
**Mestrado Integrado em Engenharia Informática e Computação**

**Laboratório de Programação Orientada a Objetos**

# TETRIS

**Relatório Final de Projeto**


| | |
|---|---|
| Luís Melo | 201206020 |
| Teresa Conceição | 201202874 |


06-06-2016

# ÍNDICE

# ÍNDICE DE FIGURAS



# ÍNDICE DE TABELAS



# INDICE DE ACRÓNIMOS

UML - Unified Modeling Language

MVC - Model-view-controller.

API - Application Programming Interface.

IDE - Integrated Development Environment

SDK - Software development kit

XML - eXtensible Markup Language

# INTRODUÇÃO

Este trabalho prático vem integrado na Unidade Curricular de Laboratório de Programação Orientada a Objetos do Mestrado Integrado em Engenharia Informática e Computação, tendo como principal objetivo o desenvolvimento de uma aplicação em Java para dispositivos móveis Android.

A aplicação escolhida foi o jogo do Tetris e tendo em conta que seria a primeira experiência para ambos da criação em Android a principal meta definida foi a implementação de uma aplicação simples, eficaz e fluída que cumprisse o seu propósito.

Serve o presente relatório para documentar a implementação do jogo, estando divido nas seguintes partes:

1. Manual de Utilização, onde se descreve o jogo e as funcionalidades do mesmo

2. Conceção, implementação e teste. Numa primeira parte apresentam-se vários diagramas que exemplificam a estrutura do programa. Depois referem-se as várias ferramentas, bibliotecas e tecnologias usadas bem como os *Design Patterns* aos quais se recorreu. Por fim demonstram-se os testes implementados ao software.

3. Problemas Encontrados, onde se descrevem os principais problemas na implementação do projeto e se foram ou não ultrapassados

4. Conclusão, na qual se comenta os resultados obtidos e possíveis melhoramentos futuros.

# MANUAL DE UTILIZAÇÃO

## O que é o Tetris?

O Tetris é um jogo do tipo puzzle criado em 1984 por 3 russos. É um jogo muito popular e que ao longo dos anos foi sendo alterado e lançado para as mais diversas plataformas. Existem 7 tipos de peças diferentes cada uma constituída por 4 blocos. Essas peças caem do topo da board, uma de cada vez e quando tocam no "chão" ou em outras peças param, acumulando-se. O objectivo é, portanto, evitar que a pilha amontoada de blocos chegue até ao topo do campo de jogo. Para isso, é necessário ir completando linhas horizontais de blocos para que as mesmas vão desaparecendo.

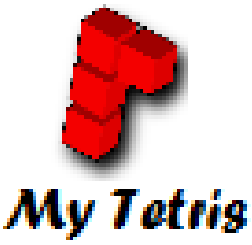


*Figura 1 Icon da Aplicação*

## Sistema de Pontos

O sistema de pontos é bastante simples. O utilizador recebe 1, 2, 4 ou 8 pontos consoante se faça 1,2,3 ou 4 linhas na mesma jogada, respetivamente. Por cada 16 pontos completados, o nivel sobe e consequentemente a velocidade das peças a cair também é incrementada

## Comandos

Tratando-se de um jogo em Android, os comandos tendem a ser intuitivos. *Swipe* para a esquerda ou para a direita move a peça nessa direção, *swipe* para cima ou para baixo, faz com que a peça caia mais um pouco mais rápido. Para rodar a peça, basta tocar no ecrã.

## Configurações

De modo a deixar o jogo simples, apenas foram implementadas configurações que permitam mudar o tema, entre modo diurno (fundo claro) e modo noturno (fundo escuro) e a possibilidade de ter som ou não (estado *mute*). Estas configurações podem ser encontradas clicando na roda dentada no menu principal e são guardadas num ficheiro xml no próprio dispositivo para que se mantenham as mesmas numa futura utilização. É nesse ficheiro xml que é também guardado o high score para o dispositivo em uso.

## Como Jogar?

Para jogar, é necessário descarregar a aplicação enviada conjuntamente com o relatório para um dispositivo móvel e clicar na mesma para iniciar o jogo. Em alternativa foi feita também uma versão Desktop para correr no computador para a qual basta também correr o executável.

## Modo de Utilização

A imagem aqui apresentada representa as possíveis navegações entre os vários ecrãs do jogo (Inicial, Menu Principal, Settings, Ecrã de Jogo, Ecrã de GameOver)

