Projeto Final

"Sheep Party"

Laboratório de Programação Orientada por Objetos

8 de junho de 2015

Turma 3
Ricardo Cerqueira – up201304000
Miguel Pereira – up201305998

Índice

Lista de figuras

1. Introdução

No âmbito da unidade curricular de Laboratório de Programação Orientada por Objetos, foi-nos pedido que desenvolvêssemos um programa em Java, aplicando as técnicas de desenho e programação orientada por objetos.

O projeto escolhido foi um jogo multijogador (2-4 jogadores) divertido e cativante que qualquer um pode desfrutar.

Em Sheep Party, um dos jogadores assume o papel de *sniper*, que tem o objetivo de proteger as ovelhas em festa, enquanto os restantes infiltram-se entre os convidados com a intenção de os matar.

Este relatório tem como objetivo explicar o funcionamento da aplicação desenvolvida e mostrar de que maneira esta foi implementada. Inicialmente, irá fazer-se uma breve descrição da aplicação e explicar-se-á detalhadamente o seu funcionamento. De seguida, será realizada uma explicação da implementação feita, recorrendo aos diagramas de classes/*packages* apropriados e às bibliotecas e padrões de desenhado utilizados.

2. Manual de utilização

O utilizador deve iniciar a aplicação no computador e, através da leitura do código QR ou do *link* fornecido, fazer *download* da aplicação Android para o seu tablet ou telemóvel. Assim que a aplicação tiver sido instalada, carrega-se em "play". Nas definições, a opção "Fontes desconhecidas" deve estar ativada.

Figura 1. Menu inicial

No *lobby*, os utilizadores devem fazer uso da funcionalidade de *scan* da aplicação Android para ler o código QR apresentado, que encripta o IP e a porta de rede a serem utilizados. Desta feita, o utilizador estará ligado em rede e deverá aparecer no *lobby* onde pode escolher, através do botão "A", ser uma personagem ou o *sniper*.

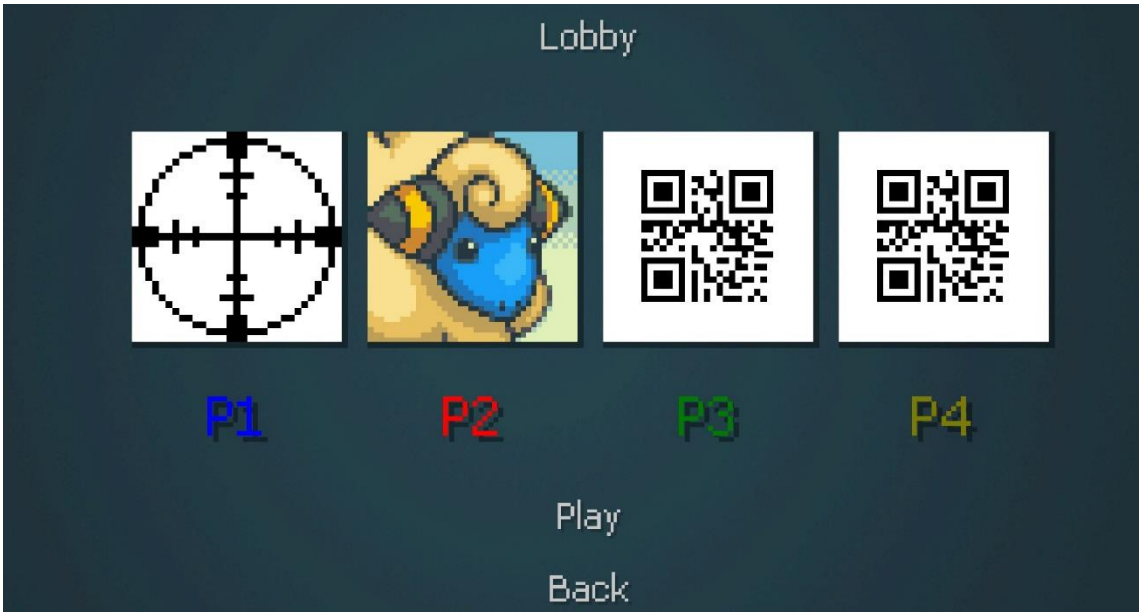

Figura 2. *Lobby*

Assim que estiverem ligados, no mínimo, dois jogadores – um deles tem de ser obrigatoriamente o *sniper* – o jogo pode ser iniciado carregando em “play”. A partir daqui, os jogadores devem utilizar o controlador da aplicação Android. O *joystick* move a mira do *sniper* ou a ovelha controlada pelo jogador em todas as direções, conforme o papel a desempenhar escolhido no *lobby*. O botão “A” faz o *sniper* disparar um tiro. No caso dos atores, este botão serve para matar os outros personagens. Os restantes botões não têm qualquer funcionalidade.

