Bowling



Desenvolvido por:

António Ramadas up201303568

Pedro Carvalho up201306506

Turma 5 | Grupo 7

Índice

[Introdução 3](#_Toc421416917)

[Manual de utilização 4](#_Toc421416918)

[Resumo de funcionalidades suportadas 4](#_Toc421416919)

[Pré-Requisitos 5](#_Toc421416920)

[Instalação do programa 6](#_Toc421416921)

[Arranque do programa 8](#_Toc421416922)

[Modo de utilização 9](#_Toc421416923)

[Formatos de ficheiros 14](#_Toc421416924)

[Conceção e implementação 15](#_Toc421416925)

[Estrutura de packages 15](#_Toc421416926)

[Estrutura de classes 17](#_Toc421416927)

[Padrões de desenho utilizados 19](#_Toc421416928)

[Mecanismos importantes 20](#_Toc421416929)

[Bibliotecas, tecnologias e ferramentas utilizadas 21](#_Toc421416930)

[Dificuldades encontradas e sua resolução 22](#_Toc421416931)

[Conclusões 23](#_Toc421416932)

# Introdução

O relatório visa explicar formalmente o modo de funcionamento de jogo e tecnologias aplicadas. O relatório vai ser um pequeno resumo das diversas etapas de desenvolvimento. Contudo, é possível adiantar que o jogo é uma réplica de um jogo de bowling (daí o nome ser “Bowling”). Desta maneira, existe o suporte para 1 ou 2 jogadores. Caso seja só 1 jogador, o outro será a máquina. Mais detalhes serão referidos mais à frente no relatório.

De referir também que com este programa criamos desafios que estavam fora da nossa zona de conforto. Um sinal deste facto é o do jogo ser em 3D e a comunicação entre dispositivos android e computador. Por conseguinte, este programa não se resume para nós como um simples jogo, mas uma compreensão por novas tecnologias e a utilização delas (incluindo as vantagens e desvantagens inerentes). Podemos deixar desde já antever que alcançamos com sucesso os objetivos traçados.

Concluindo, o relatório pode ser dividido em, pelo menos, 3 partes: o Manual de utilização, a Conceção e implementação e Conclusões. Esta divisão encontra-se mais detalhada no índice.

# Manual de utilização

## Resumo de funcionalidades suportadas

O jogo é uma interação entre o computador e o android. O computador gera um ambiente 3D para calcular a física de jogo, toca música e sons de modo a criar o ambiente de jogo e calcula a tabela de pontuações. Relativamente ao android, este conecta-se ao computador através de um scan de QRcode do computador que disponibiliza o ip local. Adicionalmente, é usado o acelerómetro para captar o movimento do jogador e o android também permite mudar as opções de jogo (tipo de bola e translação).

## Pré-Requisitos

O jogo tem como base uma interação entre jogador e máquina. Para tal, o computador serve apenas como ecrã e centro de processamento, isto é, não é necessário nenhuma operação de introdução de dados. O computador serve então como “output” gráfico. Desta maneira, os dispositivos android serão os controladores. Para simplificar, basta imaginar o computador como uma consola de jogos. Contudo serão agora explicadas as funcionalidades para cada um dos dispositivos:

* O computador: o jogo por ser em java pode ser então executado nos mais variados sistemas operativos que suportem esta tecnologia. Contudo, para o executar existem alguns pré-requisitos como uma ligação de rede sem fios entre o computador e os dispositivos (não é necessária uma ligação à internet), ou ainda um processador e placa gráfica capazes de lidar com um ambiente 3D.
* O controlador: como referido anteriormente, tem de ser android e é recomendável que seja um telemóvel (pela comodidade ao jogar). Recomenda-se então que tenha entre 4 a 6 polegadas. A nível de pré-requisitos: o dispositivo tem de ter acelerómetro, bússola omnidirecional e ligação sem fios ao computador. Atualmente, todos os android fabricados têm estes requisitos físicos.

Enfim, para o desenvolvimento do jogo foi utlizado o ambiente de desenvolvimento de jogo libGDX (<http://libgdx.badlogicgames.com/>) e o gerador de QRcode (<https://github.com/zxing/zxing>).

## Instalação do programa

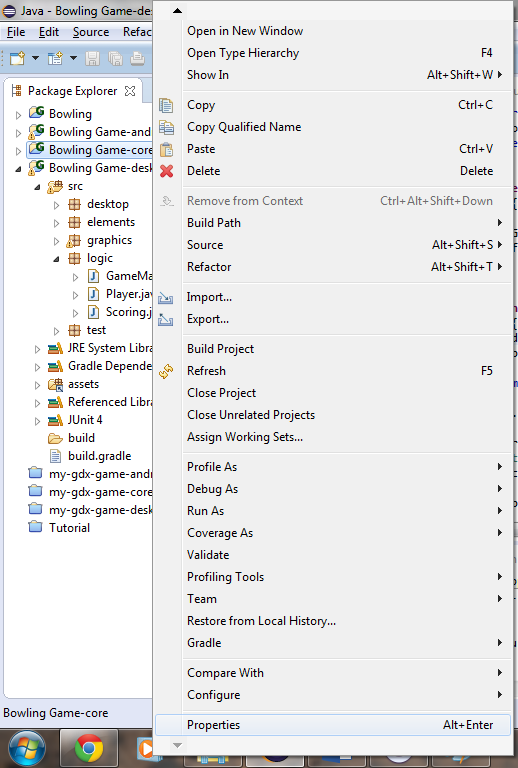
Apesar do uso do libGDX, para importar o projeto no Eclipse, basta importá-lo como projeto Gradle. Contudo, vão surgir alguns erros relacionados com o uso de QRcode. Assim, é necessário fazer o dowload dos ficheiros [core-3.2.0-javadoc.jar](http://repo1.maven.org/maven2/com/google/zxing/core/3.2.0/core-3.2.0-javadoc.jar) e [core-3.2.0.jar](http://repo1.maven.org/maven2/com/google/zxing/core/3.2.0/core-3.2.0.jar) e, depois de importar o projeto, adicioná-los ao build path do projeto adicionando os jar. Se se pretender apensar executar o jar do jogo tem de se ter em atenção que de ter uma pasta com os recursos necessários.