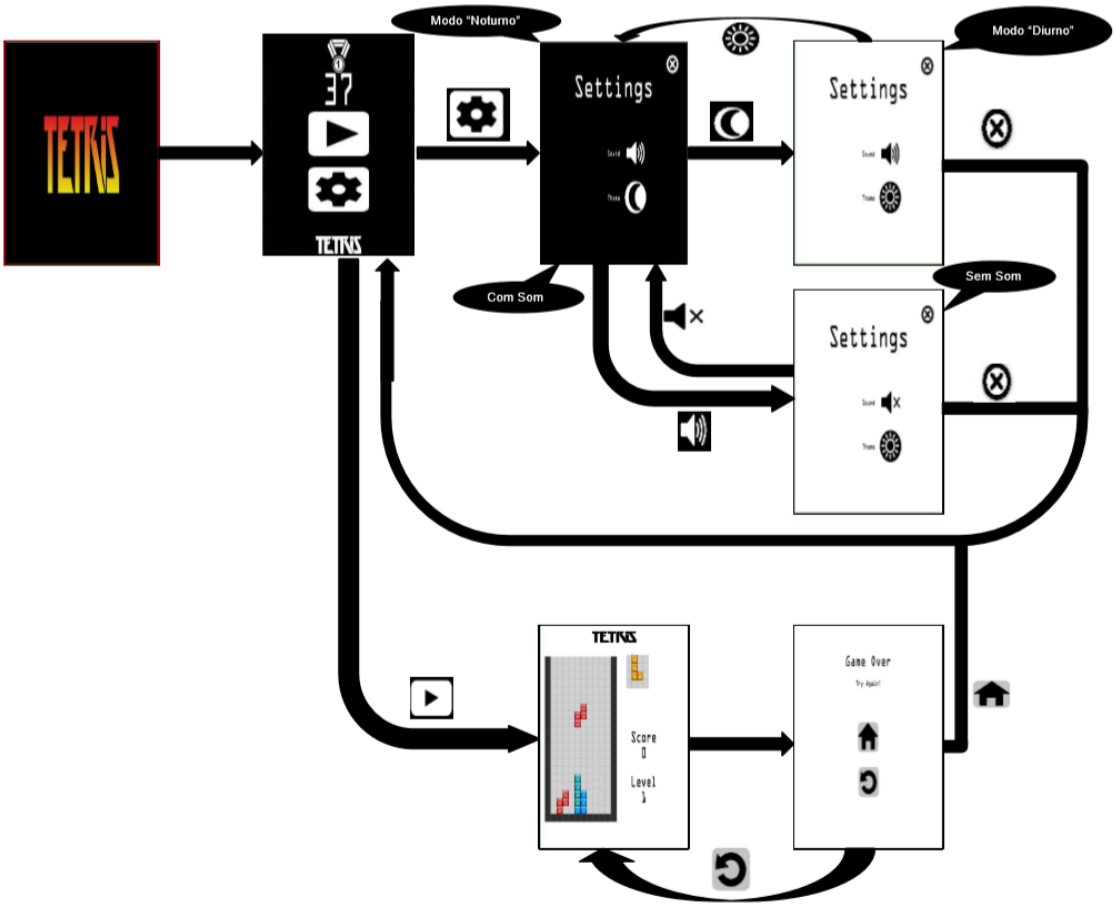


*Figura 2 Navegabilidade entre ecrãs*

O primeiro ecrã representa a introdução do jogo ao qual segue a apresentação do ecrã com o menu principal. No mesmo clicando na seta de play começará um novo jogo. Caso o utilizador escolha a roda dentada apresentará um ecrã de configurações onde se poderá definir o tema (modo noturno escuro ou modo diurno claro) ou a existência ou não de som no jogo. Clicando no botão de saída a aplicação voltará ao menu anterior.

No caso do utilizador querer jogar será levado para o ecrã de jogo onde é apresentado o campo, a pontuação e o nível atual. Após a derrota o jogador será direcionado para o ecrã de Game Over onde terá a possibilidade ir para o menu principal ou de voltar a jogar. Ainda neste ecrã, caso o utilizador faça a sua pontuação máxima aparecerá uma pequena animação idincativa.

# CONCEÇÃO, IMPLEMENTAÇÃO E TESTE

## ESTRUTA DE *PACKAGES*

Decidiu-se dividir o código do programa em vários packages consoante a função de cada um deles de forma a facilitar a implementação, navegação e usabilidade do mesmo.

Os packages implementados encontram-se descritos na seguinte tabela:

| PACKAGE | DESCRIÇÃO |
|---|---|
| **Android** | Cria a interface para android do jogo |
| **Desktop** | Cria a interface para desktop do jogo |
| **Tetris** | Contém a classe principal que implementa o jogo |
| **Logic** | Contém o estado e a lógica do jogo |
| **Helpers** | Conjunto de classes que servem como auxilio à implementação do jogo desde recursos, manipulação de eventos de entrada, render e constantes de jogo. |
| **Accessors** | Classes usadas para implentação de efeitos de imagens no jogo |
| **Screens** | Conjunto dos vários tipos de janelas possíveis durante o jogo |
| **Tests** | Classes de testes Junit implementados |

*Tabela 1 Descrição dos vários package do programa*

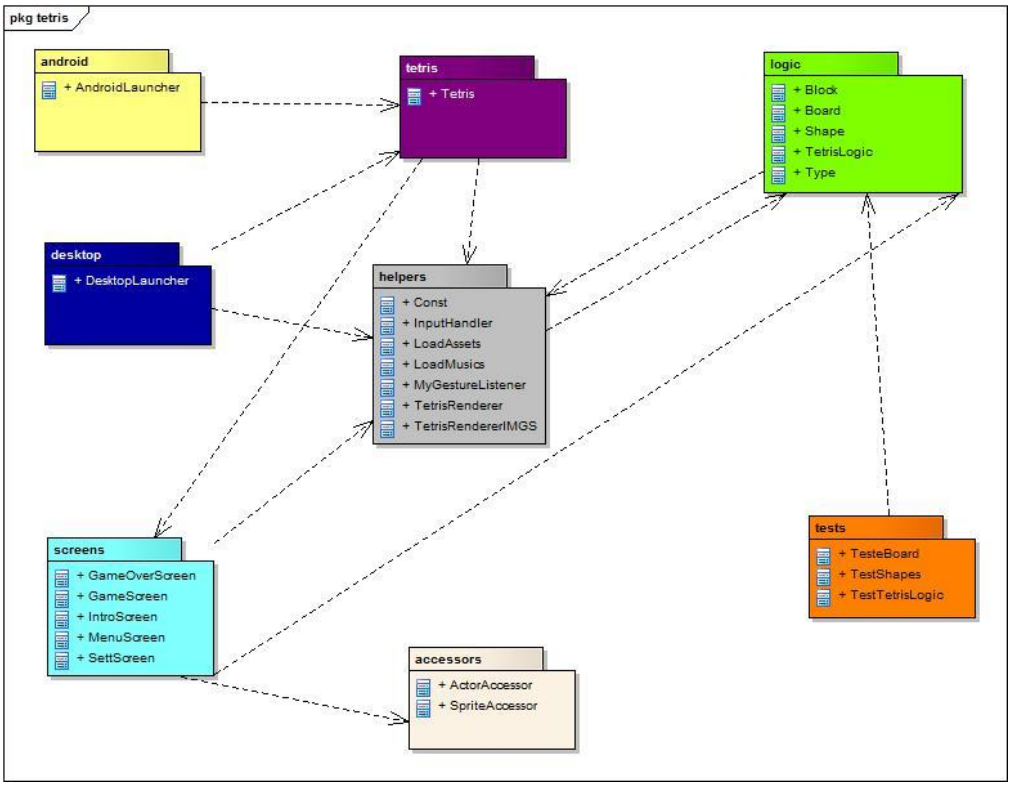


*Figura 3 Diagrama de Packages e respetivas dependências entre os mesmos*

Como pode ser visto pelo driagrama UML[1] de *Packages* quase todos se relacionam e fazem uso uns dos outros apesar das diferentes funcionalidades.

