Programação Orientada a Objetos 2022/2023

Relatório Final

Bruno Trindade (2019132612)

Índice

Detalhes

Tipos de animais

Coelhos

- Representados pela letra “C”.
- Saúde inicial lida da variável “SCoelho” localizada no ficheiro “Constantes.txt”
- Campo de visão = 4
- Desloca-se 1-2 unidades em cada instante.
- Tem um peso entre 1kg a 4kg (Aleatório quando criado).
- Ticks de vida lida da variável “VCoelho” localizada no ficheiro “Constantes.txt”
- +1 Fome a cada instante. Se tiver mais que 10 fome, passa a perder 1 HP e desloca-se 1-3 unidades. Se tiver mais que 20 fome, passa a perder 2 HP e desloca-se 1-4 unidades.
- Cheira alimentos “verdura”.
- Caso veja um animal com mais do que 10kg foge.
- A cada 8 instantes, faz nascer um coelho com 50% de probabilidade.

Ovelha

- Representados pela letra “O”.
- Saúde inicial lida da variável “SOvelha” localizada no ficheiro “Constantes.txt”
- Campo de visão = 3
- Desloca-se 1 unidade em cada instante.
- Tem um peso entre 4kg a 8kg (Aleatório quando criado).
- Ticks de vida lida da variável “VOvelha” localizada no ficheiro “Constantes.txt”
- +1 Fome a cada instante. Se tiver mais que 15 fome, passa a perder 1 HP e desloca-se 1-2 unidades. Se tiver mais que 20 fome, passa a perder 2 HP.
- Cheira alimentos “erva”.
- Caso veja um animal com mais do que 15kg foge.
- A cada 15 instantes, faz nascer uma ovelha (com o mesmo HP).
- Quando morre, deixa um Alimento do tipo Corpo (Valor nutricional igual ao peso da ovelha e o nível de toxicidade = 0).

Lobo

- Representados pela letra “L”.
- Saúde inicial lida da variável “SLobo” localizada no ficheiro “Constantes.txt”
- Campo de visão = 5
- Desloca-se 1 unidade em cada instante.
- Tem um peso inicial de 15kg.
- Não têm ticks de vida.
- +2 Fome a cada instante. Se tiver mais que 15 fome, passa a perder 1 HP e desloca-se 2 unidades. Se tiver mais que 25 fome, passa a perder 2 HP.
- Cheira alimentos “carne”.
- Caso veja um animal vai atrás dele a 2 unidades/instante (Se tiver mais do que 15 de fome são 3 unidades/instante).
- Caso o animal tenha mais peso que ele aleatoriamente morre um deles. Se tiver menos, mata-o.

- Faz nascer outro lobo uma única vez (entre 5 a variável “VLobo” localizada no ficheiro “Constantes.txt” instantes).
- Quando morre, deixa um Alimento do tipo Corpo (Valor nutricional = 10 e nível de toxicidade = 0).

Canguru

- Representados pela letra “G”.
- Saúde inicial lida da variável “SCanguru” localizada no ficheiro “Constantes.txt”
- Campo de visão = 7
- Desloca-se 1 unidade em cada instante.
- Tem um peso inicial de 10kg. Após 20 instantes, passa a pesar 20kg.
- Ticks de vida lida da variável “VCanguru” localizada no ficheiro “Constantes.txt”
- Não têm fome.
- Faz nascer outro Canguru a cada 30 instantes.
- Quando morre, deixa um Alimento do tipo Corpo (Valor nutricional = 15 e nível de toxicidade = 5).

Animal-Mistério

- Representados pela letra “M”.
- Campo de visão = 1.
- Tem um peso inicial de 20kg.
- HP = 200.
- Cheira alimentos “especial”.
- Desloca-se 1 unidade em cada instante.
- Foge de animais com mais do que 15kg.
- Quando morre, deixa um Alimento do tipo Misterio (com coordenadas aleatórias).
- Faz nascer outro Animal-Mistério a cada 50 instantes.
- A cada instante aumenta 2 fome. Quando tem mais do que 10 fome, passa a perder 5 HP.

Tipos de alimentos

Relva

- Representados pela letra “r”.
- Valor nutritivo = 3.
- Toxicidade = 0.
- Cheira a “erva” e a “verdura”.
- Ticks de vida lida da variável “VRelva” localizada no ficheiro “Constantes.txt”.
- Faz nascer outra relva passado 75% do seu tick de vida.

Cenoura

- Representados pela letra “t”.
- Valor nutritivo = 4.
- Toxicidade = 0. Aumenta 1 ponto a cada 10 instantes até máximo de 3.
- Cheira a “verdura”.
- Não têm ticks de vida (infinito).

