Faculdade de Engenharia da Universidade do Porto

Sprint – Projeto final de LPOO

RELATÓRIO

JOÃO CABRAL

JOÃO MOTA

1 INTRODUÇÃO

1.1 OBJETIVO DO RELATÓRIO

Serve o presente relatório para apresentar o processo de desenvolvimento do trabalho prático final realizado para a cadeira de Laboratórios de Programação Orientada a Objetos do 2º ano do Mestrado Integrado em Engenharia Informática e da Computação.

1.2 OBJETIVO DO PROGRAMA

Pretendeu-se, com o desenvolvimento deste programa, elaborar um pequeno jogo de corridas de carros com suporte a vários jogadores, controlável remotamente a partir de um dispositivo *Android*, estando a lógica do jogo isolada num servidor central ao qual os vários jogadores se ligam.

1.3 ESTRUTURA DO RELATÓRIO

2 MANUAL DE UTILIZAÇÃO

2.1 FUNCIONALIDADES SUPORTADAS

O jogo suporta a utilização concorrente de até cinco jogadores, que se devem ligar através de uma rede *LAN*, usando o IP facultado pelo servidor central. O servidor central suporta também, para além da sua funcionalidade principal, a utilização de uma suite de testes.

Esta também implementada funcionalidade para mostrar, aquando do fim do jogo, os resultados obtidos no decorrer do mesmo por cada utilizador numa tabela classificativa final.

2.2 INSTALAÇÃO DO PROGRAMA

O programa servidor não necessita de nenhuma instalação especial, bastando correr o ficheiro *.jar* incluído. Para efetuar a ligação ao servidor por parte dos dispositivos *Android* é necessária a instalação no mesmo do ficheiro *apk* incluído com o projeto.

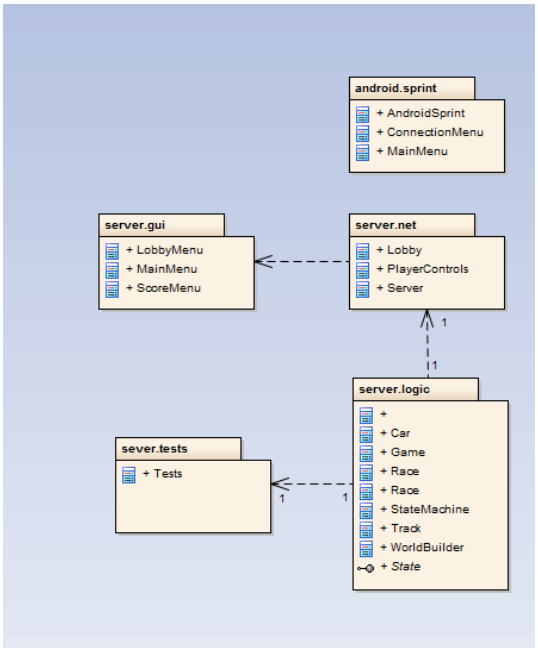
2.3 ARRANQUE E UTILIZAÇÃO DO PROGRAMA

Para o arranque do servidor basta clicar em *Start Server* no menu principal da aplicação de servidor[1]. Para o arranque do cliente basta iniciar a aplicação e escolher a opção *Enter Game* e facultar o endereço fornecido pelo servidor no seu ecrã principal. O telemóvel deve estar ligado à mesma rede *LAN* do servidor. Em alternativa a rede da máquina do servidor deve estar configurada para reencaminhar o tráfego da aplicação para o servidor, devendo nesse caso ser facultado aos clientes o endereço IP adequado. Neste caso basta que os clientes estejam ligados à internet, não sendo necessário estar na mesma *LAN* que o servidor.

[1] O porto 8888 deve estar aberto nas configurações de rede.

