24/25

Instituto Superior de Engenharia
Politécnico de Coimbra

Simulador de Caravanas

TRABALHO PRÁTICO DE PROGRAMAÇÃO ORIENTADA A OBJETOS

FREDERICO ANTÓNIO PACHECO RODRIGUES MAROCO QUELHAS – 2022135081

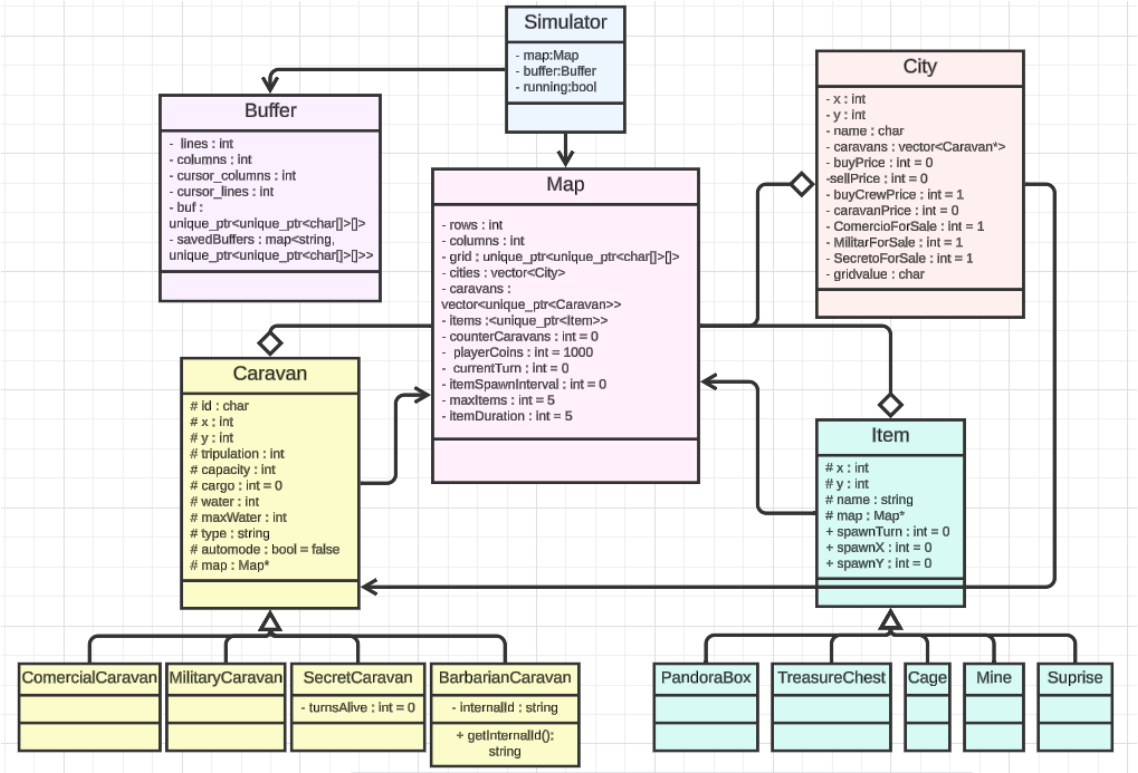
AFONSO DA SILVA – 2021133861

Índice

Introdução

O objetivo deste trabalho prático de Programação Orientada a Objetos (POO) centra-se no desenvolvimento de um simulador de viagens no deserto, implementado em C++. O utilizador controla caravanas que exploram uma grelha retangular representativa do deserto, transportam mercadorias, enfrentam desafios como combates com bárbaros e tempestades de areia, e interagem com cidades para reabastecimento e comércio. O programa utiliza conceitos de programação orientada a objetos e exige uma estrutura modular e eficiente, incluindo a implementação de um buffer em memória para representar o mapa e a lógica associada ao comportamento das caravanas. A interação ocorre exclusivamente através de comandos de texto na consola, seguindo um paradigma de simulação por turnos.

Diagrama de Classes



Este diagrama de classes representa a estrutura de um sistema de simulação, modelando as principais entidades e as suas inter-relações. O sistema é constituído por um *Simulator*, que gere um *Map*, um *Buffer* para a interface do utilizador e várias entidades dinâmicas, nomeadamente cidades (*City*), caravanas (*Caravan*) e itens (*Item*).