## ESTRUTURA DE CLASSES

Em cada package foram implementadas várias classes cujos diagramas e respetivas funções se encontram a seguir descritos, cada um com direito a uma tabela explicativa e um diagrama de classes com as principais associações "inter e intra *package".*

### Package android

| CLASSE | DESCRIÇÃO |
|---|---|
| **AndroidLaucher** | Inicializa a aplicação em Android instanciando também um objeto da classe Tetris do package tetris |

*Tabela 2 Descrição das classes do package android*

[1] Todos os diagramas UML aqui apresentados estão incluidos no ficheiro do modelo criado no Enterprise Architect

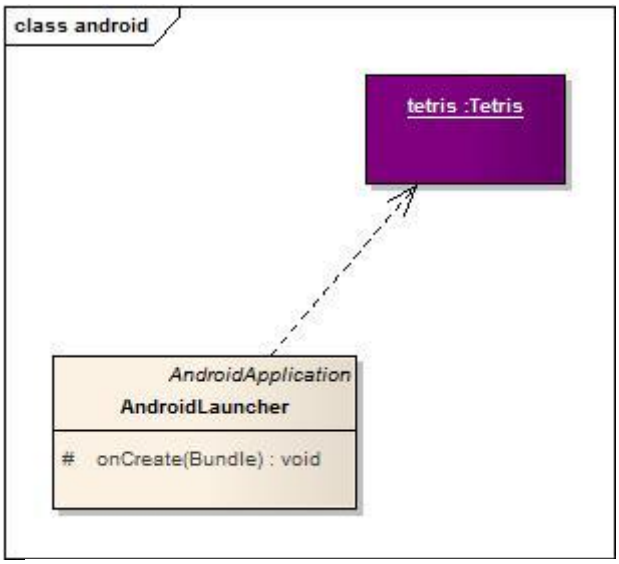


*Figura 4 Diagrama de Classes package android*

## Package desktop

| CLASSE | DESCRIÇÃO |
|---|---|
| **DesktopLaucher** | Inicializa a aplicação Desktop instanciando também um objeto da classe Tetris do package tetris |

*Tabela 3 Descrição das classes do package desktop*

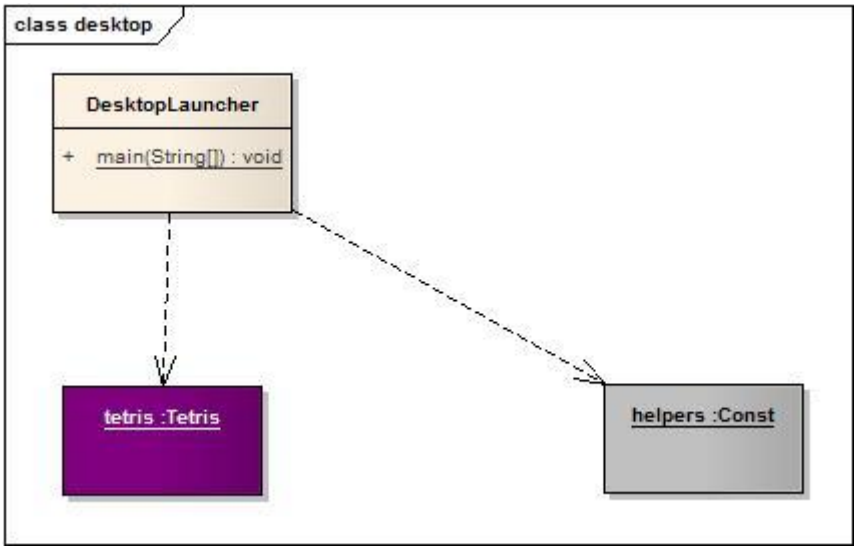


*Figura 5 Diagrama de Classes package desktop*

### Package Tetris

| CLASSE | DESCRIÇÃO |
|---|---|
| **Tetris** | Classe principal que representa o jogo (implementa a a interface Game do libgdx) |