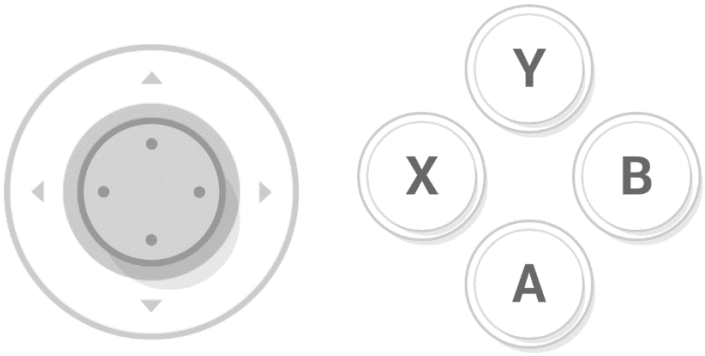

Figura 3 - Aplicação Android (controlador)

O mapa do jogo encontra-se na escuridão, que esconde o estado atual das ovelhas. Apenas o sniper possui uma luz que rodeia a sua mira e que deverá incidir sobre as ovelhas para as revelar. O objetivo dos atores é matar o maior número de ovelhas possível; para tal devem atacá-las quando estas se encontram próximas. O ataque é acompanhado de uma animação que apenas é visível aos jogadores se o atacante se encontrar na luz e, caso seja bem-sucedido, a ovelha atingida irá balir. No entanto, esta continuará a andar pelo mapa até que a luz a revele, ficando então estática e escurecida.

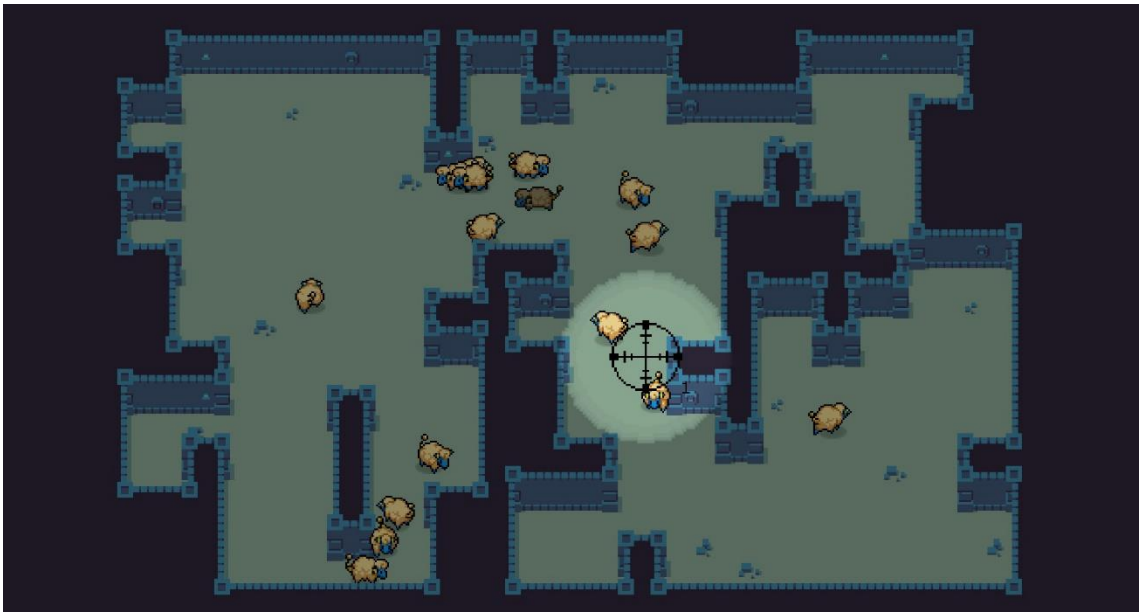

Figura 4. Jogo

O sniper deve deduzir qual ou quais das ovelhas são as assassinas com base no seu movimento e nos padrões observados nas suas vítimas, e matá-las. Os atores, por outro lado, devem simular o comportamento dos convidados de modo a passarem despercebidos, tentando matar os convidados. A tarefa do sniper é ainda tornada mais difícil devido ao número limitado de balas ao seu dispor.