O *Simulator* utiliza um *Map* para modelar o ambiente de simulação e um *Buffer* para gerir a apresentação visual. Por sua vez, o *Map* contém cidades, representadas por *City*, caravanas (*Caravan*) e itens (*Item*).

Existe uma relação de agregação entre o *Map* e as *City*, *Caravan* e *Item*, indicando que o *Map* contém estas entidades, mas que as mesmas podem existir de forma independente. As *City* associam-se às *Caravan*, mantendo um conjunto de referências às caravanas existentes dentro da cidade. As *Caravan* e os *Item* também mantêm uma associação com o *Map*, para que estes consigam alterar informações no *Map*.

A hierarquia de herança é utilizada para definir tipos específicos de caravanas e itens. Existem vários tipos de *Caravan* que herdam atributos e métodos da classe base *Caravan*: *ComercialCaravan*, *MilitaryCaravan*, *SecretCaravan* e *BarbarianCaravan*. De forma semelhante, vários tipos de *Item* herdam da classe base *Item*: *PandoraBox*, *TreasureChest*, *Cage*, *Mine* e *Surprise*. Esta estrutura hierárquica promove a reutilização de código e a flexibilidade do sistema.

Funcionalidades Implementadas

- Buffer
  - Esta classe é responsável pela exibição do mapa em consola.
  - Funcionalidades Principais:
    - Criação e gestão de um buffer de caracteres com dimensões dinâmicas (construtor Buffer(int l, int c)).
    - Manipulação de um cursor virtual para posicionamento da escrita (moveCursor(int l, int c)).
    - Escrita de diferentes tipos de dados no buffer (escrever(char c), operator<<(char c), operator<<(const std::string& s), operator<<(int n)).
    - Renderização do buffer para a consola (render()).
    - Sistema para salvar, carregar, listar e apagar cópias do buffer em memória (saveBuffer(const std::string& bufferName), loadBuffer(const std::string& bufferName), listBuffer(), deleteBuffer(const std::string& bufferName)).
- Caravan
  - A classe Caravan e suas derivadas (ComercialCaravan, MilitaryCaravan, SecretCaravan e BarbarianCaravan) representam diferentes tipos de veículos de transporte no deserto. Cada tipo tem atributos (tripulação, capacidade, carga, etc.) e comportamentos específicos.
  - Funcionalidades Principais (Classe Base):
    - Representação de uma caravana com atributos básicos, como identificador, posição, tripulação, capacidade de carga, água e moedas (construtor Caravan(...)).
    - Abstração do comportamento de movimento (move(int newX, int newY)) e comportamento autónomo (autonomousBehavior(bool automatico)).
    - Gestão de recursos como água (refillWater(), reduceWater(int quantity)) e tripulação (reduceTripulation(int n), addTripulation(int n), setTripulation(int n)).
    - Getters e Setters para a manipulação dos atributos das caravanas.
    - Verificação do estado do modo automático (getAutoMode()).
    - Definição do mapa (setMap(Map* map)).
    - Representação em formato de string (getAsString() const).
  - Classes Derivadas (Funcionalidades Específicas):
    - ComercialCaravan: Focada no transporte de carga, procura proteção e itens, com movimento mais lento (move(int newX, int newY), autonomousBehavior(bool automatico)).
    - MilitaryCaravan: Focada no combate, persegue caravanas bárbaras, com movimento mais rápido (move(int newX, int newY), autonomousBehavior(bool automatico)).
    - SecretCaravan: Procura as caravanas do utilizador para lhe dar água e procura cidades para obter água, desaparecendo após algum tempo (move(int newX, int newY), autonomousBehavior(bool automatico)).
    - BarbarianCaravan: Movimento aleatório com perseguição de caravanas do utilizador (move(int newX, int newY), autonomousBehavior(bool automatico)) e obtenção do identificador interno, para identificação das várias caravanas bárbaras (getInternalId() const).