Figura 1 - Passo 1

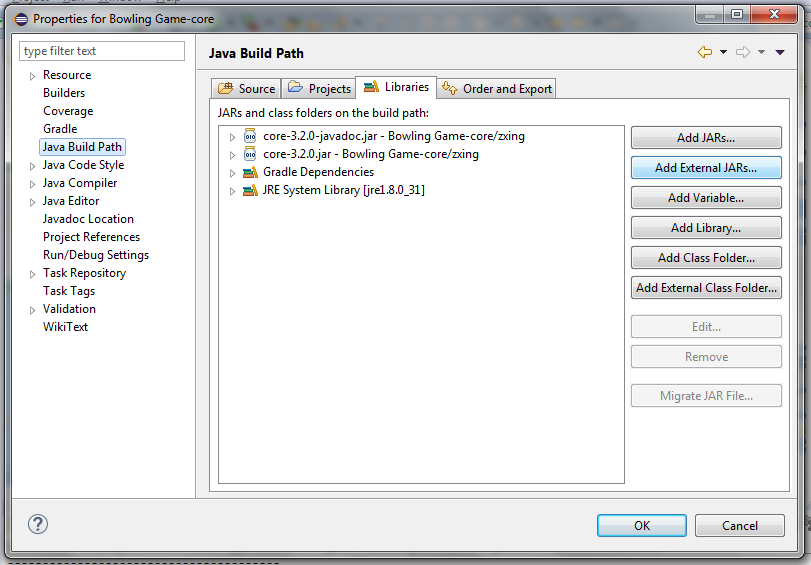


Figura 2 - Passo 2

Para o android é necessário que exista o programa de leitura e interpretação de QRcode (está incluído na maioria dos dispositivos por predefinição).

## Arranque do programa

Por predefinição, o jogo é executado no computador na resolução 1366x768 em ecrã inteiro. O jogo tem de ser inciado separadamente no computador e em cada dispositivo. No máximo podem estar dois dispositivos ligados ao computador. Não importa a ordem em que são iniciados, mas importa a ordem com que se ligam ao computador, pois este é o nervo central do jogo. O android basta escolher um nome de jogador, estar na mesma rede que o computador, apontar o telemóvel ao QRcode e jogar. O jogo quando acabar é terminada aplicação tanto nos dispositivos como no computador.

## Modo de utilização

No computador:

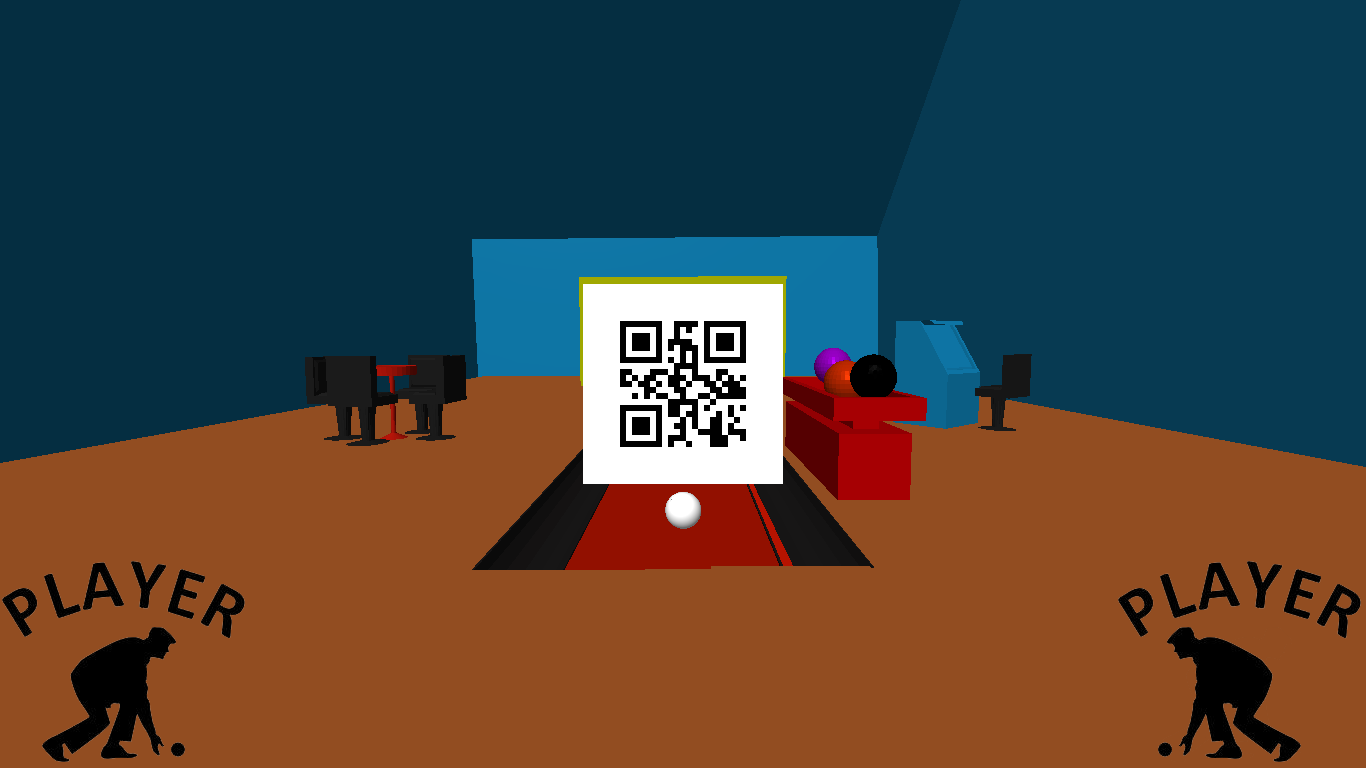


Figura 3 - Ecrã inicial do computador

Como podemos ver, mal é executada a aplicação é mostrado o QRcode para os jogadores se ligarem com o telemóvel. Em cada um dos cantos inferiores é mostrada a ligação dos 2 jogadores. O jogador 1 é o da esquerda. Um jogador verde significa que está feita a ligação, a preto significa espaço disponível (ainda ninguém se ligou).