O jogo termina assim que o destino das ovelhas inocentes possa ser inequivocamente determinado. Isto é, quando:

- As ovelhas inocentes morrem todas, caso em que os atores cumpriram o seu objetivo;
- Os atores morrem todos, ainda sobrevivendo algumas inocentes, salvas pelo sniper;
- O *sniper* fica sem balas, sendo que os convidados não terão ninguém que os proteja.

Quando o jogo termina, há uma indicação da equipa que ganhou, assim como da percentagem de ovelhas inocentes que não sobreviveram, e são, finalmente, reveladas as identidades das ovelhas. Para voltar ao *lobby* deve carregar-se na tecla “escape”.

Figura 5. Final do jogo

A aplicação permite definir o número de ovelhas inocentes no mapa, o número de balas extra do *sniper* (a ser somado ao número de atores em jogo), a altura e a largura do mapa, o volume dos sons produzidos e ligar ou desligar a música de fundo.

Figura 6. Menu de opções

3. Conceção e implementação

3.1. Diagrama de *packages*

A tabela a seguir apresenta a divisão por *packages* feita no projeto, acompanhada pelo respetivo conteúdo de cada *package*.

Package	**Conteúdo/Responsabilidade**
Logic	Lógica do jogo; geração do mapa
Network	Ligação cliente-servidor usando TCP
Pathfinder	Determinação do caminho mais curto usando o Algoritmo A*
Audio	Gestão dos sons do jogo
GUI	Interface do jogo
Control	Controlo das personagens pelo utilizador/computador
AndroidGUI	Interface da aplicação Android
Test	Testes unitários

Tabela 1 - Distribuição de packages

E o respetivo diagrama de *packages*:

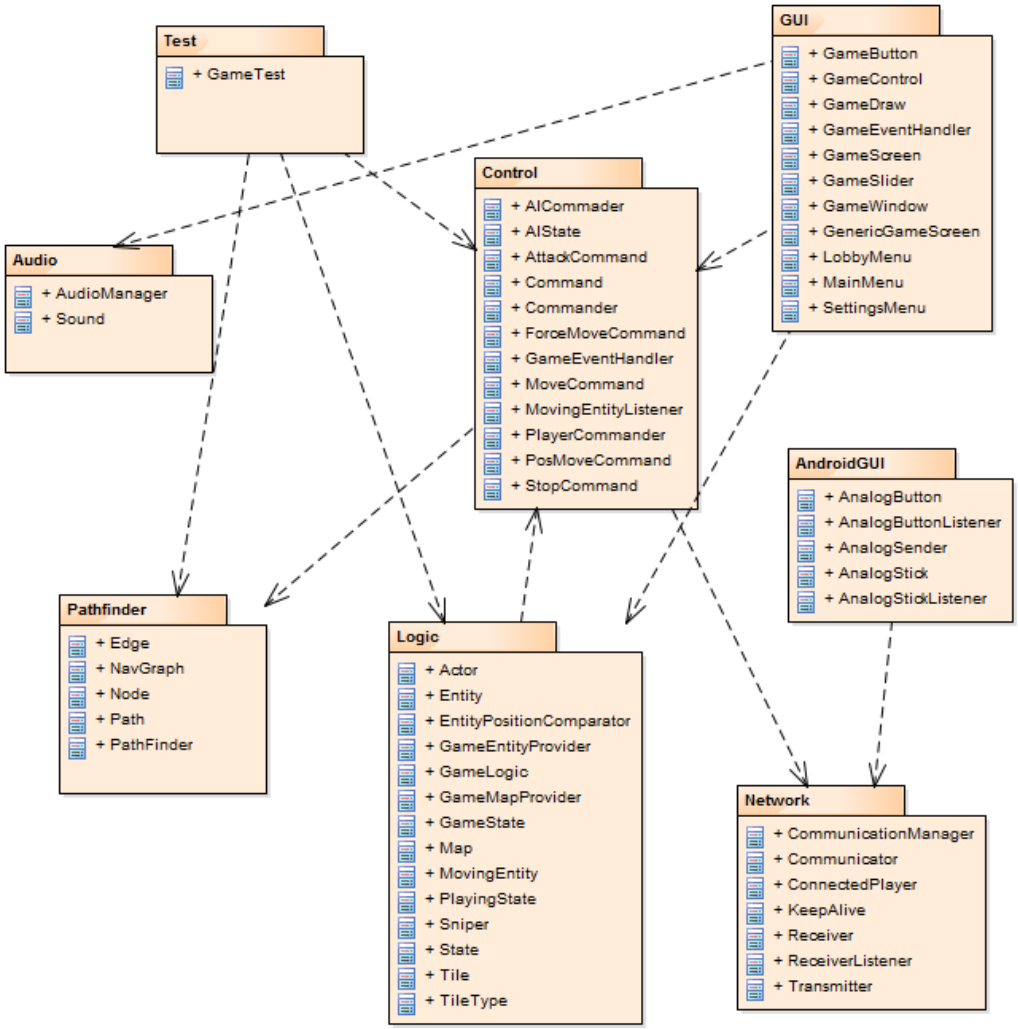

Figura 7. Diagrama de *packages*

3.2. Diagrama de classes

3.2.1. Logic

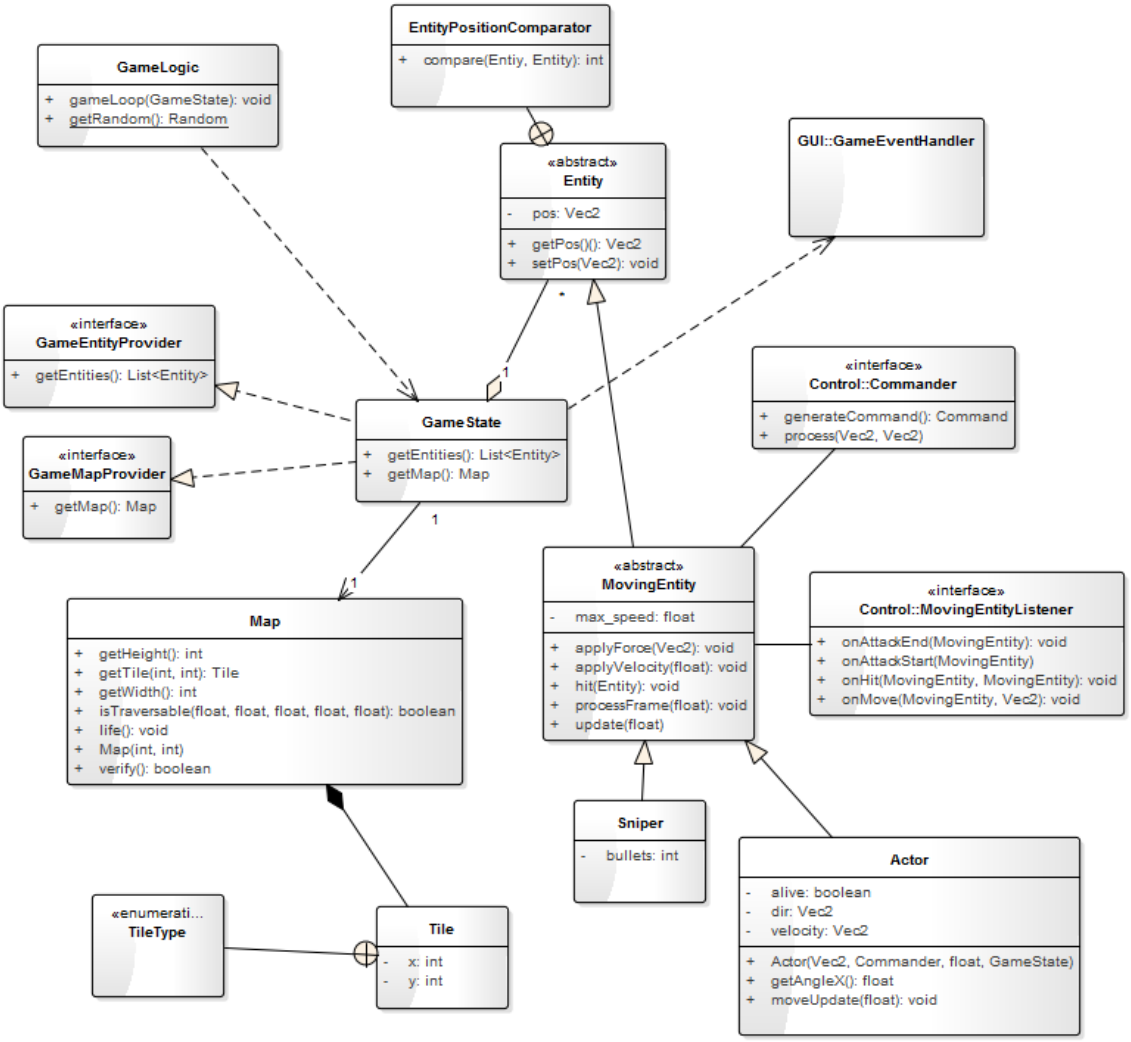

Figura 8. Diagrama de classes para o *package* Logic

Map representa o mapa do jogo e *Tile* uma parte desse mapa. *Entity* é uma superclasse que representa as entidades do jogo – *Sniper* e *Actor*. *GameState* contém o mapa e todas as entidades do jogo. *GameLogic* contém a lógica do jogo.

