ISEC - I See, Eu Conquisto!

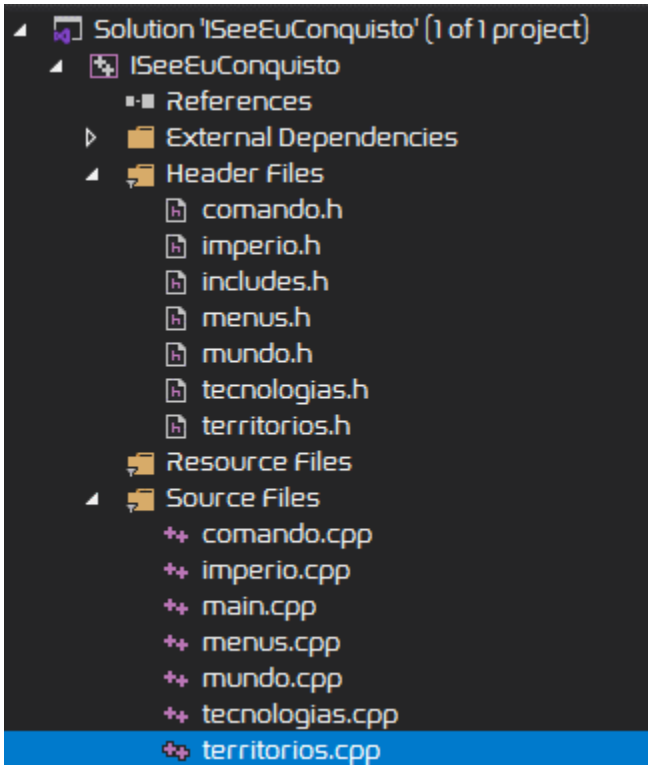
Programação Orientada a Objetos – Ano Letivo 2020/2021

Por João Oliveira e José Marques.

Índice:

Organização do Código

1. Introdução



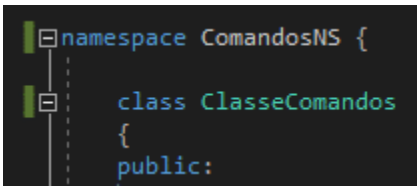
Após várias mudanças no programa em geral, estes são os ficheiros finais que o nosso programa necessita para funcionar por completo.

Quero dizer já que o programa foi feito no standard C++14 para não haver conflito com os conteúdos lecionados nas aulas.

Vou entrar mais em detalhe nas classes criadas até agora.

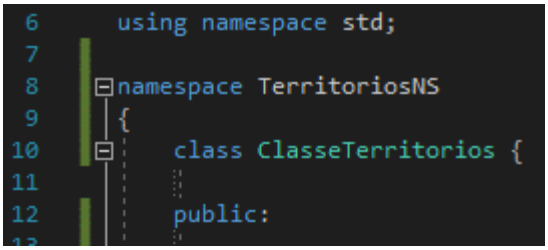
2. Classes implementadas

A primeira classe de que vou falar é a Classe *Comandos:*



A classe "*comandos*" é a mais importante do programa até agora, pois é ela segura as funções principais do programa, é a responsável por chamar a criação de terrenos, manipular os ficheiros – tanto para gravar como para abrir e retornar o progresso do jogador, e de listar o progresso do jogador.

Outra classe que temos de igual importância é a classe *Territórios:*



Onde todos os territórios que criamos usam as variáveis que os definem. Esta classe é crucial na criação de tipos de territórios, sendo a origem para classes derivadas para uma próxima fase do projeto.

3. Classes/conceitos encontradas no enunciado

As classes que identificámos foram:

- Territórios
- Comandos
- Tecnologias
- Eventos

Os conceitos que identificámos foram:

- *Comandos*
- *Territórios*
- *Território Inicial*
- *Castelo*
- *Duna*
- *Fortaleza*
- *Império*
- *Mina*
- *Montanha*
- *Planície*
- *Tecnologias*

4. Classes onde são criados/destruídos objetos

Territórios: os objetos desta classe são criados e destruídos na classe Comandos.

Comandos: os objetos desta classe são criados e destruídos na classe Comandos.

5. Classes com responsabilidades e encapsulamentos

A responsabilidade "criar territórios" está atribuída à classe **ClasseComandos** porque esta tem a coleção de comandos.

6. Classes com objetivos focados

Classe Territórios: Tem responsabilidade relativa à criação dos territórios (*construtor*) e dado que pertencem a cada território.

7. Classes com responsabilidades de interface com o utilizador com e sem lógica

Responsabilidade de interface: main.cpp, menus.cpp

Responsabilidade de lógica da aplicação: comandos.cpp, territorios.cpp

8. Primeiro objeto para além da camada de interação com o utilizador

As ordens vindas da camada de interação com o utilizador são recebidas no *comandos.cpp* e processadas por um objeto da **classe Comandos**.

9. A classe que representa a envolvente de toda a lógica, executa em pormenor muitas funcionalidades, e delega noutras classes

A classe **Comandos** representa a envolvente de toda a lógica. Para criar todos os territórios, delega à classe **Territórios** para que esta os crie.

10. Uma funcionalidade que varia conforme o tipo do objecto que a invoca. Indique em que classes e métodos está implementada esta funcionalidade.

O comando Lista na Classe Comandos varia bastante da fase do programa, sendo que age diferentemente se o chamarmos antes do jogo começar, depois, e se o utilizador decidir listar um só território.

11. Principais classes da aplicação

Classe: Territórios

Responsabilidades:

Criação de atributos para os territórios

Criação de objetos Território

Administração dos Territórios criados e conquistados (Mundo, Império)

Colaboração:

Comando

Classe: Comando

Responsabilidades:

Criação e conquista de territórios

Gravar e carregar ficheiros

Listar progresso

Ajudar em funções do jogo