3 CONCEÇÃO E IMPLEMENTAÇÃO

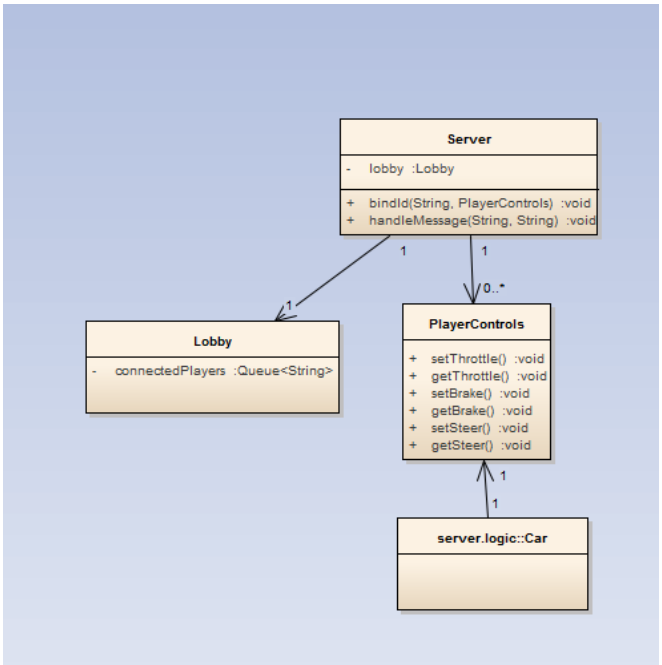
3.1 ESTRUTURA DE PACKAGES



Nome da package	**Propósito**
android.sprint	Contêm todas as classes usadas no cliente android remoto
server.net	Packages responsáveis pela comunicação com o cliente remoto
server.logic	Responsável por lidar com a física do jogo e manter o estado atual do programa
server.tests	Implementa os testes unitários
server.gui	Mantêm a interface gráfica com o utilizador, usando a API do LibGdx