*Tabela 4 Descrição das classes do package Tetris*

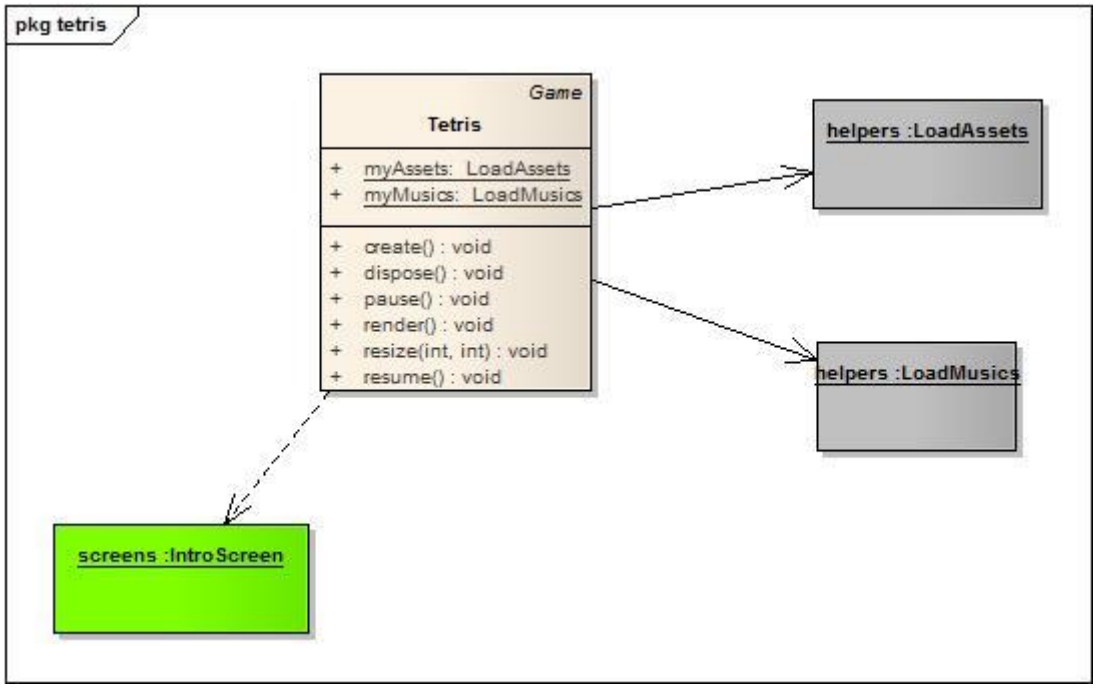


*Figura 6 Diagrama de classes package Tetris*

### Package Logic

| CLASSE | DESCRIÇÃO |
|---|---|
| **TetrisLogic** | Classe principal do jogo |
| **Board** | Campo de jogo que composto por vários *Blocks* e *Shapes* (quando criadas) |
| **Shape** | Figura (que pode ser de vários tipos) constituida por uma matriz de *Blocks* |
| **Block** | Blocos (elemento unitário de uma Board) |
| **Type** | Enumeração de vários tipos de peças |

*Tabela 5 Descrição das classes do package Logic*

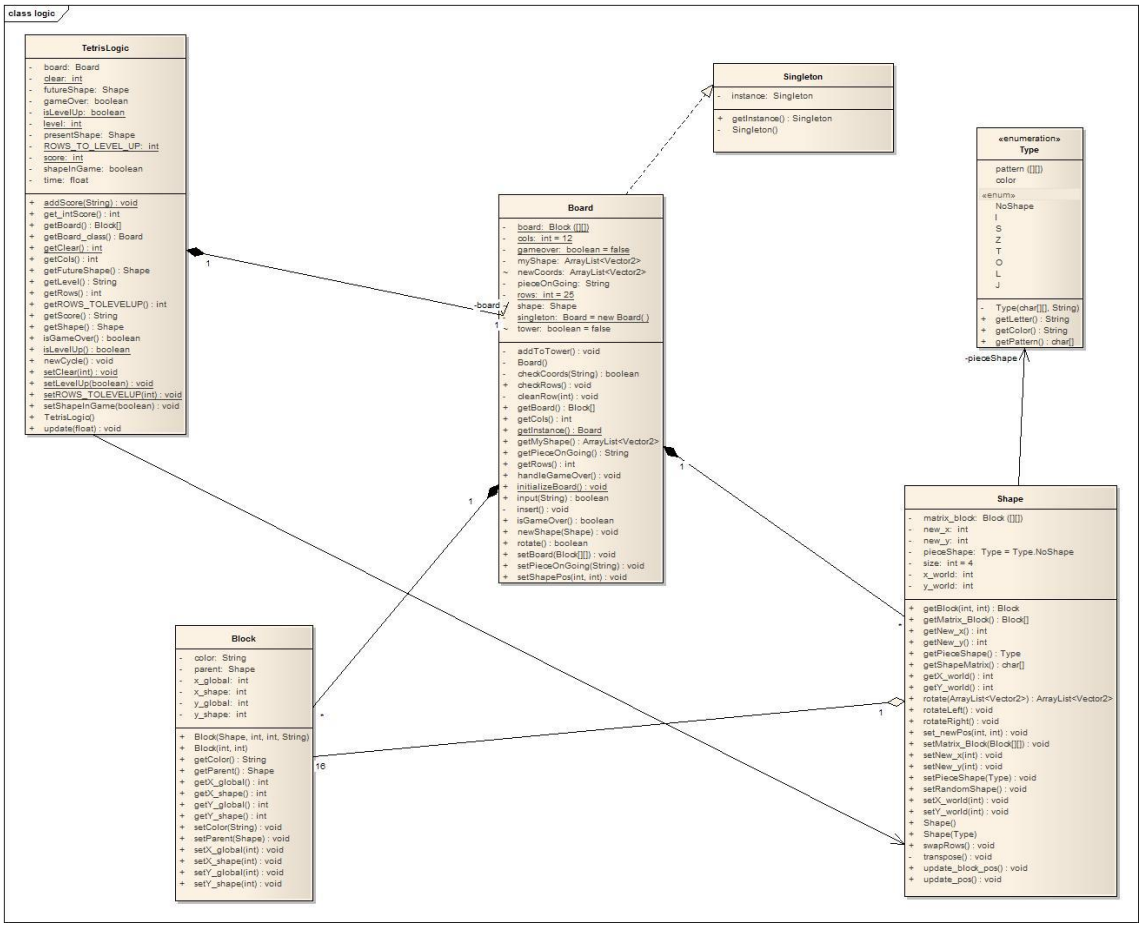


*Figura 7 Diagrama de Classes do package Logic*

Devido a ser o *package* que implementa a lógica por de trás do jogo, no geral as suas classes não contém referências nem dependências a classes de outros *packages* (tirando uma referência ao ciclo de jogo atual dado pela classe Const do *package* helpers).