- City
  - Esta classe representa uma localização estática no mapa onde as caravanas podem interagir para obter recursos e realizar trocas.
  - Funcionalidades Principais:
    - Representação de cidades com posição e nome (construtor City(...)).
    - Gestão das caravanas presentes na cidade (listagem e adição/remoção) (addCaravan(Caravan* caravan), removeCaravan(Caravan* caravan), showCaravans()).
    - Reabastecimento de água para as caravanas (refillWater(Caravan* caravan)).
    - Realização de trocas de mercadorias (buyGoods(Caravan* caravan, int tons, Map& map), sellGoods(Caravan* caravan, Map& map)).
    - Contratação de tripulação (buyCrew(Caravan* caravan, int crewMembers, Map& map)).
    - Apresentação das caravanas paradas na cidade (showParkedCaravans() const) e à venda (showForSaleCaravans() const).
    - Venda de caravanas (sellCaravan(char type, Map& map)).
    - Getters e setters da posição onde se encontra a cidade.
    - Apresentação de informação da cidade (showCityInfo() const).
    - Obtenção do valor da cidade no grid (getGridValue() const).
    - Getters e setters para os preços da cidade (getBuyPrice() const, setBuyPrice(int value), getSellPrice() const, setSellPrice(int value), getBuyCrewPrice() const, setBuyCrewPrice(int value), setCaravanPrice(int value)).
- Map
  - A classe Map representa o mundo do jogo, incluindo o terreno, as cidades, as caravanas e os itens. Ela gerencia as interações entre esses elementos e a lógica do jogo.
  - Funcionalidades Principais:
    - Gestão do mapa (grelha de caracteres), dimensões (construtor Map(int rows, int columns)).
    - Carregamento do mapa a partir de um ficheiro externo (loadMap(const string& filename)).
    - Gestão das cidades (getCities() const, addCity(City city)), das caravanas (getCaravans() const, addCaravan(unique_ptr<Caravan> caravan), removeCaravan(char caravanId), listAllCaravans() const) e dos itens (getItems() const, addItem(unique_ptr<Item> item), listAllItems() const).
    - Geração de UUID para as caravanas bárbaras (generateUUID()).
    - Movimento das caravanas, validando os movimentos (moveCaravan(char caravanId, int newX, int newY), moveCaravan(char caravanId, string direction)).
    - Resolução de combates entre caravanas (resolveCombat(char userCaravanId, char barbarianId)).
    - Verificação de adjacência entre caravanas (areCaravansAdjacent(char caravan1Id, char caravan2Id) const).
    - Controle do comportamento da tempestade de areia (handleStorm(int l, int c, int radius)).

- Gestão do estado automático da caravana (setAutoMode(char caravanId, bool mode)).
- Atualização do comportamento de cada caravana (updateAutonomousBehaviors()).
- Remoção de caravanas (removeCaravan(char caravanId)) e itens expirados (removeExpiredItems()).
- Gestão da geração de itens (spawnItem()).
- Gestão da colisão das caravanas com os itens (handleItemPickup()).
- Gestão do ciclo do jogo (atualização do turno) (setCurrentTurn()).
- Apresentação de informação sobre as cidades (listCityInfo(char city_id)) e caravanas (listCaravanInfo(char caravan_id)).
- Compra de caravanas nas cidades (buyCaravan(char cityId, char type)).
- Gestão da compra (buyGoods(char caravanId, int amount)) e venda (sellGoods(char caravan_id)) de mercadorias.
- Gestão da contratação de tripulantes (hireCrew(char caravan_id, int amount)).
- Controlo do número de moedas do jogador (addCoins(int amount), removeCoins(int amount), getCoins() const).
- Obtenção do estado da posição do mapa (getPosition(int x, int y) const).
- Definição do estado da posição do mapa (setPosition(int x, int y, char value)).
- Apresentação dos preços (listPrices() const).
- Obtenção e definição do contador de caravanas (getCounterCaravans() const, setCounterCaravans(int value)).
- Renderização do mapa no buffer (render(Buffer& buffer) const).
- Gestão dos combates (handleCombats()).
- Controlo da remoção dos itens expirados (removeExpiredItems()).
- Geração de novos itens (spawnItem()).
- Gestão do recolhimento dos itens (handleItemPickup()).

