INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

**INSTITUTO POLITÉCNICO DE COIMBRA**

**Licenciatura em Engenharia Informática 2º Ano – 1º Semestre 2021/2022**

**Natural Reserve Simulator**

**Rafael Couto** **Nº 2019142454**

**Rafaela Carvalho Nº 2019127935**

**COIMBRA**

7 de janeiro de 2023

Índice

[Introdução 4](#_Toc124052100)

[Implementação 5](#_Toc124052101)

[Estruturas de dados 5](#_Toc124052102)

[ Vetor de Células 5](#_Toc124052103)

[ Vetor de Reservas 5](#_Toc124052104)

[ Vetor de Animais 5](#_Toc124052105)

[Classes 5](#_Toc124052106)

[Comandos 8](#_Toc124052107)

[ animal 8](#_Toc124052108)

[ kill / killid 8](#_Toc124052109)

[ food 8](#_Toc124052110)

[ feed / feedid 8](#_Toc124052111)

[ nofood 8](#_Toc124052112)

[ empty 8](#_Toc124052113)

[ see 8](#_Toc124052114)

[ n 8](#_Toc124052115)

[ anim 8](#_Toc124052116)

[ visanim 8](#_Toc124052117)

[ store 8](#_Toc124052118)

[ restore 8](#_Toc124052119)

[ load 8](#_Toc124052120)

[ slide 9](#_Toc124052121)

[ clear 9](#_Toc124052122)

[ exit 9](#_Toc124052123)

[Funções auxiliares 9](#_Toc124052124)

[Interface 9](#_Toc124052125)

[Decisões tomadas 10](#_Toc124052126)

[Conclusão 10](#_Toc124052127)

[Anexos 10](#_Toc124052128)

# Introdução

A elaboração deste trabalho prático visa consolidar conhecimentos em linguagem C++, explorados nas aulas teóricas e práticas, criando capacidade de desenvolvimento de aplicações nesta linguagem de programação.

Pretende-se criar um simulador, denominado *Natural Reserve Simulator*, destinado a gestão e desenvolvimento de uma reserva natural. Ao jogador será atribuída uma reserva que o mesmo deve gerir, provando-a e alimentando todos os seres vivos.

Deste modo, será implementada uma classe geral *Reserva* responsável por suportar todo o tipo de dados relativos a cada *célula* e à sua pormenorização.

Ora, cada *Reserva* tem *nLinhas* por *nColunas* e cada unidade será uma *cell* que é descrita por *animais* e *alimento*.

Graphical user interface, text

Description automatically generatedA interação com o jogo processa-se através de comando e ação por parte do jogador.

# Implementação

No jogo *Natural Reserve Simulator* existem várias maneiras de interação, desde colocar alimentos aleatoriamente ou especificamente, bem como *animais*.

Permite fazer toda a gestão de mapa e a cada instante vão-se desempenhando ações derivadas das características de cada *animal* e *alimento*

## Estruturas de dados

### Vetor de Células

A picture containing graphical user interface

Description automatically generatedA estrutura de dados escolhida para armazenar os objetos da classe *Célula* foi um *vetor* bidimensional. Este vetor pertence à classe *Reserva* e para ser criado foi utilizado um vetor dentro de outro vetor para que seja possível aceder aos objetos utilizando a notação de coordenada (vetor[x][y]).

Figura – Vetor bidimensional de células

De seguida foi necessário criar um vetor temporário para armazenar objetos da mesma coluna nesse vetor para posteriormente ser adicionado o vetor temporário ao vetor principal.

Este vetor é criado no construtor da classe *Reserva* quando um objeto deste tipo é criado.

### Vetor de Reservas

Text

Description automatically generatedO vetor de *Reservas* é responsável por guardar os vários estados de jogo existentes à medida que os instantes de jogo vão avançando. São guardadas novas instâncias de *Reservas* sempre que o utilizador executa o comando *store*.

