|  |
| --- |
| Instituto superior de engenharia de coimbra |
| Programação Orientada a Objectos |
| Warcraft |
|  |
| **Luís Jordão e André Oliveira** |
| **21201026 | 21200800** |
| **11/01/2015** |

|  |
| --- |
|  |

índice

## Introdução

Os objectivos deste trabalho é demostrar a utilização de objectos na linguagem de programação C++, a utilização de classes, bem como outras funcionalidades que foram aprendidas ao longo do semestre.

Assim, esta aplicação permite ao utilizador jogar um jogo que faz lembrar vagamente a versão do Warcraft.

Este jogo simula a luta entre populações inimigas pelo domínio de território e de recursos.

## Considerações e opções tomadas

A mecânica implementada neste jogo é muito simples. Inicialmente, é desenhada a interface ao utilizador para este estar ocorrente do que está a decorrer, e é-lhe fornecida uma área onde é possível introduzir comandos.

Depois de introduzido um comando, e extraída a sua informação necessária, a função atribui uma ordem á unidade alvo que é identificada pelo seu id, a unidade regista a tarefa que tem de executar, mas é quando se dá uma iteração que a unidade a começa a executar.

Á excepção dos comandos set\*, que são executados “na hora”.

## considerações técnicas

Para guardar informação, a classe Jogo, que é instanciada na função main, guarda uma matriz do tipo Terreno denominada de Mapa. Então, é possível dizer que cada Terreno é uma quadrícula de mapa e é este objecto que é maioritariamente responsável por cada recurso no mapa, são ainda aqui guardadas unidades e edifício (se existir), referente a esta quadricula de mapa, no entanto, é a população referente a responsável por estes elementos.

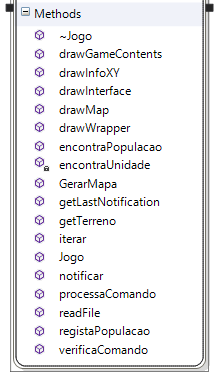
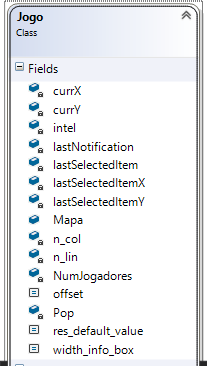
Isto porque, na criação de um elemento, o terreno e a população apontam para o mesmo elemento, deste modo, o processo para obter informação relativamente a cada item da população torna-se bastante fácil e acessível.

Na remoção de elementos, o terreno efectua um erase, mas é a população (que é responsável pelos seus elementos), que efectua o delete.

## Manual do Programador

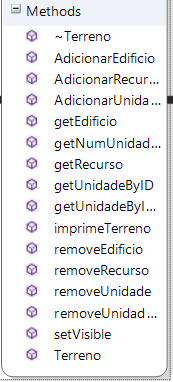
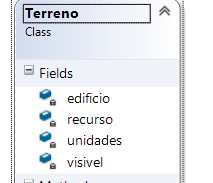
### Classes utilizadas e diagrama de classes

#### Class Jogo



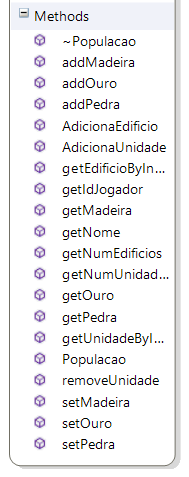
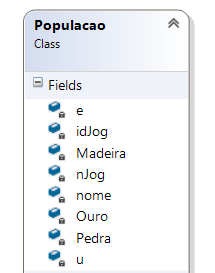
A classe jogo é a classe responsável pela gestão de quase todos os elementos do simulador, o método verificaComando é responsável pela validação de sintaxe dos comandos inseridos pelo utilizador e o método processaComando como o próprio nome indica, trata de processar os comandos para serem executados a valida informação como por exemplo o número de recursos necessários.

#### Classe Terreno



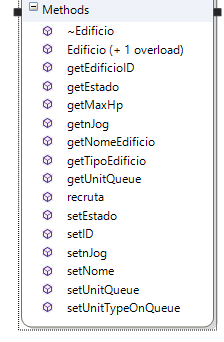
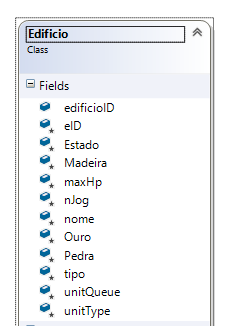
A classe Terreno representa cada quadricula no mapa. Guarda um recurso ou um edifício, um vector de unidades e se está visível ao utilizador ou não.

#### Classe População



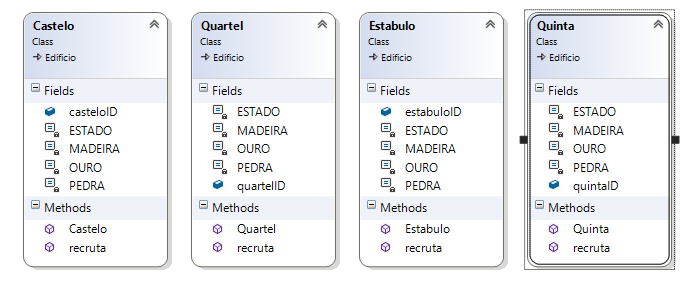
A classe população é a maior responsável pela gestão de unidades e edifícios do simulador, assim como a informação relativa a cada população como por exemplo os recursos de cada uma.

#### Classe Edificio



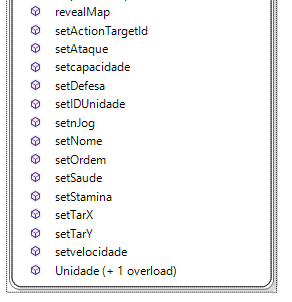
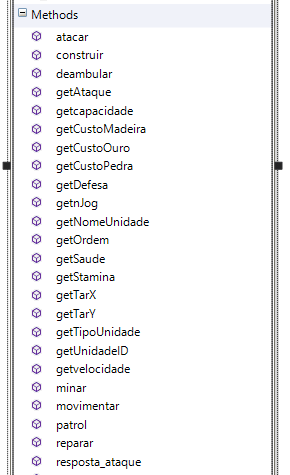
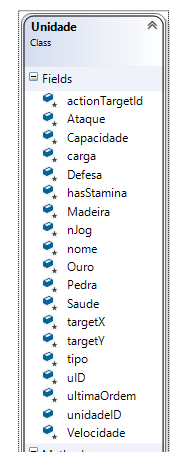
A classe edifício é responsável pela gestão da informação de todos os edifícios do simulador, guarda informação como os custos de cada estrutura, o seu estado e estado máximo, a propriedade unitQueue permite saber se este edifício está a recrutar unidades.

#### Classe edificio – Classes derivadas / Sub-classes



O método recruta, puro virtual, é o responsável pelo recrutamento de unidades por cada estrutura, que herda a classe edifício.

#### Classe Unidade



A classe unidade, é responsável pela gestão de todas as unidades no simulador, sublinho a propriedade hasStamina, que é uma propriedade que serve para bloquear as unidades para evitar que estas efectuem mais que uma iteração por cada turno.

#### Classe Unidade – Sub-classes / Classes derivadas