## Package Screens

| CLASSE | DESCRIÇÃO |
|---|---|
| **IntroScreen** | Ecrã inicial com imagem do jogo |
| **MenuScreen** | Ecrã com o menu principal |
| **SettScreen** | Ecrã com as definições do jogo |
| **GameScreen** | Ecrã com o jogo a decorrer |
| **GameOverScreen** | Ecrã quando o jogo acaba |

*Tabela 6 Descrição das classes do package Screens*

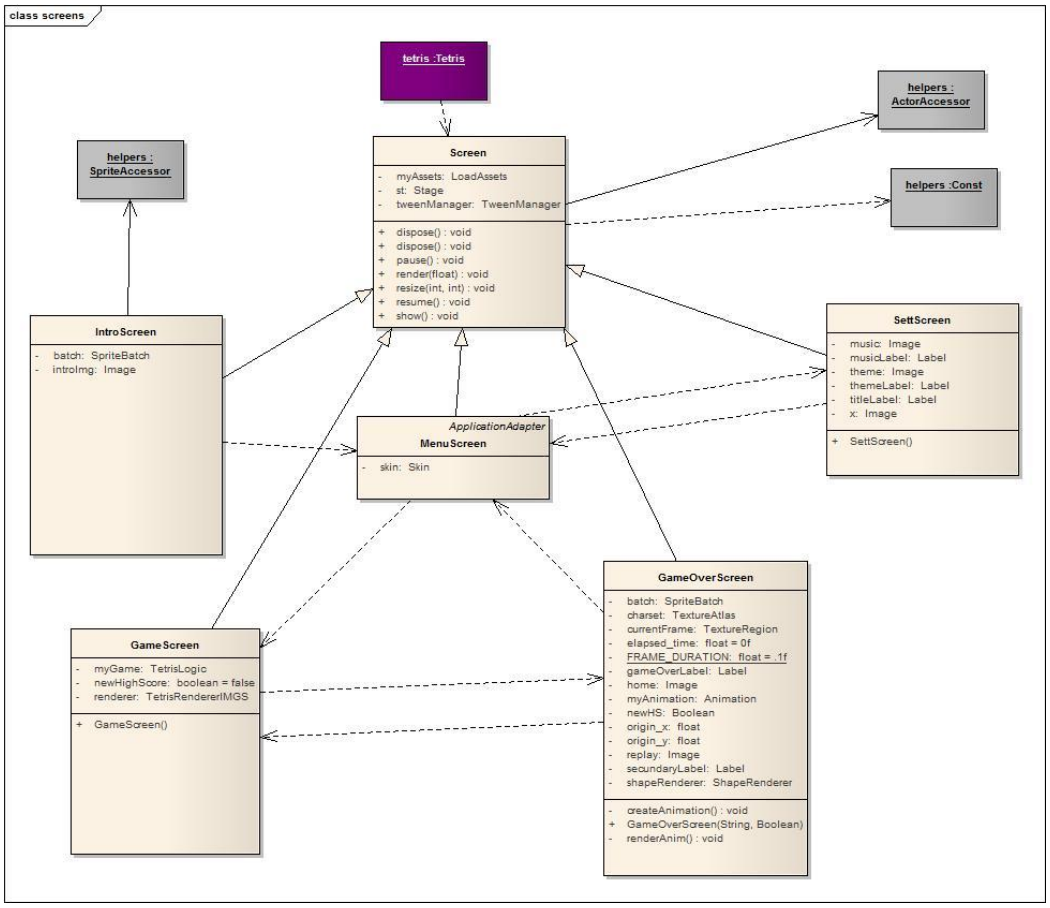


*Figura 8 Diagrama de Classes do package Screens*

## Package Helpers

| CLASSE | DESCRIÇÃO |
|---|---|
| **Const** | Classe com várias constantes inerentes ao jogo e suas definições |
| **LoadAssets** | Lida com o acesso aos vários fcheiros de recursos do sistema |
| **LoadMusics** | Lida com todos os sons do jogo |
| **InputHandler** | Processa os eventos de input na versão desktop |
| **MyGestureListener** | Processa os eventos de input na versão Android |
| **TetrisRenderIMGS** | Processa o render desenhando e atualizando os vários gráficos do jogo |