Figura - Vetor de Reservas

### Vetor de Animais

De modo que fosse possível armazenar os animais de uma célula criou-se um vetor de animais para que fosse possível armazenar vários objetos do tipo *Animal* na mesma célula.

Figura - Vetor de Animais

## Classes

Cada classe criada é representativa de um domínio do jogo. Os objetos de classes que contém múltiplas informações sobre esse domínio e funções que nos permitem alterar e ou obter várias informações à cerca de um objeto.

### Reserva

A classe *Reserva* é a classe principal do jogo. Representa um mapa com várias células e também informações de animais e alimentos.

* + Funções:
    - getId()
    - getSimulatedTime()
    - getCellInfo()
    - addHistory()
    - animalActions()
    - foodActions()

### getId()

### getSimulatedTime()

### getCellInfo()

### addHistory()

### animalActions()

### foodActions()

### Célula

A classe *Célula* representa uma coordenada no mapa e pode possuir animais e um alimento.

* + Funções:
    - getInfo()
    - removeAllEntities()
    - getAnimals()
    - getFood()
    - removeAnimal()
    - copyNewAnimal()
    - removeFood()
    - adicionaRelva()

### getInfo()

### removeAllEntities()

### getAnimals()

### getFood()

### removeAnimal()

### copyNewAnimal()

### removeFood()

### adicionaRelva()

### Animais

A classe *Animais* representa um animal que pode estar posicionado numa determinada célula da reserva.

* + Funções:
    - addFoodHistory()
    - feed()
    - getInfo()
    - getMotherID()
    - feed()
    - getBirthInstant()

### addFoodHistory()

### feed()

### getInfo()

### getMotherID()

### feed()

### getBirthInstant()

### Alimentos

A classe *Alimentos* representa um alimento que pode ser posicionado em qualquer célula da Reserva.

* + Funções:
    - getCheiro()
    - operator= (Alimentos \*alimentos)
    - getDuracao()
    - getValorNutritivo()
    - getToxicidade()

### getCheiro()

### operator= (Alimentos \*alimentos)

### getDuracao()

### getValorNutritivo()

### getToxicidade()

## Comandos

Os comandos são utilizados pelo jogador para desempenhar uma série de ações de modo a efetuar operações como criar animais e alimentos.

### animal

### kill / killid

### food

### feed / feedid

### nofood

### empty

### see

### n

### anim

### visanim

### store

### restore

### load

### slide

### clear

### exit

## Funções auxiliares

# Interface

Criou-se uma classe *Interface*, onde é criado e tratado o ambiente do jogo e onde é tratado também da validação dos comandos que vão ser utilizados ao longo do simulador.

Text

Description automatically generated

# Decisões tomadas

Em relação à parte que visível no ecrã tomamos a decisão de mostrar um tamanho fixo, definido na classe *Reserva* com o nome *viewWindow*.

De modo que esta *viewWindow* seja independente de toda a visualização do tabuleiro de jogo foi criado uma variável *topLeftCorner* que permitirá fazer o cálculo de apresentação das células no intervalo desejado.

Tal como descrito no enunciado o número de linhas e colunas do tabuleiro de jogo é pedido ao utilizador em *runtime* no início da execução do programa.

Foi decidido que o jogo iniciaria sempre a sua área visual no canto superior esquerdo (0,0) e daí em diante é controlado através do utilizador.

Tomamos por iniciativa mais uma vez criar um comando clear, comando que irá ser destinado para limpar tudo o que já foi escrito no terminal para que se possa continuar a usar o simulador sem falta de informação por falta de espaço no terminal.

# Conclusão

# Anexos

* Reserva.h
* Reserva.cpp
* Cell.h
* Cell.cpp
* Animais.h
* Animais.cpp
* Alimentos.h
* Alimentos.cpp
* Interface.h
* Interface.cpp
* Utils.h
* main.cpp
* constantes.txt
* commands.txt
* POO – 2223 - Relatório.pdf