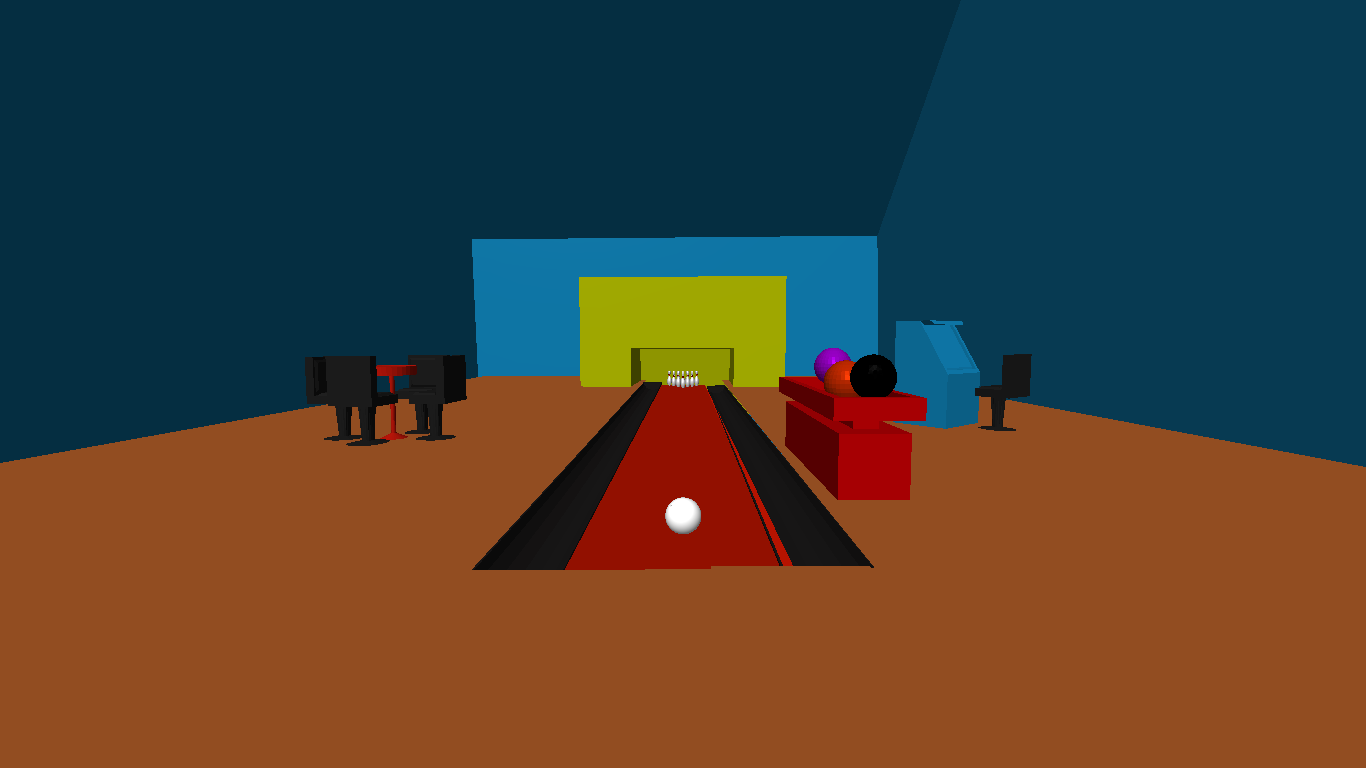


Figura 4 - Lançamento de um jogador

Este é o ecrã de espera do lançamento dos jogadores. Existem 5 bolas com cores e pesos diferentes. A bola, antes do lançamento, pode ser movida para a esquerda ou direita ou escolhida a bola pretendida (a cor muda).