*Tabela 7 Descrição das classes do package Helpers*

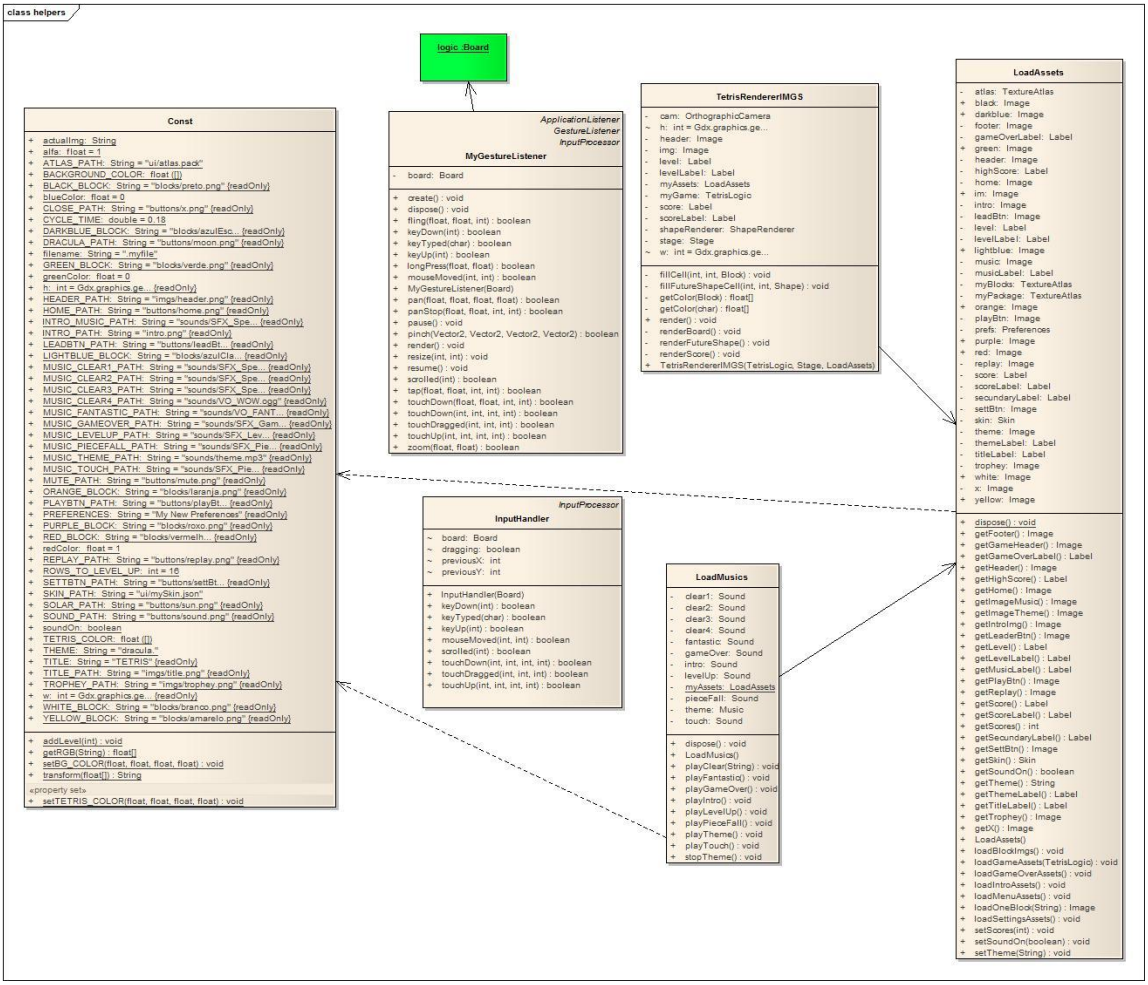


*Figura 9 Diagrama de Classes package Helpers*

## Package Accessors

| CLASSE | DESCRIÇÃO |
|---|---|
| **ActorAccessor** | Representação e dados de vários elementos gráficos, como labels, imagem |
| **SpriteAccessor** | Representação e dados de uma imagem |

*Tabela 8 Descrição das classes do package Accessors*

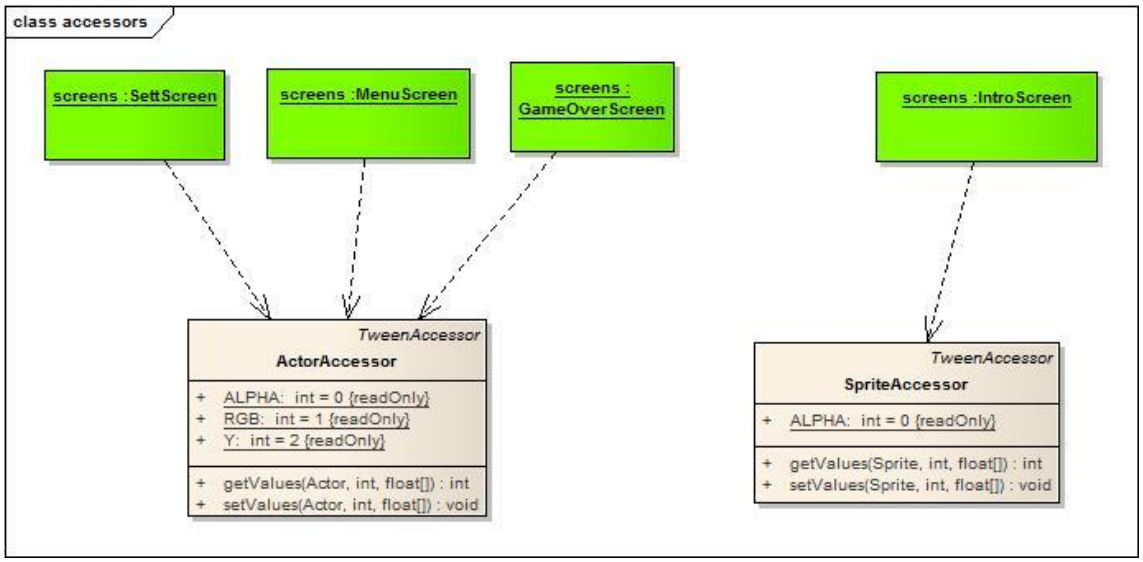


*Figura 10 Diagrama de Classes package screens*

## Package Tests

| CLASSE | DESCRIÇÃO |
|---|---|
| **TestBoard** | Testes unitários à classe Board |
| **TestShapes** | Testes unitários à classe Shape |
| **TestTetrisLogic** | Testes unitários à classe TetrisLogic |

*Tabela 9 Descrição das classes do package Tests*

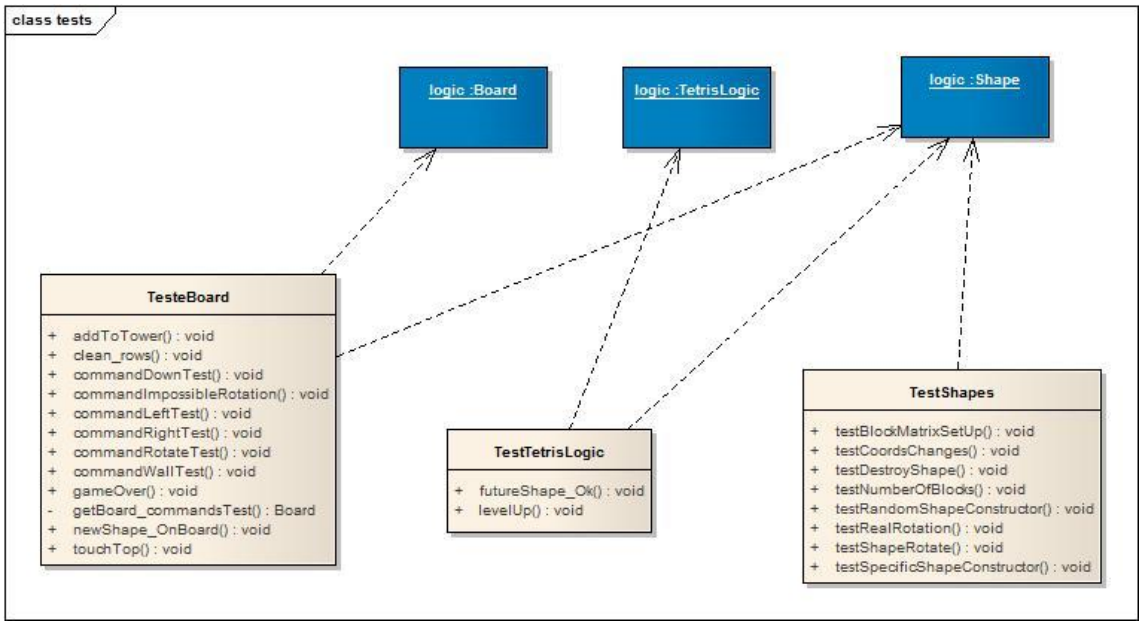


*Figura 11 Diagrama de Classes package Tests*

## PADRÕES DE DESENHO

Os Padrões de Desenho podems ser bastante úteis no desenvolvimento de software. Neste projeto foram usados alguns deles:

- ✓ Singleton, usado para a criação de uma e só uma instância da classe Board visto que o Tetris é jogado apenas com um tabuleiro, mantendo para isso um ponto global de acesso à mesma.
- ✓ Command, presente na class MyGestureListener com vista à manipulação de eventos que surgem quando o utilizador invoca algum comando como por exemplo swipe para mover a peça ou touch para rodar a peça. É um padrão bastante comum no desenvolvimento de jogos.
- ✓ Game Loop. Este padrão é já implementado pela framework libgdx. Cada classe que implemente a interface Screen tem que implementar o método render, sendo o mesmoexecutado em ciclo. O programador tem acesso ao tempo que esse ciclo demorou a ser executado através do argumento delta.

Para além disso, como já foi referido para facilitar a implementação e posteriores alterações, dividiu-se a estrutura do programa em várias partes de forma a diferenciar a lógica do jogo (Model), o densenvolvimento gráfico e respetiva representação (View) e a interface e ligação com o utilizador (Controller), daí ter sido utilizado também o padrão MVC.

Depois de implementado o código percebeu-se que se podia ter feito uso de mais padrões como o Composite. Este teria sido útil na criação das Shapes compostas por blocos, uma vez que a sua principal característica é a possibilidade de tratar as várias instâncias *child* (blocos) do respetivo *parent* (shape) como um todo, escusando para isso de estar a alterar bloco a bloco como foi feito neste programa quando requerido (rotações, movimentações, etc).

## MECANISMO E ESTADOS DO JOGO

De forma simples e geral o mecanismo a aplicação pode ser descrito pelo seguinte diagrama de estados:

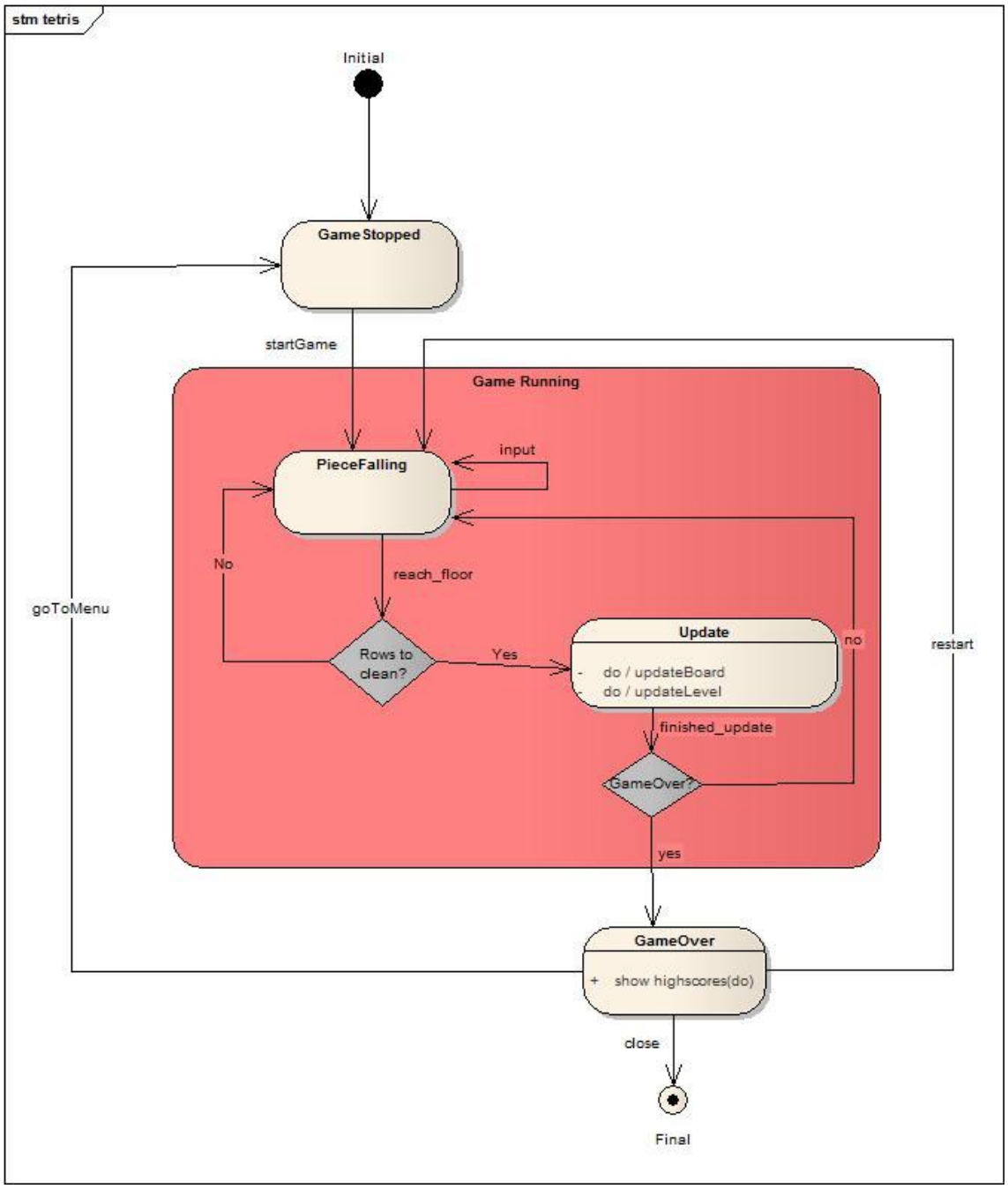


*Figura 12 Diagrama de estados geral do ponto de vista do utilizador*

Assim os principais estados serão o estado inicial, quando o jogo está parado e o utilizador se apresenta no Menu Principal(não foi descrito no diagrama mas poderá neste estado alterar definições do jogo). Depois um macro estado onde está a jogar o jogo, dentro do qual percorre sempre a mesma sequência. E um estado final, quando perde o jogo, onde é notificado caso tenha atingido o seu *highscore* e consequentemente escrito no ficheiro *xml* do aplicativo. Aqui poderá reeiniciar uma partida, voltar ao menu ou fechar a aplicação.