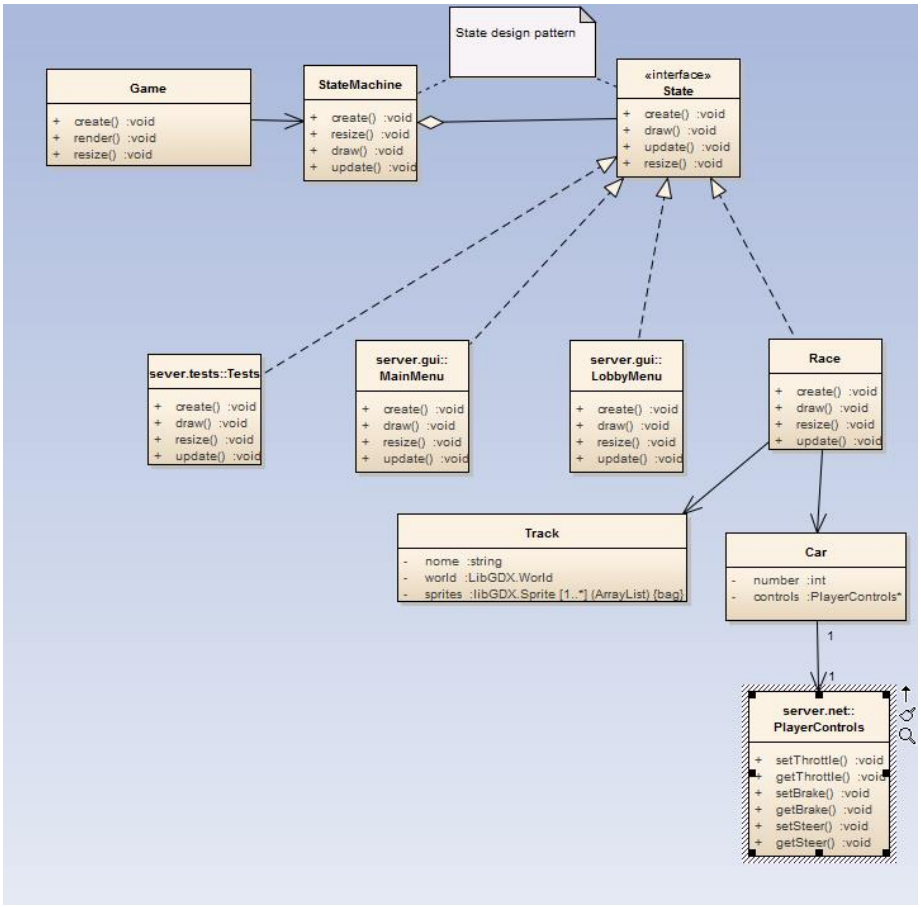
3.2 ESTRUTURA DE CLASSES

3.2.1 server.net



Nome da Classe	**Propósito**
Server	Responsável pelos detalhes de implementação da comunicação com os clientes, incluido a implementação do protocolo utilizado
Lobby	Responsável por gerir a espera dos jogadores antes do jogo começar, bem como de manter a fila de espera caso haja mais jogadores do que espaço disponivel.
PlayerControls	Responsável por fazer a ligação entre os jogadores e a lógica do carro que os mesmo controlam.

3.2.2 server.logic



Nome da Classe	**Propósito**
StateMachine	Responsável por guardar o estado atual da máquina de estados e por chamar o comportamento do estado atualmente ativo
Game	Responsável por implementar os comportamentos exigidos pela API do *LibGdx* e por inicializar a maquina de estados
Race	Responsável pela lógica principal do jogo, incluido mostrar o seu estado atual.
Car	Responsável por interagir com a representação fisica do carro de cada jogador de acordo com os *inputs* do mesmo.
Track	Responsável por definir os limites da geometria da pista e as respetivas colisões.

3.2.3 server.gui[2]

Nome da classe	Propósito
MainMenu	Responsável por mostrar ao utilizador o menu principal do servidor, e em seguida transferir o controlo da *state machine* para a classe apropriada
LobbyMenu	Reponsável por mostrar ao utilizador o estado atual da classe *Lobby*, o tempo que falta para o começo do jogo, quantos jogadores estão atualmente ligados e por transferir o controlo para a classe *Race.*
ScoreMenu	Responsável por mostrar o resultado da corrida que acabou e transferir posteriormente o controlo para a classe *LobbyMenu.*

3.3 Padrões de Desenho Utilizados

Foi utilizado o padrão de desenho *State* na implementação da aplicação servidor para permitir o fluxo entre os vários estados em que esta aplicação se pode encontrar, como sejam o menu principal, o *Lobby* onde os jogadores se ligam antes do inicio do jogo e o modo de jogo propriamente dito. O recurso a este padrão de desenho permitiu evitar a utilização de lógica condicional demasiado extensa e complexa, de dificil manutenção e expandibilidade no futuro.

3.4 Mecanismos Importantes

Os dois principais mecanismos sobre o qual se baseia o programa implementado são a conectividade fornecida pela classe *Server* e a maquina de estados implementada pela classe *StateMachine*

3.5 Bibliotecas Utilizadas

Foi utilizada a biblioteca *LibGdx,* que incluiu o simulador de física *Box2D* utilizado na implementação da lógica do movimento dos carros e deteção da sua colisão, e uma *API* para a programação de interfaces multi-plataforma.

[2] Estas classes estão ligadas entre si pela implementação da State Machine anteriormente descrita, não contendo mais nenhuma relação que seja relevante, pelo que não é incluido o seu diagrama de classes.

4 CONCLUSÕES

4.1 GRAU DE CUMPRIMENTO DOS OBJETIVOS

Foram implementadas todas as funcionalidades que estavam previstas inicialmente, quer de um ponto de vista da lógica de jogo, quer do ponto de vista da utilização de redes.

4.2 MELHORIAS POSSÍVEIS

- Implementar, no servidor, a possibilidade da comunicação ocorrer em ambos os sentidos em qualquer momento, ou seja, implementar conectividade *Full-Duplex.* Neste momento o servidor central apenas pode comunicar com os clients após receber uma mensagem.
- Refinar a apresentação gráfica do projeto, quer ao nível dos *assets* utilizados no jogo propriamente dito, quer ao nível da interface com o utilizador.~

4.3 CONTRIBUIÇÃO DOS ELEMENTOS DO GRUPO

Ambos os elementos do grupo participaram de uma forma equitativa na realização do trabalho, ajudando em todas as tarefas necessárias sem que tenha uma distribuição rigorosa das mesmas.