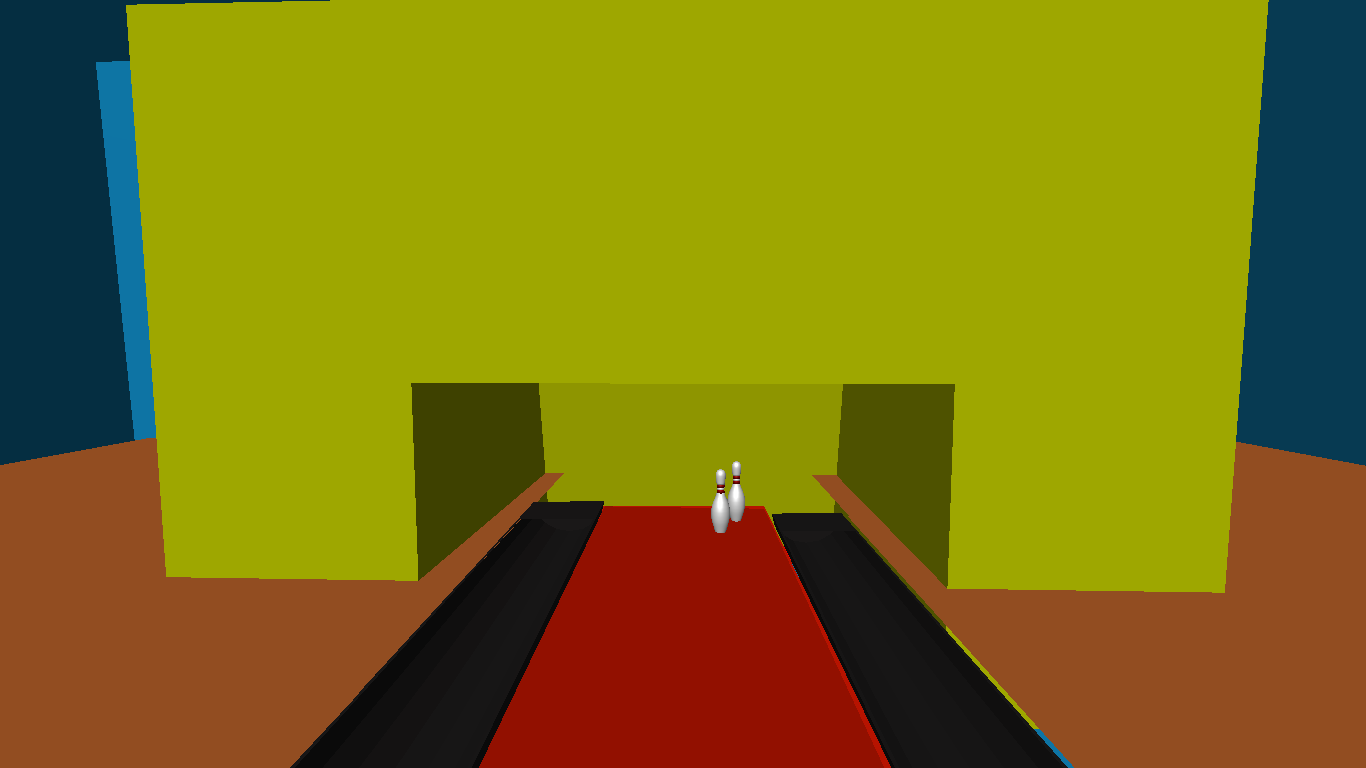


Figura 5 - Exemplo de um lançamento

Exemplo de lançamento de um jogador. Enquanto o jogo corre são tocados sons e música para maior envolvência. No final é mostrada a tabela de pontuações e é tocada uma música diferente. Como podemos ver, o computador apenas serve como centro de dados e processamento e a maior interação está nos controladores android.

No android:

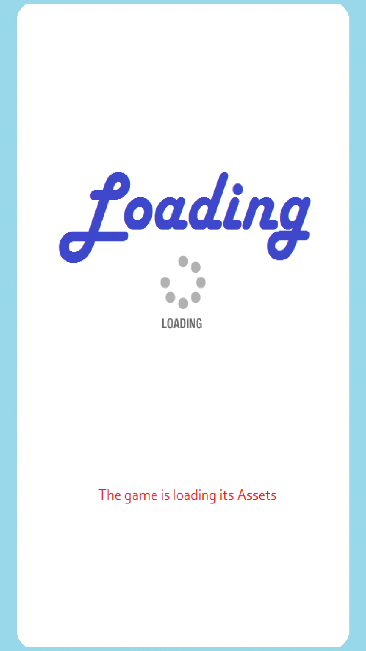
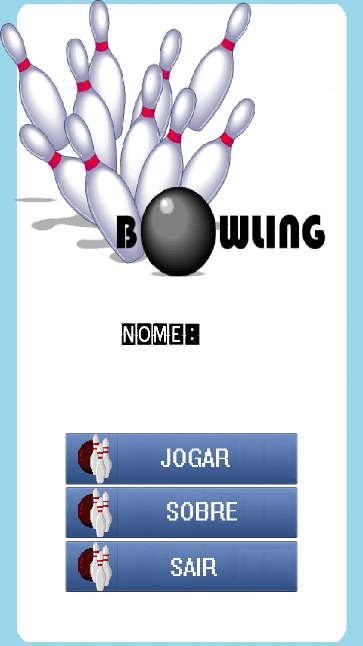
Neste momento estão a ser carregados os recursos necessários para a execução fluída do jogo.

Figura 6 - Ecrã inicial (loading)

Ecrã inicial. O botão “JOGAR” pede a introdução do nome do jogador. O botão “SOBRE” mostra algumas informações. O botão “SAIR” sai do jogo. De referir ainda que se encontra neste ecrã um easter egg que se prolonga pelo jogo todo.

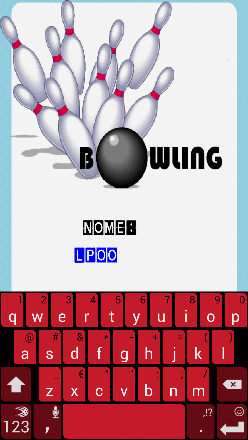
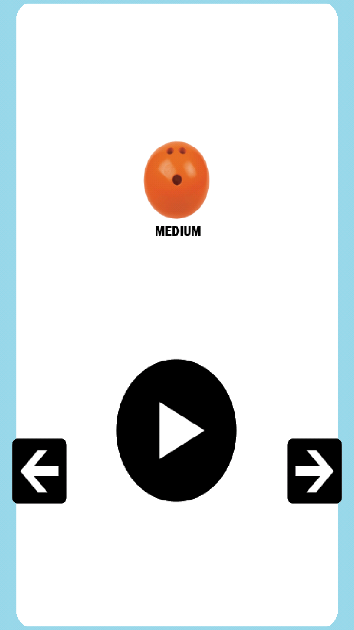
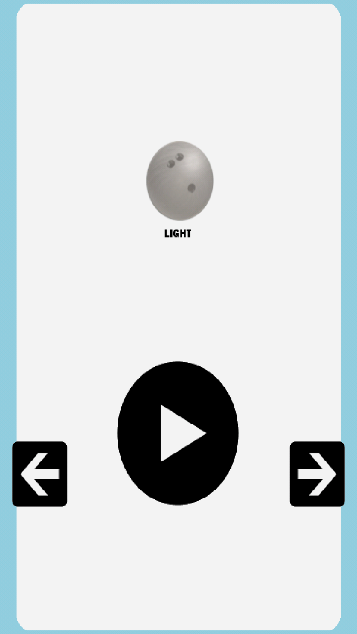


Figura 7 - Informações do jogo

Figura 8 - Ecrã de introdução do nome do jogador (enter para continuar)

Figura 9 - Ecrã inicial



Estão aqui ilustrados dois exemplos para a seleção das bolas. A bola pode ser escolhida com um swype para os lados em cima das bolas. As setas que apontam para fora do ecrã fazem a translação da bola. O botão play inicia a medição do acelerómetro para efetuar a jogada.

Figura 10 - Ecrã de jogadas

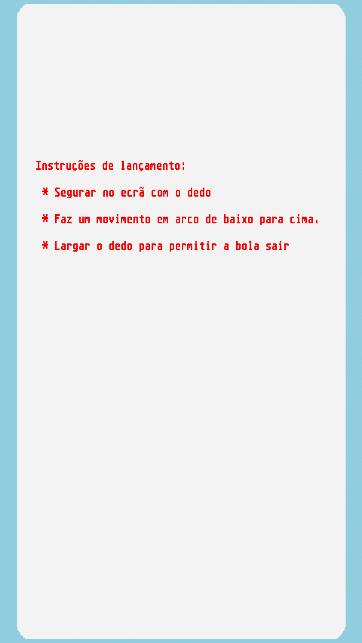
Quando se carrega no play chega-se a este ecrã que mostra as instruções de lançamento. Simplificando, o utilizador tem de fazer o movimento do telemóvel como se fosse uma bola de bowling. Quanto mais tempo o utilizador demora, mais lento é o lançamento. Neste processo, o utilizador tem que ter o ecrã premido desde o início do lançamento até ao final. Um ecrã vermelho significa que o ecrã não está pressionado ou não está na posição inicial correta. Depois passa para o ecrã amarelo que indica que o tempo começou a contar e chegou a uma posição correta. Finalmente chega-se ao ecrã verde em que o ecrã pode deixar de ser premido e o lançamento é efetuado. A inclinação lateral da bola no momento final do lançamento afeta a trajetória da bola.