3.2.2. Network

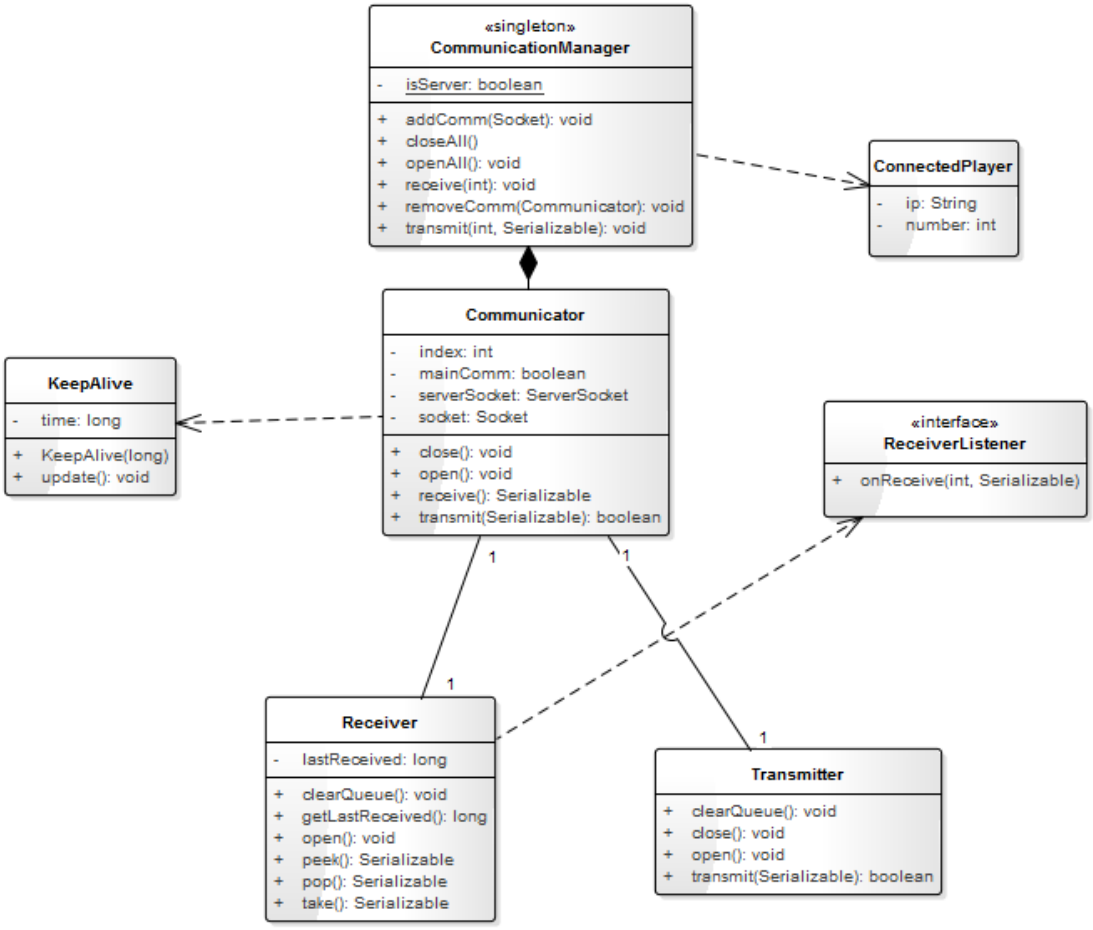

Figura 9. Diagrama de classes para o *package* Network

Communicator gere um dos pontos de acesso num determinado dispositivo e recebe a ligação, sendo responsável por detetar falhas nesta. *Receiver* é o recetor de mensagens e *Transmitter* o transmissor. *KeepAlive* envia uma mensagem periódica de modo a detetar uma eventual perda de ligação.

3.2.3. Pathfinder

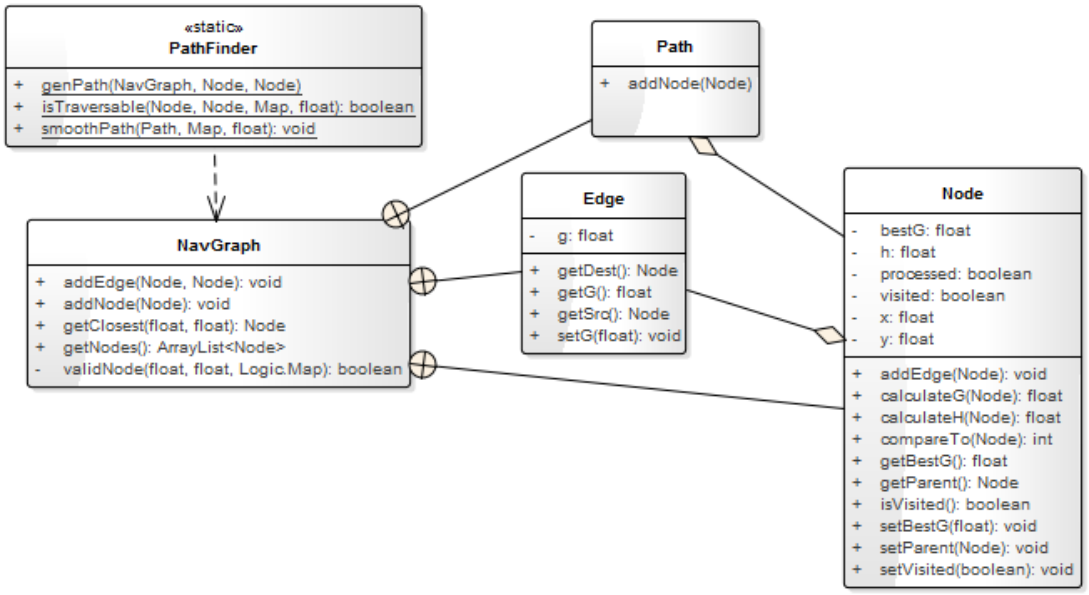

Figura 10. Diagrama de classes para o *package* Pathfinder

Pathfinder é uma classe estática que implementa o Algoritmo A* sobre um grafo de modo a determinar o caminho mais curto, bem como funções para melhorar o caminho obtido. *NavGraph* representa o grafo, que é gerado a partir do mapa do jogo, sobre o qual é efetuada a pesquisa do caminho mais curto.

3.2.4. Audio

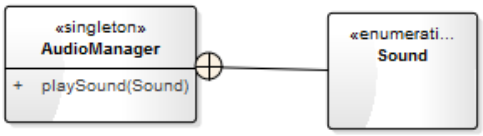

Figura 11. Diagrama de classes para o *package* Audio

AudioManager contém todas as funções de controlo do áudio, tais como reprodução de um determinado som ou a especificação do volume dos sons.

3.2.5. GUI

Figura 12. Diagrama de classes para o *package* GUI

GameDraw contém as imagens do jogo e é responsável pelo desenho de todos os elementos da interface gráfica da aplicação. *GenericGameScreen* é uma superclasse que representa os ecrãs da aplicação. *GameButton* representa os botões da aplicação, tais como aqueles que se podem encontrar no menu inicial e *GameSlider* representa os *sliders* do menu das opões.

