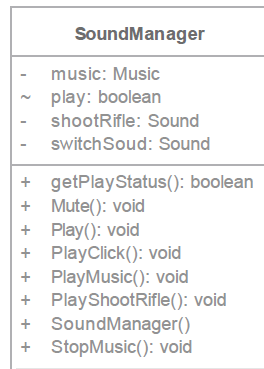
HUD
<ul style="list-style-type: none">- aButton: ImageButton- ammo: int- ammoLabel: Label- ammoMsg: Label- backButton: ImageButton- bButton: ImageButton- ButtonsPack: TextureAtlas- fontScale: int = 3- height: float- level: int- levelLabel: Label- levelMsg: Label- mute: ImageButton.ImageButtonStyle- pause: ImageButton.ImageButtonStyle- play: ImageButton.ImageButtonStyle- playButton: ImageButton- score: long- scoreLabel: Label- scoreMsg: Label- shootButton: ImageButton- skin: Skin- sound: ImageButton.ImageButtonStyle- soundButton: ImageButton- stage: Stage- style: ImageButton.ImageButtonStyle- touchBackground: Drawable- touchKnob: Drawable- touchpad: Touchpad- touchpadStyle: Touchpad.TouchpadStyle- viewp: Viewport- width: float
<ul style="list-style-type: none">+ dispose(): void+ draw(): void+ HUD()

states
<ul style="list-style-type: none">+ GameOver+ GameStateManager+ HighScores+ HowToPlay+ Menu+ Play+ <i>State</i>

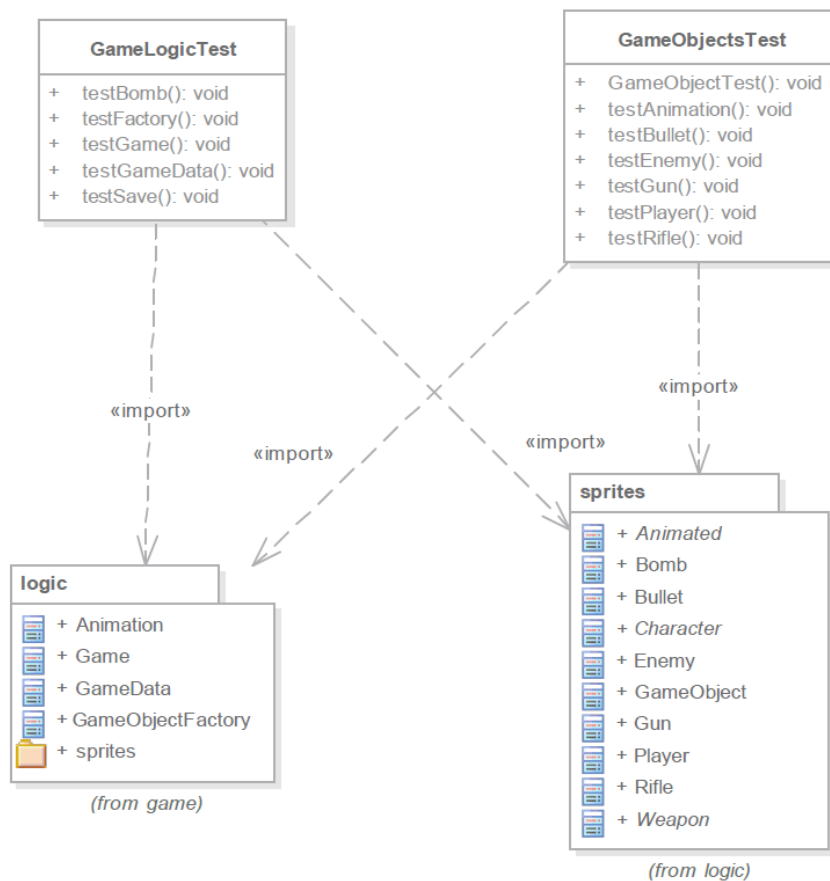
3.2.4. Package States (de gui)



3.2.5. Package áudio



3.2.6. Package tests



3.3. Padrões de Desenho

Um padrão de desenho é uma solução geral que pode ser reutilizada para um problema que acontece frequentemente. Neste projeto implementamos vários padrões de desenho, descritos abaixo, como forma de simplificar e organizar o código.