Figura 11 - Instruções para fazer o lançamento

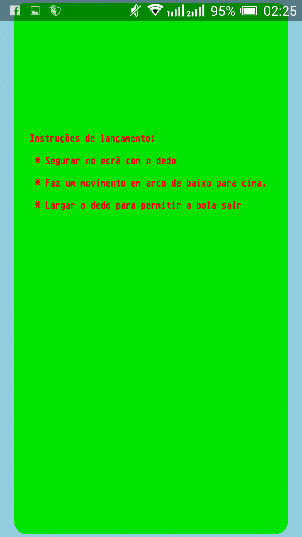
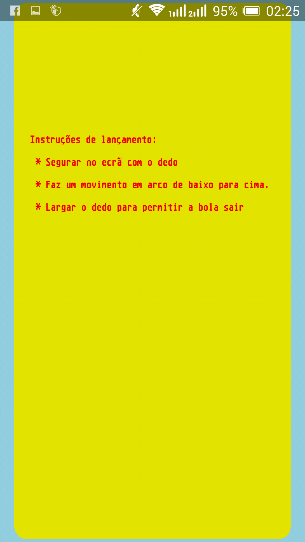
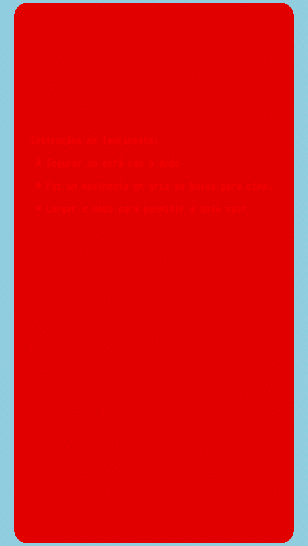
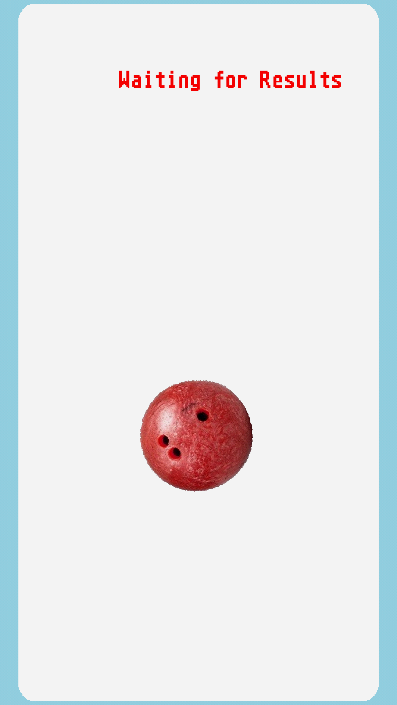


Figura 12 - Os 3 estados do lançamento



Este ecrã indica que a ação se está a passar no computador. Por outras palavras, a bola foi lançada!

Figura 13 - Lançamento da bola



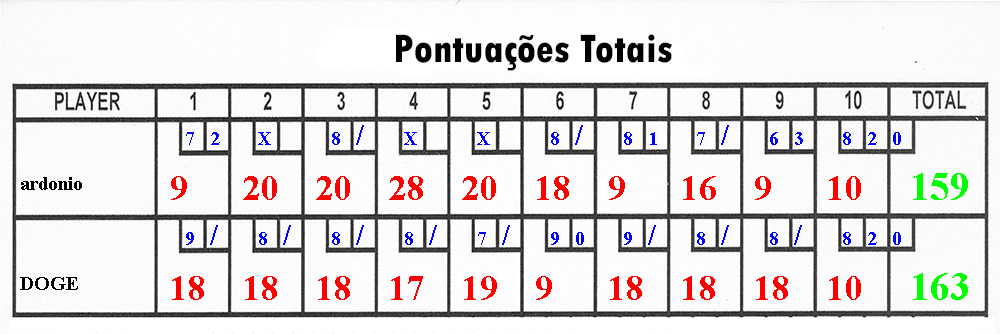
Figura 14 - Ecrã das pontuações

Este é o ecrã dos scores e é mostrado ao jogador que não está a fazer a jogada. O segundo nome é o nome do outro jogador. Em baixo dos nomes estão as respetivas pontuações. A coroa ao lado do nome indica quem está atualmente a ganhar. Este é o ecrã de transição para o ecrã das jogadas.

Quando acabar o jogo todos os dispositivos que estão em jogo saem da aplicação.