❖ Item

- A classe Item e as suas derivadas representam os diferentes tipos de objetos mágicos que aparecem no mapa e interagem com as caravanas.
- Funcionalidades Principais (Classe Base):
  - Representação genérica de um item com posição, nome e duração (construtor Item(...)).
  - Abstração do efeito do item (applyEffect(Caravan& caravan)).
  - Gestão da duração dos itens (decreaseDuration(), isActive()).
  - Obtenção do nome do item (getName() const, getAsString() const).
  - Getters da posição da item.
- Classes Derivadas (Funcionalidades Específicas):
  - PandoraBox: Aplica um efeito negativo na caravana (perda de tripulação) (applyEffect(Caravan& caravan)).
  - TreasureChest: Aplica um efeito positivo na caravana (ganho de moedas) (applyEffect(Caravan& caravan)).
  - Cage: Aumenta a tripulação de uma caravana (applyEffect(Caravan& caravan)).
  - Mine: Destrói a caravana (applyEffect(Caravan& caravan)).
  - Surprise: Se o mapa tiver mais que 5 caravanas remove metade das caravanas, se não acrescenta 5 tripulantes (applyEffect(Caravan& caravan)).

- Simulator
  - Esta classe controla o ciclo principal do jogo e a interação com o utilizador.
  - Funcionalidades Principais:
    - Inicialização do jogo (construtor Simulator(int rows, int columns), start()).
    - Processamento dos comandos do utilizador (processCommand(const std::string& command)).
    - Simulação dos turnos do jogo (simulateTurn()).
    - Execução de comandos a partir de um ficheiro (processCommandsFromFile(const std::string& filename)).
    - Geração de caravanas bárbaras (spawnBarbarian(int l, int c)).
    - Controlo do ciclo de jogo (start()).

Funcionalidades Parcialmente implementadas

- Movimento Autónomo da Caravana Secreta
  - O que funciona:

        - A caravana secreta move-se autonomamente, procurando caravanas do utilizador para absorver água e cidades para reabastecer a sua água caso esteja abaixo de 20% do total.
        - A caravana move-se em direções aleatórias quando não encontra outros objetivos, e tem um ciclo de vida limitado a 10 turnos, desaparecendo após esse período.
    - O que não funciona:
        - A lógica faz com que a caravana Secreta uma vez que reabastece uma caravana ande sempre na sua orbita, o que está errado pois ela deveria reabastecer e procurar outras caravanas para reabastecer.
- Movimento Autónomo da Caravana Comercial
    - O que funciona:
        - A caravana comercial procura caravanas do utilizador num raio de 10 posições e tenta mover-se para a caravana mais próxima.
        - Caso não encontre nenhuma caravana, a caravana comercial procura por itens para recolher.
    - O que não funciona:
        - Uma vez que a caravana encontre uma caravana Militar fica sempre adjacente a esta e não procura itens.
- Comandos
    - O que funciona:
        - A maioria dos comandos funcionam como esperado no enunciado.
    - O que não funciona:
        - O comando terminar não dá a opção ao utilizar de começar outra simulação nem de sair da aplicação.

Funcionalidades Não Implementadas

- Remoção e adição de Caravanas Bárbaras de acordo com uma quantidade de turnos passados.
    - Não existe lógica que remova as Caravanas bárbaras depois de um valor de turnos nem as adicionar depois de um valor de turnos.
    - Estas só se removem se ficarem sem tripulação e em situação de combate.
- Lógica de movimentação se a caravana não tiver tripulação.

- Esta simplesmente é removida. Não havendo movimentação autónoma ou com certas configurações como referido no enunciado.

❖ Lógica de remoção de tripulantes se o nível de água iguala 0
- Não existe qualquer lógica que aplique algum efeito uma vez que a caravana fique sem água.

Conclusão

Com base nas funcionalidades implementadas, o simulador de caravanas apresenta um bom desenvolvimento, demonstrando uma estrutura modular e eficiente, com a implementação de conceitos de Programação Orientada a Objetos. A interação através de comandos de texto na consola, juntamente com o sistema de simulação por turnos,

proporciona uma experiência de jogo envolvente. Embora algumas funcionalidades ainda não tenham sido totalmente implementadas ou apresentem comportamentos inesperados, a base do simulador está sólida.