3.3.1. States

Este padrão foi utilizado na organização da estrutura da aplicação desde o menu inicial até às várias opções. A máquina de estados resultante deste padrão de desenho pode ser vista sobre forma de diagrama no ponto 3.4.

3.3.2. FlyWeight

Este padrão foi utilizado com o intuito de reduzir os custos computacionais da aplicação durante o jogo, fazendo com que vários objetos da mesma classe partilhem determinados atributos.

Foi aplicado dentro da classe *GameObjectFactory* para a partilha de atributos dos diferente objetos do jogo que serão criados (ver 3.3.3), tais como: texturas e animações.

3.3.3. Factory

Este padrão tem com objetivo centralizar a criação de objetos durante o decorrer do jogo e melhorar a gestão do mesmo. Foi aplicado na classe *GameObjectFactory*, uma classe que cria instâncias de *GameObject*, classe que simboliza todos os objetos em jogo. Os objetos desta classe ou derivados da mesma têm que ser frequentemente criados e colocados em jogo, daí este padrão ser bastante útil. Para ser mais específico, os objetos que são instanciados através deste padrão são: zumbis (classe *Enemy*), *AmmoBoxes* (classe *GameObject*), balas (classe *Bullet*) e bomba (classe *Bomb*).

3.3.4. Singleton

O *singleton* foi implementado com o intuito de só haver uma instanciação da classe *Save*, assim apenas um objeto é responsável por todos os ficheiros de entrada/saída do jogo não havendo possíveis conflitos ou perdas de informação ao guardar a informação do jogo

3.4. Mecanismos e Aspetos Importantes

3.4.1. Gestão do Jogo

A gestão do jogo é feita por níveis e com o objetivo de aumentar a dificuldade a cada nível, tivemos que implementar alguns métodos para a gestão de recursos do mesmo. Esta gestão faz-se relativamente a: número de zumbis, número de *ammo box*, recarga e durabilidade de cada arma.

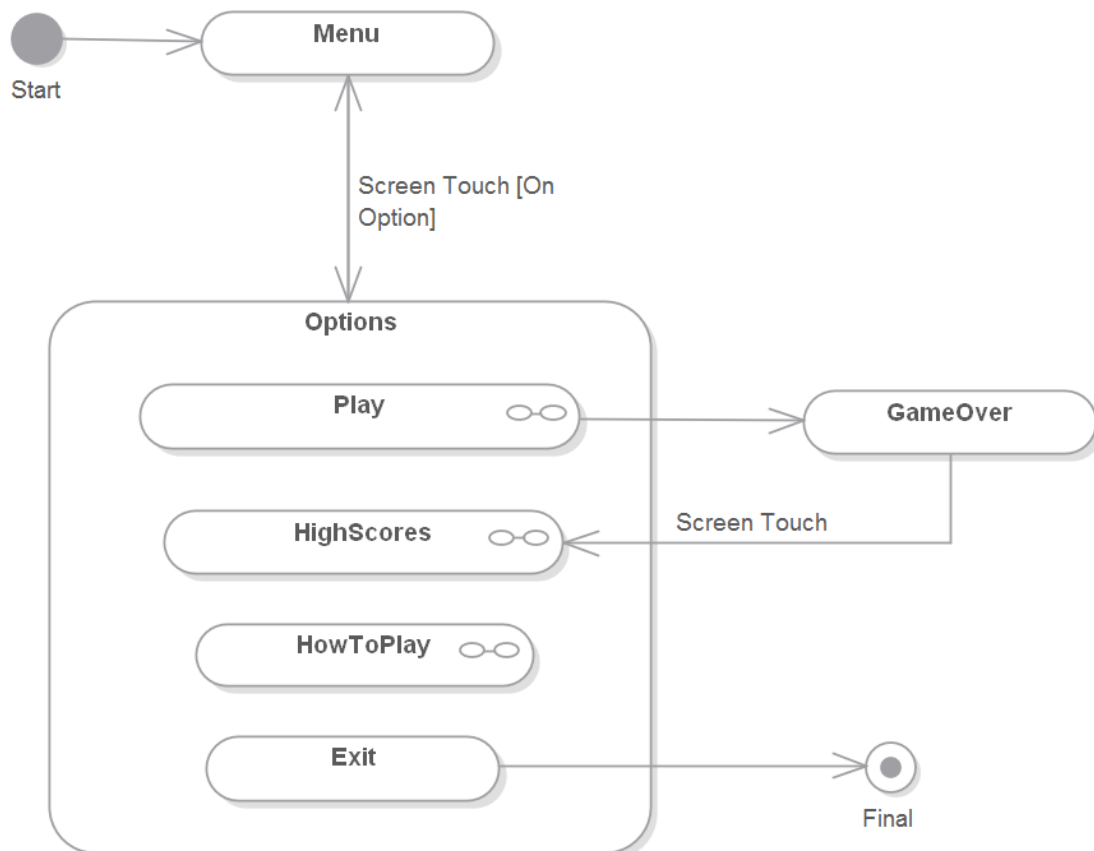
Por nível a gestão é feita da seguinte forma:

- Número de zumbis = $6 + \text{nível}$
- Número de *ammo boxes* = $2 + \text{nível} / 3$
- Recarga da *rifle* = $5 + \text{nível} / 2$
- Recarga da *gun* = $10 + \text{nível}$
- Durabilidade da *rifle* = $2 + \text{nível} / 6$

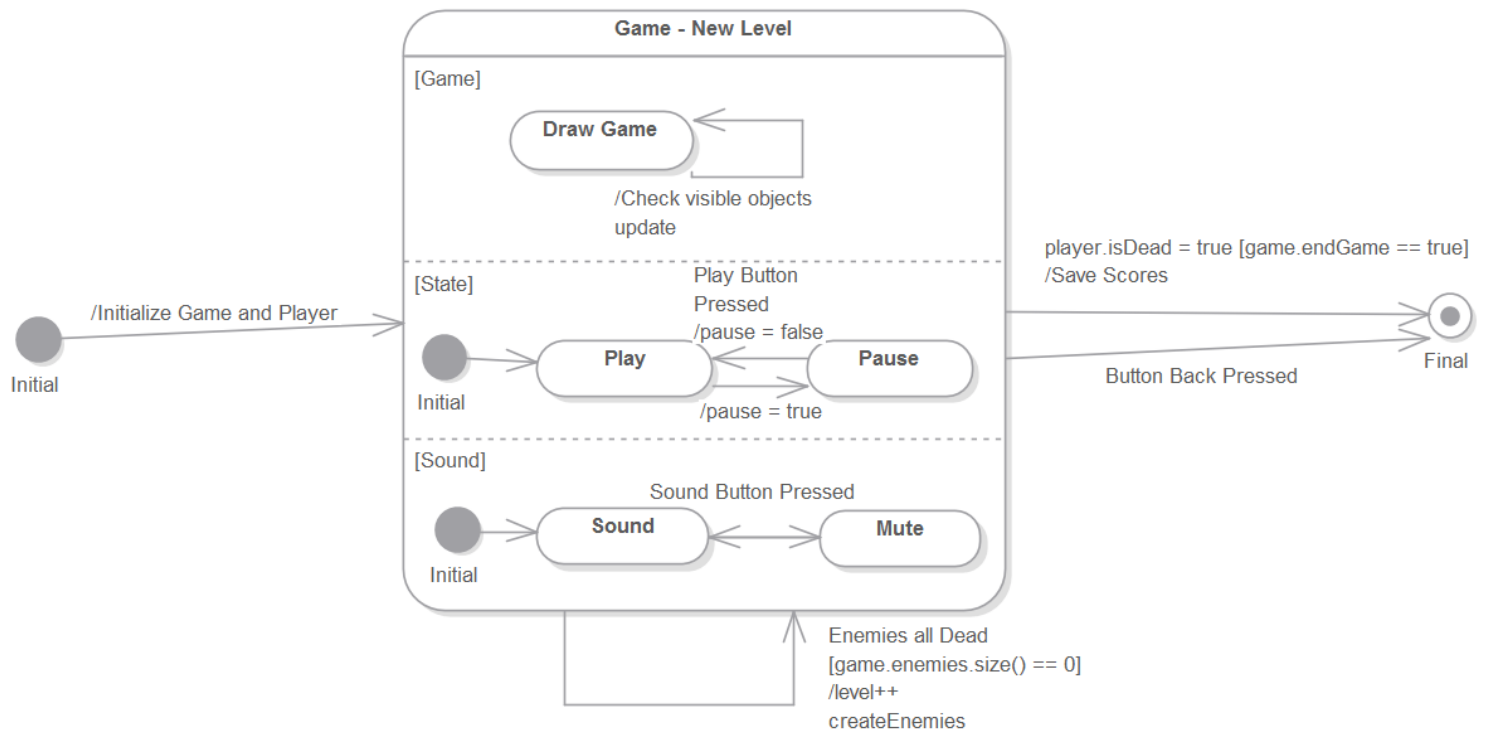
3.4.2. Máquina de Estados

Diagramas de navegação da aplicação

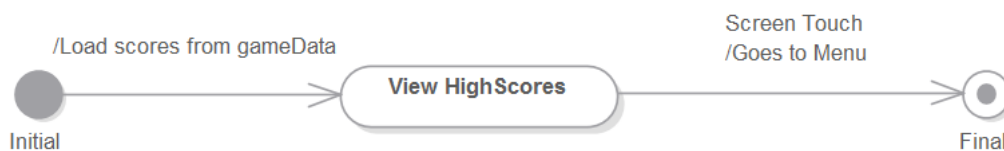
- Máquina de estados do Menu:



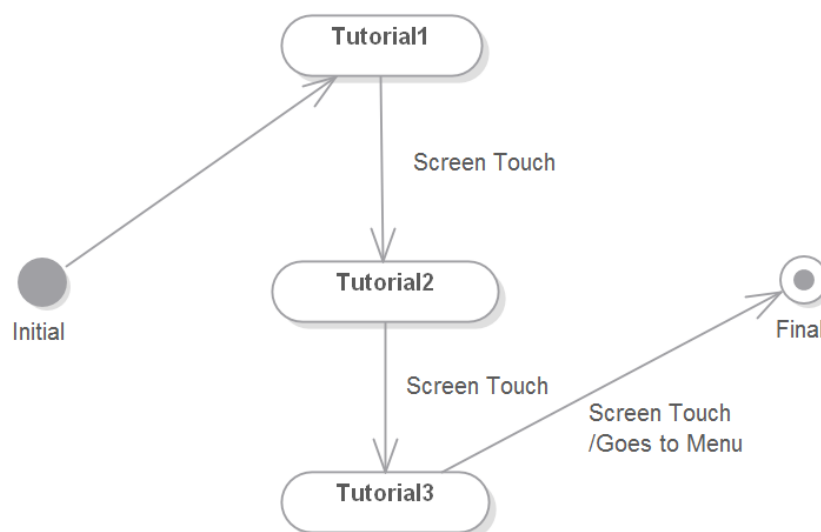
- Máquina de estados de *Play*:



- Máquina de estados *HighScores*:

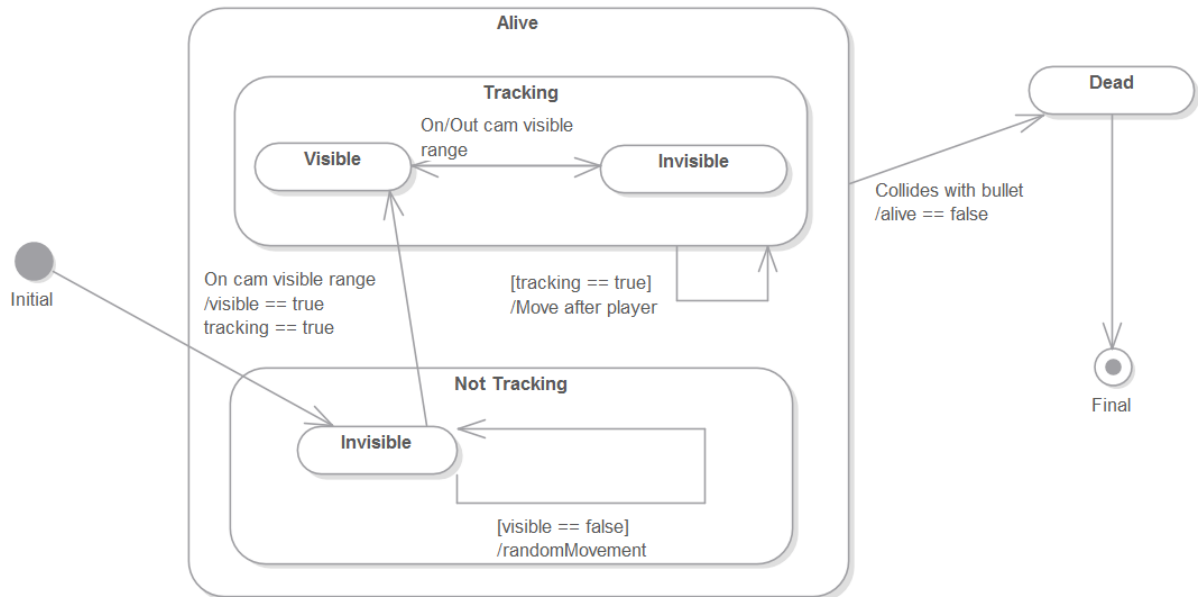


- Máquina de estados *HowToPlay*:

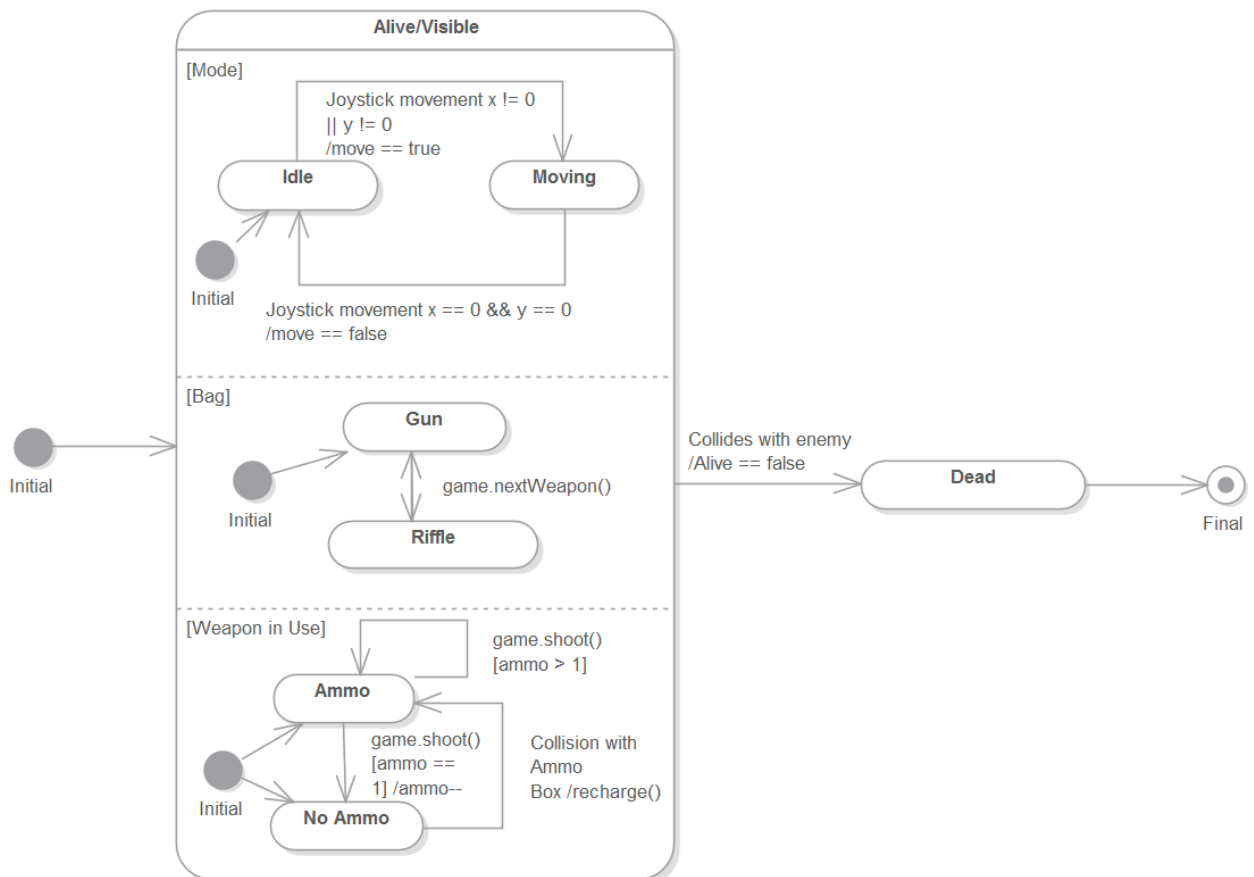


Estados de Objetos em jogo

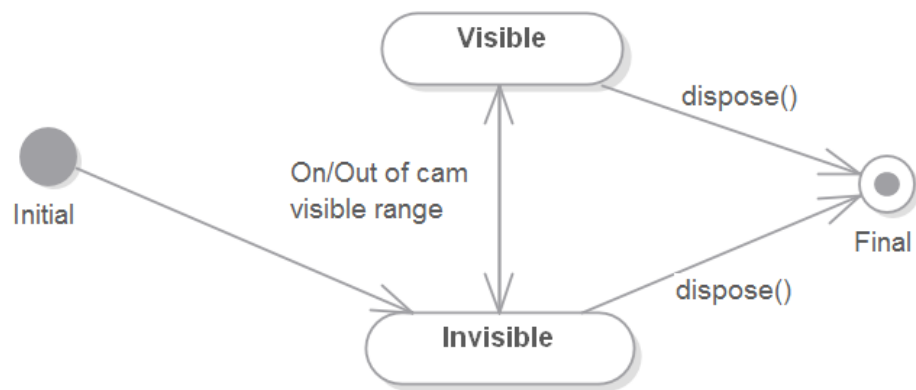
- Enemy (zumbi)



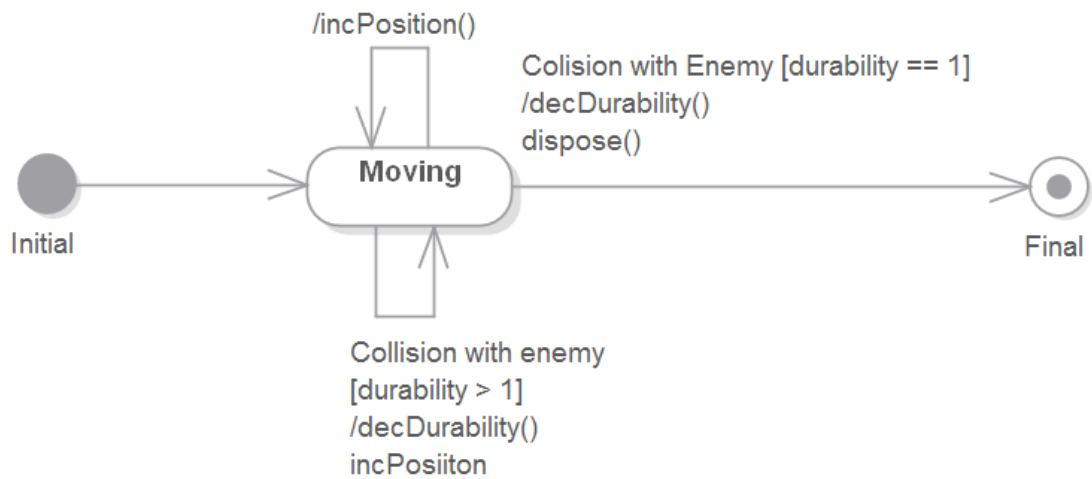
- Player



- GameObject



- Bullet (Bala)



3.5. Ferramentas, Bibliotecas e Tecnologias utilizadas

O ambiente de desenvolvimento utilizado foi o *Android Studio* pois facilita o uso do emulador para o teste do jogo, que neste caso, faz se de uma forma direta.