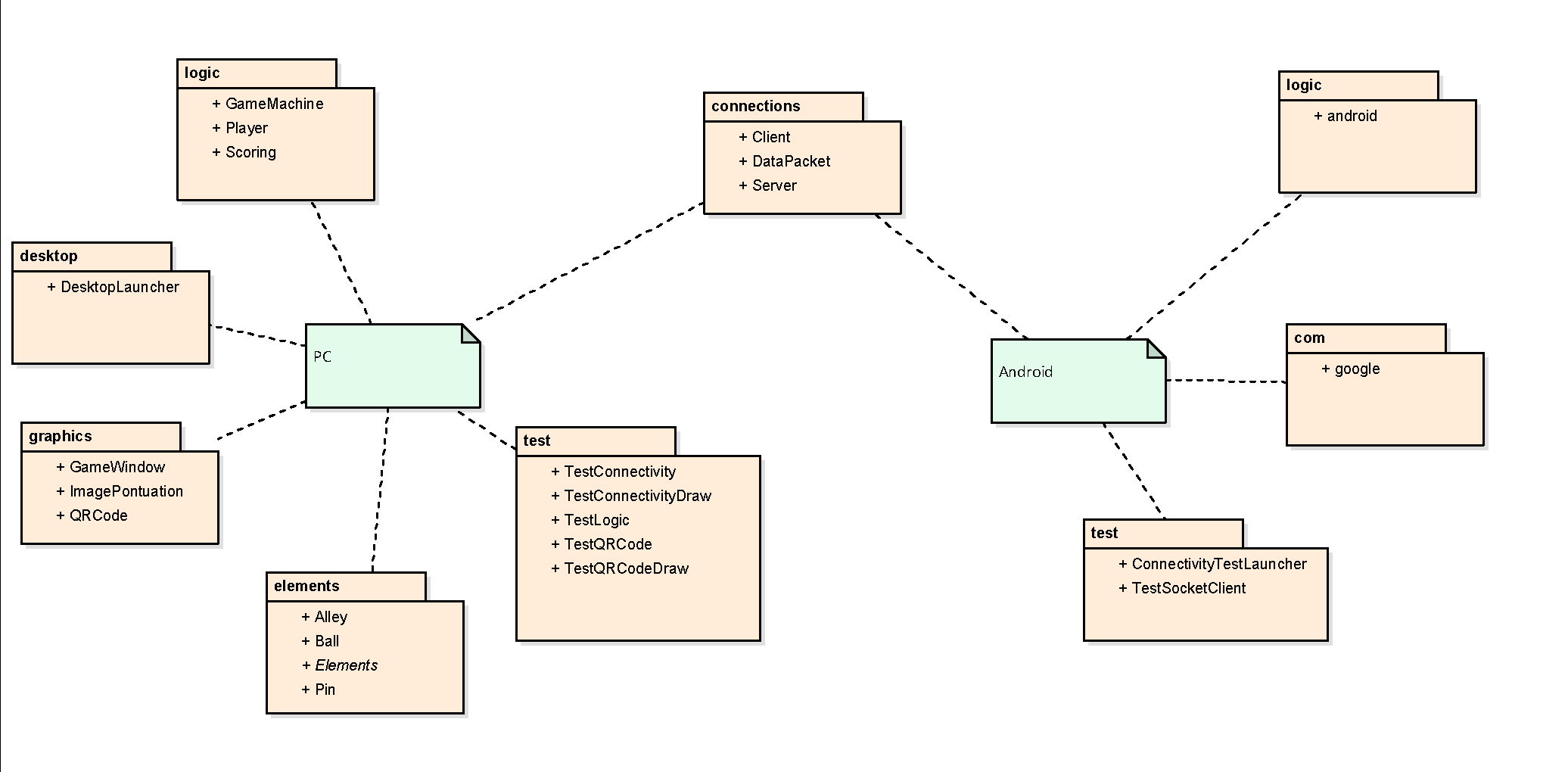
## Formatos de ficheiros

A título de curiosidade para o utilizador, a tabela de pontuações mostrada no final do jogo encontrar-se-á na pasta dos recursos necessários ao jogo com o nome de scoresheet.png.

