

Programação Orientada a Objetos

ISEC – Instituto Superior de Engenharia de Coimbra

**Trabalho prático – Meta 2**

2021110042 - Jorge Ricardo Marques Duarte

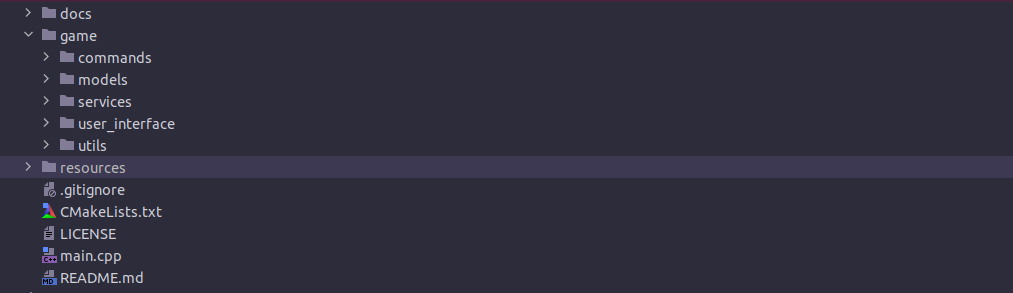
2021146383 - João Alexandre Caravela Marques

**Introdução**

Neste trabalho prático pretende-se aplicar os conhecimentos adquiridos durante as aulas teóricas e práticas da disciplina construindo em C++ um simulador de uma reserva natural povoada por diversos animais.

Este relatório vem com o intuito de demonstrar as diversas decisões tomadas ao longo do desenvolvimento, tanto a nível de estruturas usadas, assim como organização e estrutura da aplicação.

**Estrutura de pastas**

 Na pasta *docs,* temos todos os documentos do projeto, ou seja, todos os objetivos, perguntas e relatórios que foram realizados durante o desenvolvimento.

Na pasta *commands*, temos implementada toda a lógica relacionada a comandos e validações.

Na pasta *docs*, temos documentos e objetivos do projeto, estando estes seperados por metas.

Na pasta *models*,é onde temos todos os modelos utilizados pela aplicação, desde tipos de comida e animais a classes utilizadas como estrutura base para a interface visual.

Na pasta *services* é onde contemos a base da nossa aplicação, sendo no ficheiro GameService.cpp que definimos qual o tamanho da nossa reserva natural, assim como a leitura do ficheiro de constantes e também a inicialização para a receção dos comandos por parte do utilizador.

Na pasta *userInterface*, está contida a lógica de apresentação visual do nosso jogo, isto é, desde a nível de informação sobre os animais, assim como a apresentação da reserva natural com todas as entidades existentes.

Por último, a pasta *utils* apenas contem funções utilitárias que podem ser reutilizadas por outras classes ou funções.

**Classes principais da aplicação**

Como descrito na nossa estruturas de pastas, a nossa pasta principal é a services, contendo toda a lógica principal da aplicação. Nesta pasta contemos a classe GameService contendo classe Game e funções principais privadas para definição de critérios do jogo.

Observe a imagem seguinte referente a classe GameService:



A estrutura Game como podemos visualizar está presente dentro do nosso GameService. Esta estrutura contem todas as informações referentes a reserva natural, como a matriz sendo um vetor de vetores representando a nossa reserva natural, uma lista da classe Animal, outra lista da classe Food, uma classe referente a configurações do jogo e uma lista de estados de jogo a serem guardados em memória.

Observe a imagem seguinte referente a classe Game:



Outra classe importante que contemos na nossa aplicação é a classe MatrixCell que contem os ponteiros de animais ou comidas na respetiva célula de jogo.

O objetivo das listas de ponteiros de animais e comidas é sabermos em que posição de encontra o animal ou comida no tabuleiro. Sendo ponteiro facilita a alteração dos dados do mesmo, sendo que podemos ir diretamente mudar os valores destas entidades.

Isto acaba por trazer uma enorme vantagem a nível de gestão do jogo. Da mesma forma podemos alterar um animal diretamente na lista de animais e sabemos que este será alterado também dentro da célula de jogo.

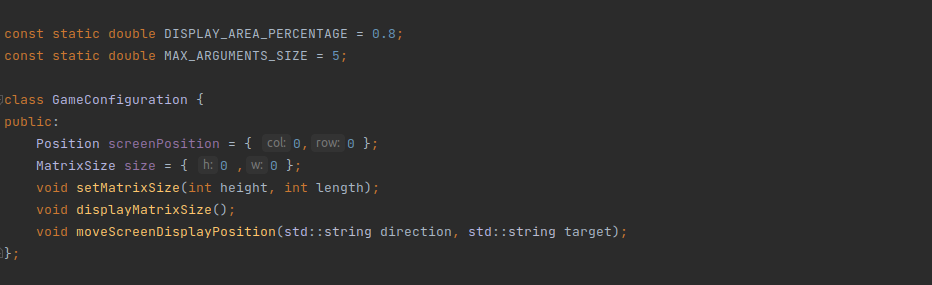
A movimentação dos animais na matriz é feita removendo o ponteiro do animal na célula atual, e adicionando na posição desejada.

Observe a imagem seguinte referente a classe MatrixCell:



Ainda dentro do Game, podemos observar a classe GameConfiguration, sendo a classe responsável por tudo que tenha a ver com configurações do nosso jogo. A sua estrutura contem o tamanho da nossa reserva natural, assim como uma posição de visualização do ecrã. Definimos como limite de área visível 80% do tamanho do jogo quando criado numa variável global constante. Nesta classe é possível modificar a área de jogo visível através da função moveScreenDisplayPosition.

Observe a imagem seguinte referente a classe GameConfiguration:



**falta falar da classe Animal, Comida e do SaveGame**

**Decisões tomadas**

**Animal mistério**

A entidade por definir escolhida como animal mistério foi a raposa. Este animal tem as seguintes características:

Visualiza alimentos do tipo vegetal que neste caso é o alimento mistério amora.

A raposa visualiza animais e comida até 6 posições de distancia em área.

A velocidade da raposa é de 3 posições sempre e em caso de encontrar outro animal, fica com medo e tenta fugir dele.

A cada instância é acrescentada um valor de fome, a não ser que a raposa coma uma amora, redefinindo o valor de fome para zero.

A sua reprodução é uma chance de 40% a cada 15 instâncias de jogo. Se a raposa se conseguir reproduzir, surgira uma nova raposa até 10 posições de distancia.

**Comida mistério**

Para a comida mistério escolhemos a amora, esta comida só é percecionada pelas raposas, sendo a raposa o animal mistério.

O seu valor nutritivo é de 15 valores, não tem toxidade, é identificada pela letra ‘a’, e o seu cheiro é vegetal.

**Bolsa marsupial do Canguru**

**por falar**

**Matriz bidimensional de ponteiros**

**por falar**

**Comando slide**

**por falar**

**Biblioteca ncurses**

**por falar**

**Gestão de comandos**

- Como é feito o handle dos comandos ( base Command.h)

- implementação de cada um extendendo o Command.h

**Testes aplicados durante o desenvolvimento**

**por falar e como fazemos os testes**

**Lista de requisitos implementados**

**Dificuldades e resoluções**

**por falar, por exemplo a visualização da área que o animal vê tendo em conta que o jogo é um “torus”.**