Para facilitar a implementação do nosso jogo utilizamos o libgdx, uma plataforma *Framework* de desenvolvimento de jogos *open-source* em JAVA. O uso desta *framework* permitiu-nos poupar bastante tempo em funcionalidades desde a parte gráfica à parte da lógica. Para mais informações, o site oficial desta *framework* encontra-se nas referências no final deste relatório.

3.6. Dificuldades

Durante a realização do projeto deparamos com algumas dificuldades relacionadas com a biblioteca “LibGdx”.

A principal dificuldade foi a implementação do “Box2d” que devido à falta de conhecimento e experiência perdeu-se bastante tempo. Em alternativa ao “box2d” foi implementado um sistema de colisões compatível com o tipo de jogo. Também encontramos dificuldade na configuração dos testes com a biblioteca “LibGdx” devido à falta de suporte na configuração para o Android Studio.

3.7. Testes Realizados

Para a garantia de qualidade do projeto final foram realizados testes manuais e testes automáticos que serão descritos de seguida.

Na categoria de teste automáticos foram implementados serão apresentados por classes de lógica presentes no projeto.

- GameObject:
 - Teste das funções de alteração e retorno de variáveis.
 - Teste da translação do objeto.
- Player
 - Teste das funções de alteração e retorno de variáveis.
 - Teste da arma equipada
 - Teste do número de armas na mochila
- Rifle
 - Teste das funções de alteração e retorno de variáveis.
 - Teste da alteração da animação
- Gun
 - Teste das funções de alteração e retorno de variáveis.
- Enemy
 - Teste de inicialização do objeto.
- Bomb
 - Teste de inicialização do objeto.
- Bullet
 - Teste do movimento do objeto.
- Character
 - Teste é realizado nas classes derivadas.
- Animation
 - Teste das funções de alteração e retorno de variáveis.
- GameData
 - Teste da inicialização do objeto.

- Adição de novos resultados
- Save
 - Ler conteúdo do ficheiro
 - Gravar conteúdo do ficheiro.
- Animated
 - Testado nas classes derivadas
- Weapon
 - Testado nas classes derivadas.
- Zumbi Spawner
 - Teste na criação de Enemies.
- Bullet Factory
 - Teste na criação de Bullet.
- Ammo Factory
 - Teste da criação das Ammo boxes.
- Game
 - Movimento do player.
 - Disparo da arma.
 - Troca de arma.
 - Teste da criação de Ammo, zombis.

Na categoria de testes manuais foram feitos os seguintes

- Funcionamento da interface.
- Teste de colisões dos objetos do jogo.
- Verificação do funcionamento das animação e sons.
- Interação do Player com os vários objetos de jogo

4. Conclusões

Em geral, cumprimos com todos os objetivos previamente planeados, no entanto, não conseguimos fazer a implementação da versão de jogo *multiplayer* que seria um extra.

Uma das possíveis melhorias que tínhamos em mente seria implementar um mini mapa com a indicação onde se encontram os zumbis e, possivelmente, também as caixas com munições. Também a possibilidade de outros tipos de armas ou itens em posse do jogador seria um assunto a analisar. Além disso, o melhoramento do disparo e a criação de uma animação para a arma.

A contribuição pela parte dos elementos do grupo foi equitativa, tendo cada um efetuado 50% do trabalho. O tipo de contribuições prestadas estão descritas abaixo e pelo elemento do grupo em questão:

Catarina:

- Lógica do jogo e classes necessárias para a sua implementação
- Package “gui” maioritariamente (parte da HUD)
- Tratamento de imagens em *sprites*
- Relatório

Pedro:

- *Animações*
- HighScores e classe GameData
- Sons da aplicação
- Testes
- HUD
- Relatório

5. Referências

www.kenney.nl

www.libgdx.badlogicgames.com

[https://en.wikipedia.org/wiki/HUD_\(video_gaming\)](https://en.wikipedia.org/wiki/HUD_(video_gaming))

<http://stackoverflow.com/questions/16622843/how-do-i-export-a-project-in-the-android-studio>

<https://www.youtube.com/watch?v=a8MPxzkwBwo&list=PLZm85UZQLd2SXQzsF-a0-pPF6IWDDdrXt>