Figura 15 - Exemplo de tabela de pontuações

# Conceção e implementação

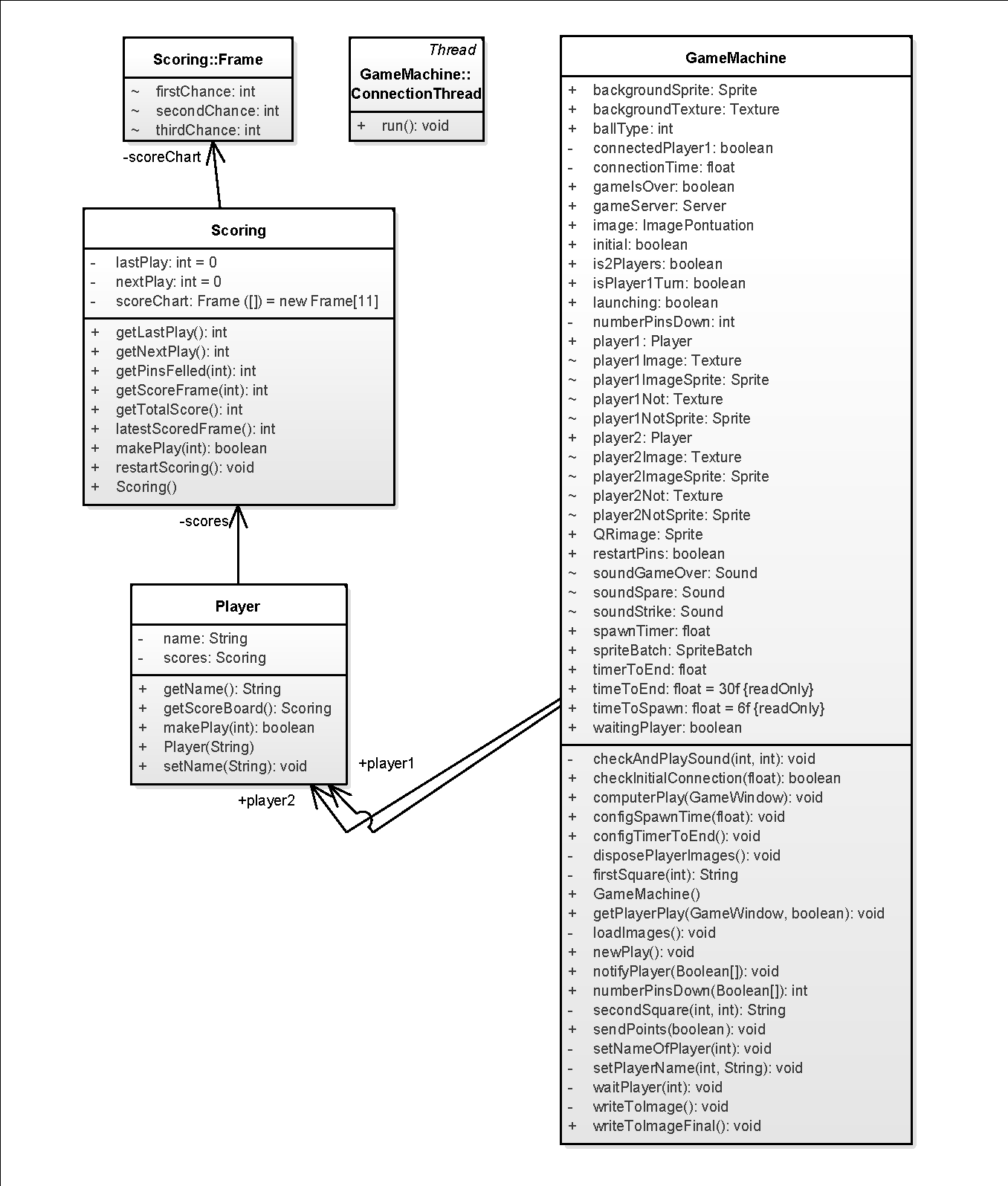
## Estrutura de packages



|  |  |
| --- | --- |
| Logic PC | Este pacote controla a lógica de jogo no lado do computador. Armazena a informação dos jogadores e a calcula as pontuações. |
| Desktop | Inicia o jogo e configura a janela de jogo. |
| Graphics | Mostra o ambiente 3D do jogo incluindo a tabela de pontuações e o QRcode. |
| Elements | Abre os diferentes elementos de jogo e armazena a sua informação. |
| Test PC | Testa as diferentes funções de jogo para garantir que tudo funciona da maneira pretendida. |
| Connections | Armazena o processamento da conectividade entre o android e o computador. |
| Logic Android | Armazena a lógica de jogo do lado do android. |
| Com | Armazena métodos para o lançamento de intents/atividades que utilizam o reader do QRcode e retornam valores dessas atividades para a lógica do jogo normal. |
| Test Android | Testa as funções de conectividade por parte do android. |

## Estrutura de classes

O diagrama de classes por ser demasiado extenso pode ser consultado no ficheiro Bowling.eap. Assim, fica aqui apenas o pacote que consideramos ser mais relevante: o da lógica do pc.



|  |  |
| --- | --- |
| Frame | Container os pinos tombados em jogados. |
| Scoring | Classe para lidar com o tratamento do número de pinos e da pontuação relevantes a jogadas |
| Player | Contém a informação do jogador (é uma abstração do jogador) |
| Thread | Liga a Game Machine aos jogadores |
| GameMachine | Armazena a lógica de jogo e envia e recebe a informação dos jogadores |

## Padrões de desenho utilizados

Pensamos em aplicar o padrão de um Singleton às classes relacionadas às conexões entre android e computador (pois só é necessário um objeto cliente por jogador e um servidor por jogo), mas uma vez que encontramos avisos sobre sistemas android usarem o coletor de lixo para efetuarem dispose dos dados guardados por singletons e reiniciarem a class, decidimos evitar a aplicação desse padrão nesta instância.

Ainda relevante ao tema da recolha de lixo, em imensos pontos foi necessário implementar a capacidade da Game Machine identificar objetos no espaço 3D (principalmente os pinos que são objetos com física) e deixar os ambiente runtime libertar os seus recursos quando deixam de ser necessários, o que é difícil de identificar sem ação propositada e dinâmica da máquina, e causaria um aumento enorme do espaço e tempo utilizados no processo do programa. Existem portanto incorporados nos seus métodos comportamentos de Dispose Handling.

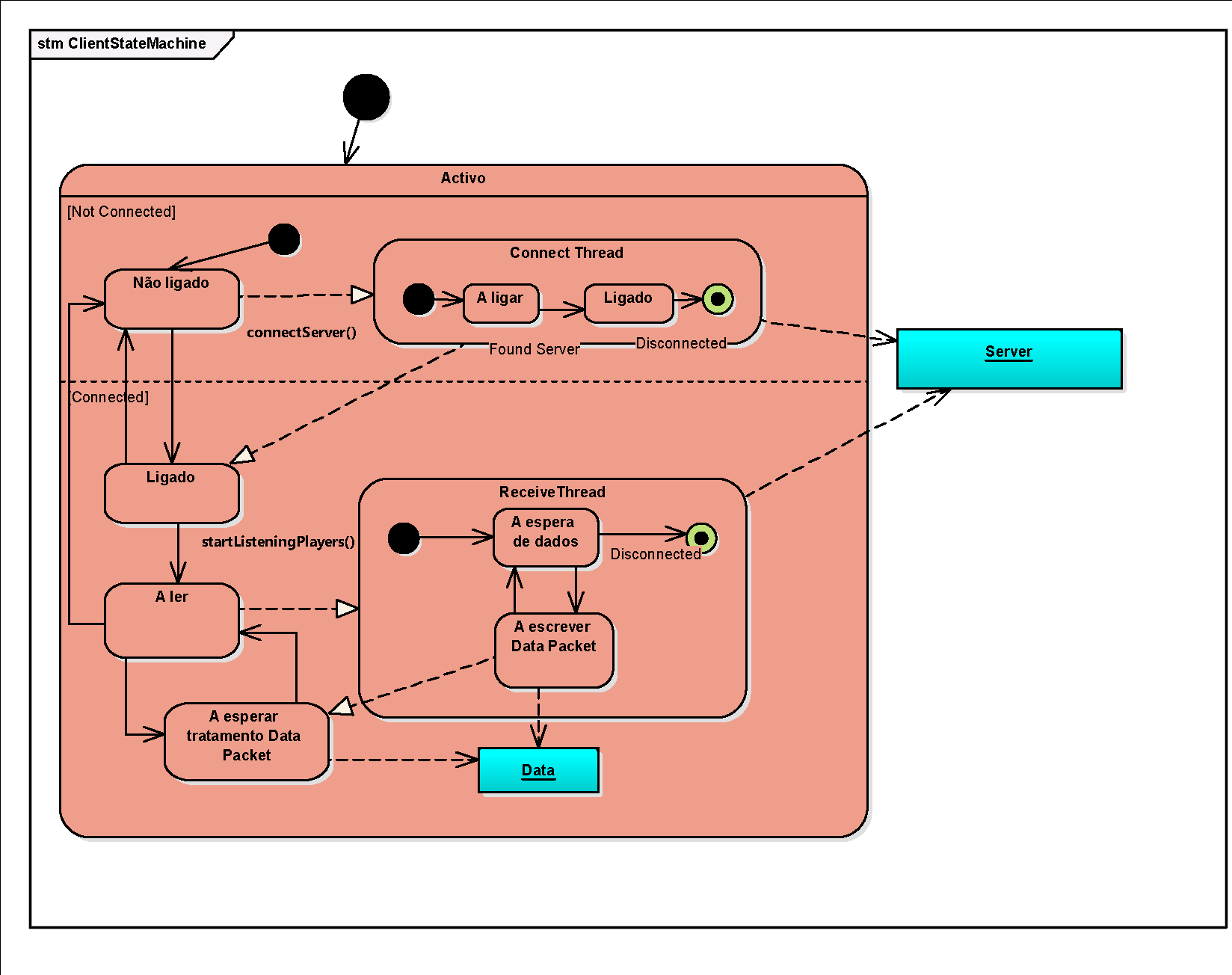
Durante a execução do programa existem imensas ações necessárias ao cálculo da pontuação de um jogador a ser tratadas pela Game Machine, principalmente no caso de ser necessário fazer uma nova jogada (o que implica diversas alterações nas pontuações de diferente 'Frames' de bowling. No entanto, a Game Machine não contém uma única tabela de pontuação. Esta é incluída dentro de cada objeto Jogador que por sua vez serve de Proxy nas ordens da Game Machine às tabelas de pontuação.