Corpo

- Representados pela letra “p”.
- A cada instante diminui o seu valor nutritivo em 1 unidade e aumenta o valor de toxicidade em 1 unidade. O valor de toxicidade para de aumentar quando passar 2x o seu valor nutritivo inicial.
- Cheira a “carne”.
- Não têm ticks de vida (infinito).

Bife

- Representados pela letra “b”.
- Valor nutritivo = 10.
- A cada instante diminui o seu valor nutritivo em 1 unidade até chegar a 0.
- Toxicidade = 2.
- Cheira a “carne” e “ketchup”.
- Ticks de vida = 30.

Alimento-mistério

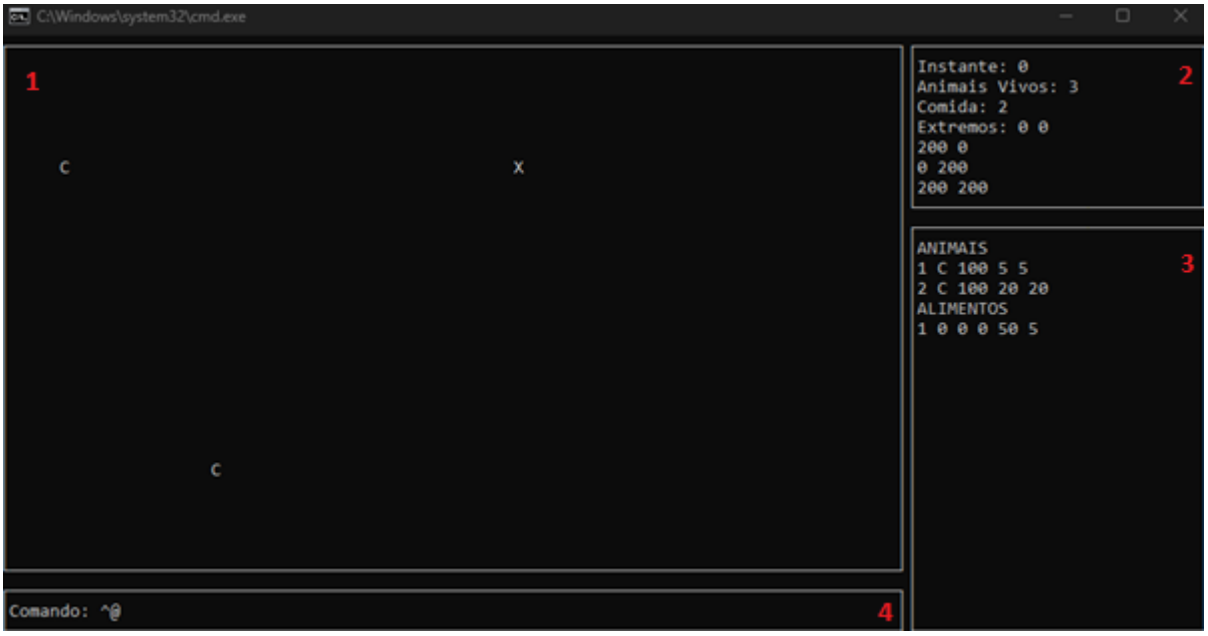
- Representados pela letra “a”.
- Valor nutritivo = 100.
- Toxicidade = 0.
- Cheira a “especial”.
- Ticks de vida = 50.
- A cada instante perde 1 ponto de valor nutritivo e ganha 1 ponto de toxicidade.