## FERRAMENTAS, BIBLIOTECAS E TECNOLOGIAS

Para facilitar a implementação do programa foram utilizadas várias ferramentas, entre elas:

Eclipse Mars como IDE com diversas aplicações e tecnologias instaladas como por exemplo EclEmma para cobertura dos testes, Pitclipse para correr testes de mutações dos testes unitários e o Graddle para poder ser desenvolvido para Android. Para além disso a base de todo o código fez-se recorrendo naturalmente ao sdk do Android (API 16) e à biblioteca do Junit4.

Biblioteca libGDX que contém vários classes utilitárias para aplicações android. Fez-se uso de várias classes desta biblioteca como Screens, Game, Scenes, Graphics, InputProcessor, entre outras.

Biblioteca Tween Engine com vista a usar efeitos, nomeadamente fade-in e fade-out presente no primeiro screen.

Por último, fez-se uso do git e github como repositório e plataforma de partilha de código entre os dois elementos do grupo que serviu de auxílio durante toda a fase de desenvolvimento.

## TESTES

Para testar as funcionalidades do software e de modo a obter um jogo mais consistente e com menos *bugs* foram executados uma série de testes unitários recorrendo à ferramenta já conhecida Junit.

Para isso, criaram-se 3 classes que testam a classe principal da lógica do jogo, as peças e o tabuleiro.

Uma vez que ao testar as peças e a board do jogo a classe Block era também já testada não se implementou uma classe para testes da mesma.

Foram então testadas as seguintes funcionalidades:

- ✓ Construção de uma peça com forma aleatória
- ✓ Construção de uma peça com forma especificada
- ✓ Alteração de coordenadas de uma peça
- ✓ Rotação de uma peça para ambos os lados
- ✓ Composição da peça pelo número de blocos certo
- ✓ Introdução de uma dada peça especificada numa Board
- ✓ Comandos de vários tipos para mover peças
- ✓ Tentativa de mover peças para além dos limites
- ✓ Imposibilidade de rotação de uma peça numa dada Board
- ✓ Eliminação de uma linha completa

- ✓ Peça encontra outra peça
- ✓ Passagem de nível
- ✓ Teste se a peça apresentada como futura é realmente a que vem a seguir
- ✓ Game Over quando chega ao topo da Board

Apenas se achou necessário testar o package Logic e definiu-se inicialmente como objetivo uma cobertura de testes na ordem dos 90%. Obteu-se uma cobertura de 95,3% usando o plugin EclEma.

Quanto aos testes PIT, obteu-se um *Line Coverage* de 92% e um *Mutation Coverage* de 81%.

Para além destes testes foram também realizados testes manuais do ponto de vista da utilização da aplicação.

## DIFICULDADES ENCONTRADAS

As principais dificuldades durante o projeto prenderam-se na adaptação ao ambiente de desenvolvimento de aplicações Android que era ainda um pouco desconhecido e como tal teve que haver uma fase prévia de ambientação. Ao longo de toda a implementação houve alguns problemas na definição dos comandos para android uma vez que era desejado comandos que originassem uma aplicação fluida e a precisão dos toques no ecrã não era muito elevada. Após alguma pesquisa estes problemas foram resolvidos conseguindo um conjunto de inputs eficaz.

Para além disso, outro dos problemas encontrados foi o facto de o *render* de uma peça em específico (peça com forma estritamente vertical quando rodada na horizontal) parecer ter um pouco de atraso na aplicação em Androird. Este problema não foi ultrapassado pois não se conseguiu perceber exatamente a sua origem uma vez que apenas acontecia com esta peça em especifico e que na versão desktop o erro não estava presente. Apesar disso julga-se que possa ter a ver com o tempo de atualização da aplicação.

# Conclusão

A realização deste trabalho revelou-se bastante útil para aplicar os conceitos adquiridos ao longo do semestre na unidade curricular de forma mais prática.

Inicialmente o grupo decidiu implementar um jogo simples e funcional, tendo as funcionalidades requeridas sido todas alcançadas.

Apesar disso há alguns aspetos que podiam (e podem) ser ainda melhorados, nomeadamente a fluidez do jogo na versão Android e a implementação de um sistema multi-jogador. Este ultimo ponto não foi feito desde início uma vez que na versão original o Tetris não é se quer um jogo que o permita e como tal decidiu-se apenas traçar essa meta caso houvesse tempo (que acabou por não haver). Para além disso a parte gráfica pode ainda ser melhorada inserindo por exemplo mais animações ao longo do jogo e permitindo ter mais tipos de configurações.

Relativamente à contribuição de cada elemento ao longo do projeto a mesma foi repartida. Apesar de ambos terem trabalhado nas várias partes do projeto a distribuição principal foi feita essencialmente da seguinte forma:

Luís Melo : implementação gráfica da aplicação, passagem para Android e testes unitários

Teresa Conceição: implementação lógica do jogo e testes unitários

# REFERÊNCIAS


Android. (s.d.). Obtido de Android Developers: https://developer.android.com/develop/index.html?hl=id

Aurelien Ribon. (2012). *Package aurelienribon.tweenengine*. Obtido de Aurelien Ribon's dev Blog: http://www.aurelienribon.com/universal-tween-engine/javadoc/aurelienribon/tweenengine/package-summary.html

TutorialsPoint. (2016). *UML Tutorial*. Obtido de http://www.tutorialspoint.com/uml/

Wikia. (s.d.). *Tetromino*. Obtido de http://tetris.wikia.com/wiki/Tetromino

Zechne, M. (2013). Obtido de libGDX: https://libgdx.badlogicgames.com/documentation.html

Slides e exemplos disponibilizados pelos docentes