O padrão mais relevante a mencionar é o design orientado a Máquinas de Estado, a Game Machine, os objetos de conectividade e a actividade do jogo Android. Por exemplo no caso da classe AndroidBowlingGame é possível verificar a existência de 10 métodos correspondentes exatamente à transição entre estados com nomes apropriados à sua natureza, e consequentemente, responsáveis à alteração do comportamento do jogo (ativa inputs diferentes e desenha imagens diferentes no ecrã).

## Mecanismos importantes

No jogo desenvolvido são utilizados diversos mecanismos, mas para se ser breve apenas se vai falar das mais importantes. Assim, importa referir a conectividade no cliente.

É possível demonstrar com este exemplo um mecanismo comum que se encontra no trabalho: A utilização de threads que efetuam uma tarefa sincronamente com um objeto Cliente que as envolve. As Threads, quando acabam a sua tarefa (ou efetuam uma de múltiplas tarefas) levam à alteração do estado e comportamento do objeto envolvente.



No exemplo: O Cliente, antes de ligar, cria a thread que procura uma ligação a um servidor, permitindo-se, o Cliente, efetuar outras funções entretanto. Após haver uma ligação a um servidor, a thread que a fez termina e o Cliente passa para o estado “Ligado”, onde já não consegue lançar a thread de ligação outra vez, entre outros comportamentos alterados. Já a thread de receber dados do servidor não termina logo após uma só tarefa, mas sim, desliga quando deixar de haver ligação, pelo que não existirá (exceto em estado de terminação) quando Cliente retornar a “Não Ligado”. O Cliente, tal como antes, está livre de efetuar outras tarefas.

Este mecanismo encontra-se vastamente expandido nas classes AndroidBowlingGame e GameMachine.

## Bibliotecas, tecnologias e ferramentas utilizadas

Música/Sons:

* Sons dos pinos: <https://www.freesound.org/>
* Voz do strike e spare: Modificação da voz do João Estrada up201303988 (FEUP | MIEIC | 2º Ano)
* Música de jogo: Mr. Belt & Wezol - Homeless (Original Mix)
* Música final: The Doors - The End

Modelos 3D:

* Sala de bowling: <http://www.turbosquid.com/3d-models/free-bowling-3d-model/433462>
* Bola e pino: <http://www.turbosquid.com/3d-models/bowling-set-obj-free/520062>

Imagens e restantes ferramentas utilizadas:

* Ícone do jogo: <http://www.iconarchive.com/>

Software de desenvolvimento:

* Blender: <https://www.blender.org/>
* Cinema 4D: <http://www.maxon.net/products/cinema-4d-studio>
* Eclipse: <https://eclipse.org/>

Bibliotecas:

* libGDX: <http://libgdx.badlogicgames.com/>
* Zxing: <https://github.com/zxing/zxing>

## Dificuldades encontradas e sua resolução

Ao longo do projeto foram encontradas várias dificuldades. Ao nível do android, estas dificuldades estão relacionadas com as diferentes resoluções dos dispositivos e os diferentes “aspect ratio”. Já em relação ao computador surgiram algumas dificuldades em relação à modelação 3D, pois alguns dos modelos gerados tinham demasiados vértices que não eram suportados pelo libGDX. Deste modo, foram utilizados modelos 3D livres de direitos de autor.

Para testar as funções da lógica, foram feitas alguns cenários de teste. Contudo, como o jogo é em 3D, a lógica inerente a este aspeto não pode ser testada.

# Conclusões

Logo na altura da escolha do tempo traçamos alguns objetivos que tinham de ser cumpridos por nós mesmos. Estes objetivos consistiam em fazer um jogo atrativo, acessível e divertido, adotar novas métodos, ferramentas e tecnologias com que ainda não tínhamos trabalhado, ajudar-nos mutuamente e divertirmo-nos a fazer isto tudo. Felizmente, podemos afirmar com certeza que todas as metas foram alcançadas.

Contudo, apesar de estarmos satisfeitos com o nosso jogo, existem limitações que escapam ao nosso controlo como o facto do libGDX não funcionar de modo coerente em todos os computadores. Por outras palavras, uma das limitações conhecidas do libGDX é que em ambientes 3D, o java runtime nem sempre é fluído. Para não nos prolongar sobre esta limitação, está aqui uma explicação mais detalhada por parte do utilizador toverna no dia 6 de janeiro de 2014 (<http://badlogicgames.com/forum/viewtopic.php?t=12085&p=54743>). Resumindo, num computador o jogo pode ser fluído, tanto como noutro a performance deteriorar-se. Antes de passar a outro tópico, são feitos dispose mal os objetos saem da cena (um pino quando cai por exemplo), logo esta não é uma das razões para a perca de fluidez (esta questão também está esclarecida no tópico).

Finalmente, cada elemento do grupo trabalhou equitativamente, pois fazíamos o jogo em conjunto, mas uma possível divisão seria em que o António Ramadas esteve mais concentrado no desktop (ambiente 3D e jogo) e o Pedro Carvalho no android, a lógica de jogo e a sua conectividade. O relatório foi feito em modo cooperativo.

FIM