Comandos implementados

- Criar animal
  - animal c – cria um animal do tipo “coelho”
  - animal o – cria um animal do tipo “ovelha”
  - animal l – cria um animal do tipo “lobo”
  - animal g – cria um animal do tipo “canguru”
  - animal m – cria um animal do tipo “mistério”
  - animal X <linha> <coluna> - cria um animal do tipo X com coordenadas X Y.
- Matar animal
  - kill <linha> <coluna> - mata animal na posição X Y.
  - kill <id> - mata animal por ID.
- Colocar alimento
  - food r – cria um alimento do tipo "relva”.
  - food t – cria um alimento do tipo “cenoura”.
  - food b – cria um alimento do tipo “bife”.
  - food a – cria um alimento do tipo “mistério”.
  - food X <linha> <coluna> - cria um alimento do tipo X com coordenadas X Y.
- Alimentar diretamente animais
  - feed <linha> <coluna> <valor nutritivo> <toxicidade> - alimenta os animais nas coordenadas X Y com +X valor nutritivo e -X toxicidade.
  - feedid <ID> <valor nutritivo> <toxicidade> - alimenta o animal com esse ID com +X valor nutritivo e -X toxicidade.
- Remover alimento
  - nofood <linha> <coluna> - apaga alimento em X Y.
  - nofood <ID> - apaga alimento com ID.
- Eliminar o que quer que esteja numa posição
  - empty <linha> <coluna> - remove o que quer que esteja na posição X Y.
- Ver o que se encontra numa posição
  - see <linha> <coluna> - indica o que está em X Y.
- Ver informação acerca de um elemento do simulador (animal ou alimento)
  - info <ID> - mostra informação do ID.
- Passar para o instante seguinte da simulação
  - n – avança 1 instante.
  - n <X> - avança X instantes.
  - n <X> <S> - avança X instantes com S segundos entre cada instante.
- Listar ID dos animais na reserva
  - anim – mostra informação dos animais na reserva.
- Listar ID dos animais na área visível da reserva
  - visanim – mostra informação dos animais na área visível da reserva.
- Armazenar o estado da reserva em memória
  - store <nome> – grava o estado da reserva em memória.
- Reativar um estado da reserva previamente armazenado em memória
  - Restore <nome> - recupera o estado da reserva em memória.
- Carregar e executar comandos a partir de um ficheiro de texto
  - load <nome> - lê do ficheiro os comandos.
- Deslocar a área de visualização
  - slide <direcao> <linha> <coluna> - move a área visivel da reserva.

  É possível deslocar a área com as setas.
- Encerrar o simulador
  - Exit.

Aspeto Gráfico



Em 1 temos a reserva principal.
Em 2 temos a informação atual do jogo (o instante, o número de animais vivos, o número de comida existente na reserva e os extremos visíveis da reserva).
Em 3 temos a janela de "output", o feedback dos comandos que o utilizador mande.
Em 4 temos a janela de "input", ou seja, os comandos que o utilizador vai mandar para o programa.

Classes implementadas

Animais

```
class BaseAnimal
{
    int ID;
    int HP;
    int campoVisao;
    int instante;
    double Peso;
    Coordenadas Location;
    std::string Description;
    char especie;
    bool kill;
```

Guarda a informação relativa a todos os animais.

```
class AnimalH: virtual public BaseAnimal
{
    int hunger;
    History **FoodHistory;
    int eatenfood;
```

Guarda a informação relativa aos animais que tenham fome.

```
class AnimalL: virtual public BaseAnimal
{
    int lifeTick;
```

Guarda a informação relativa aos animais que tenham ticks de vida.

```
class CompleteAnimal: public AnimalH, public AnimalL
```

Guarda a informação relativa aos animais que tenham fome e ticks de vida.

```
class Coelho: public CompleteAnimal
```

```
class Ovelha: public CompleteAnimal
```

Coelhos e Ovelha têm ambos ticks de vida e fome.

```
class Lobo: public AnimalH
{
    int VLobo;
    int spawn;
```

```
class Misterio: public AnimalH
```

Os lobos e animais-mistério não têm ticks de vida mas têm fome.

```
class Canguru: public AnimalL
```

Cangurus não têm fome mas têm ticks de vida.

Alimentos

```
class BaseAlimento
{
    int ID;
    int ValorNutritivo;
    int Toxicidade;
    int Instante;
    char letra;
    std::vector<std::string> Cheiro;
    std::string Description;
    Coordenadas Location;
    bool kill;
```

Guarda a informação relativa a todos os alimentos.

```
class AlimentoTV: virtual public BaseAlimento
{
    int TempodeVida;
```

Nem todos os alimentos têm tempo de vida.

```
class Relva: public AlimentoTV
{
    double instanteSpawn;
    bool spawned;
```

```
class Bife: public AlimentoTV
```

```
class AMisterio: public AlimentoTV
```

Relva, bife e Alimento Mistério têm Ticks de vida.

```
class Cenoura: public BaseAlimento
```

```
class Corpo: public BaseAlimento
{
    int ogVN;
```

Cenoura e corpo não têm ticks de vida.

Coordenadas

```
class Coordenadas
{
    int x, y;
```

Esta classe guarda as coordenadas de cada animal/alimento.

Histórico de Comida

```
class History
{
    int valorNutricional;
    int Toxicidade;
    std::string Description;
```

Esta classe guarda informação da comida que cada animal comeu.

Reserva

```
class Reserva
{
    static int ID;
    static int tamanho;
    int instante;
    std::vector<BaseAnimal*> Animais;
    std::vector<BaseAlimento*> Alimentos;
    std::vector<Reserva*> Copias;
    std::vector<std::string> CopiaName;
    std::vector<int> viewarea = {0, 87, 0, 24};
```

O jogo é centrado nesta classe. Esta classe guarda a informação de tudo.