3.2.6. Control

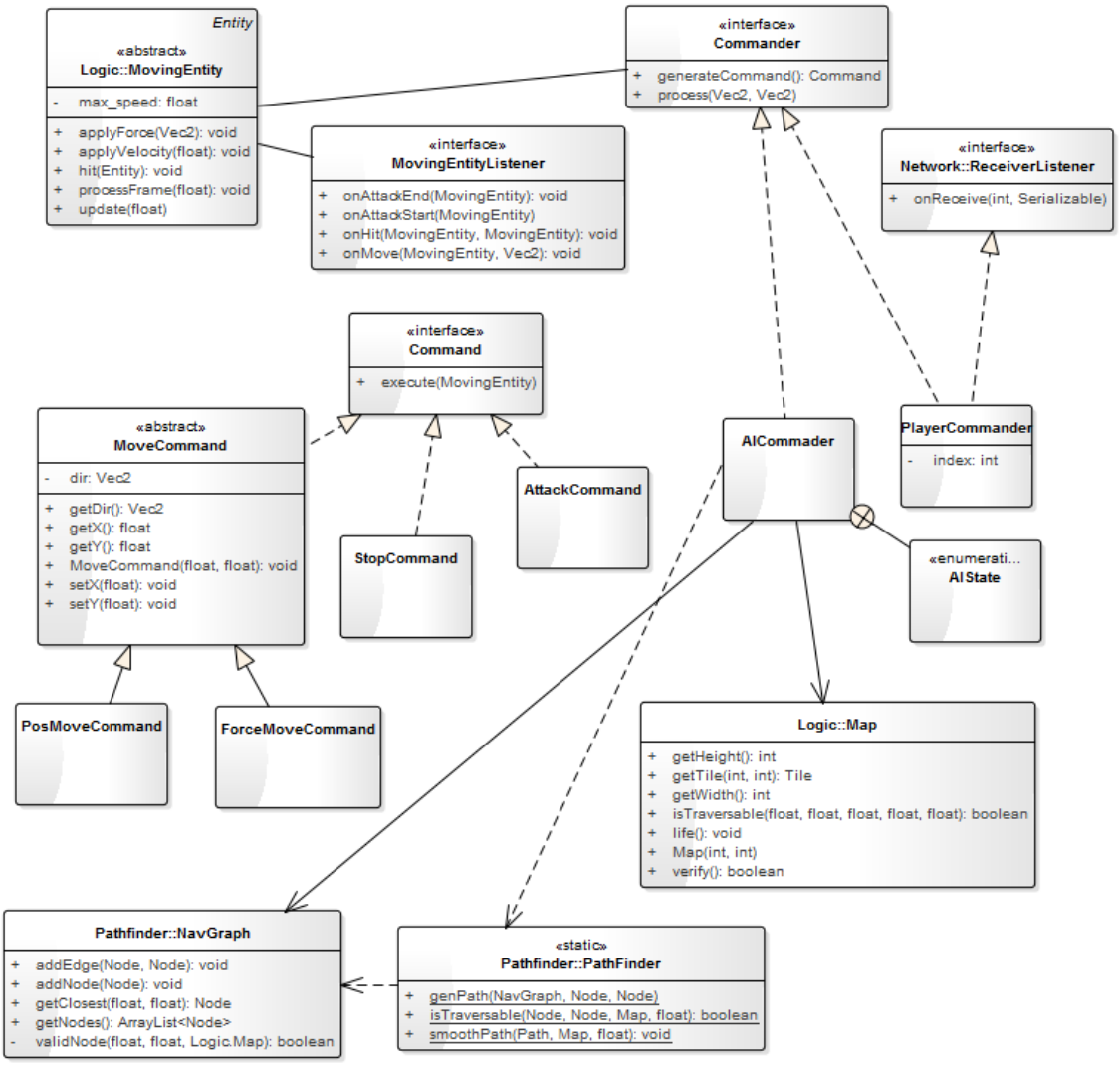

Figura 13. Diagrama de classes para o *package* Control

Commander é uma interface que representa uma entidade que controla um personagem do jogo. *AICommander* permite ao jogo controlar um dos personagens. *PlayerCommander* permite a um jogador controlar um dos personagens. *Command* representa um comando proveniente do jogador ou do jogo que é executado sobre um personagem do jogo.

3.2.7. AndroidGUI

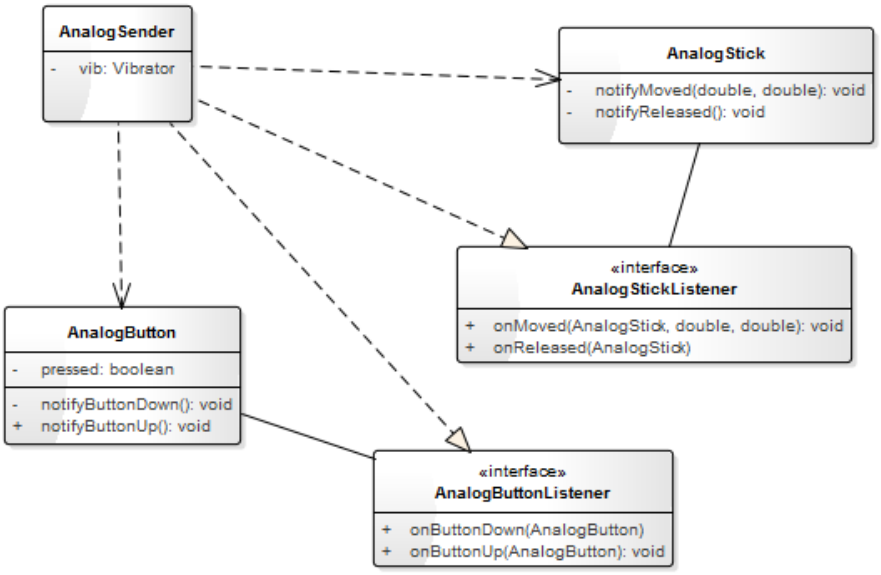

Figura 14. Diagrama de classes para o *package* AndroidGUI

AnalogButton representa os botões do controlador na aplicação Android e *AnalogStick* representa o *joystick* do controlador.

3.2.8. Test

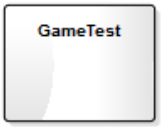

Figura 15. Diagrama de classes para o *package* Test

GameTest contém os testes unitários da aplicação.

3.3. Padrões de desenho

Os padrões de desenho utilizados foram os seguintes:

- *Observer* – permite notificar outras classes de determinados eventos, nomeadamente do movimento do *joystick*, do clique dos botões e da receção de informação da rede;
- *Command* – utilizado nas classes “Command”, permite o tratamento uniforme dos personagens quer estes sejam controlados pelos jogadores quer pela inteligência artificial;
- *Singleton* – utilizado, por exemplo, no “CommunicationManager”, permite ter um ponto de acesso único à componente de rede;
- *Null Object* – utilizado em substituição dos *commanders* para realizar testes unitários, uma vez que que estes objetos têm um comportamento previsível e sem efeitos secundários.

3.4. Bibliotecas

Utilizaram-se as seguintes bibliotecas:

- ZXing – geração do código QR com a indicação do IP e número da porta a que os clientes se devem conectar;
- ZBar – leitura do código QR na aplicação Andoid.

3.5. Dificuldades

As principais dificuldades foram encontradas na sincronização de *threads* devido à comunicação entre a aplicação Android e aplicação do computador e na geração de grafos a partir do mapa e de caminhos a partir do grafo, isto é, colocar os personagens a mover-se por um percurso gerado pelo Algoritmo A*.

4. Conclusão

4.1. Cumprimento dos objetivos

Este projeto engloba a totalidade das matérias estudadas, onde se incluem interface gráfica, manipulação de ficheiros, animação de objetos gráficos e funcionamento em rede. Pode dizer-se, portanto, que cumpre todos os objetivos propostos.

4.2. Melhorias

Uma melhoria possível seria a adição de mais modos de jogo, onde os jogadores teriam objetivos diferentes, como por exemplo apanhar certos objetos ou ir até determinados pontos do mapa.

4.3. Contribuições

- Ricardo Cerqueira – 55%
- Miguel Pereira – 45%