Sesc Escola Cuiabá

Curso Técnico em Programação de Jogos Digitais

Prof. Wanderson Timóteo

Esses exercícios são desafiadores e ajudam a desenvolver habilidades avançadas de POO em C# aplicadas a Jogos Digitais.

Faça os exercícios separadamente, sendo um projeto dotnet para cada exercício e adicione ao seu GitHub.

Exercícios:

"Exercícios de C# e POO em Jogos Digitais"

1. Sistema de NPCs com Inteligência Artificial

Crie uma classe abstrata NPC que represente personagens não jogáveis.

- Deve ter os métodos Mover() e Interagir(), que serão implementados por subclasses como Vendedor, Guarda e Vilão.
- O Vilão deve perseguir o jogador e atacá-lo automaticamente.

2. Sistema de Combate com Interfaces

Crie uma interface IAtacante com o método Atacar().

- Implemente essa interface em classes como Guerreiro, Mago e Arqueiro, onde cada um tem um tipo de ataque diferente.
- Adicione uma classe Inimigo, que pode ser atacado e perder pontos de vida.

3. Fábrica de Itens Aleatórios (Factory Method)

Crie um sistema de geração de itens aleatórios para um RPG.

- Use o padrão Factory Method para criar armas, poções e armaduras dinamicamente.
- Cada item deve ter diferentes efeitos no jogador (exemplo: aumentar ataque, recuperar vida, melhorar defesa).

4. Inventário e Gestão de Recursos

Crie uma classe Inventario que possa armazenar diferentes itens.

- O inventário deve ter um limite de capacidade e permitir adição e remoção de itens.
- Os itens podem ser usados para recuperar vida, aumentar dano ou aumentar velocidade.

5. Sistema de Habilidades com Herança

Crie uma classe base Habilidade com os atributos Nome, CustoDeMana e Dano.

- Crie habilidades específicas como BolaDeFogo, RaioDeGelo e GolpeDeEspada, que herdam da classe base e modificam os atributos.
- Permita que um personagem possa usar habilidades e consumir mana.

6. Estado do Jogo (Singleton)

Crie um sistema de gerenciamento do estado do jogo usando o padrão Singleton.

- O estado deve armazenar informações como nível atual, pontuação do jogador e número de vidas restantes.
- O jogo deve sempre acessar a mesma instância do estado, evitando múltiplas versões dele.

7. Sistema de Missões

Crie um sistema de missões onde o jogador pode aceitar e completar missões.

- Cada missão tem um objetivo específico (derrotar inimigos, coletar itens, explorar áreas).
- Use herança e polimorfismo para criar diferentes tipos de missões.

8. Gerenciamento de Personagens com Composição

Crie um sistema de personagens onde cada personagem tem uma arma, armadura e um tipo de ataque.

- Use composição em vez de herança para permitir que o personagem troque de arma e armadura durante o jogo.
- As armas e armaduras devem afetar os atributos do personagem.

9. Sistema de Níveis e Experiência

Crie um sistema de progressão de personagens.

- Cada personagem deve ganhar experiência ao derrotar inimigos e subir de nível.
- O aumento de nível deve melhorar atributos como força, agilidade e vida.

10. Sistema de Economia com Design Pattern Observer

Implemente um sistema de economia no jogo onde os preços dos itens variam de acordo com a oferta e demanda.

- Use o padrão Observer para notificar os NPCs comerciantes quando os preços mudarem.
- Exemplo: Se o jogador vende muitas espadas, o preço da espada diminui.

11. Simulação de Física em Plataforma 2D

Crie um sistema de movimentação para um jogo de plataforma.

- O personagem deve ser afetado por gravidade e colisões com o solo.
- Deve ter métodos como Pular(), Andar() e Cair().

12. Gerenciamento de Eventos com Delegates

Crie um sistema de eventos no jogo usando delegates e eventos em C#.

- Exemplo: Um evento de "Chefe Derrotado" que libera uma nova área no jogo.
- Outros eventos podem incluir "Jogador Subiu de Nível" ou "Missão Concluída".

13. Simulação de Jogo de Cartas

Crie um jogo de cartas estilo RPG onde cada carta representa um personagem com habilidades.

- Utilize herança para criar diferentes tipos de cartas (Guerreiro, Mago, Assassino).
- Implemente um sistema de turnos, onde cada jogador pode jogar cartas e atacar o oponente.

14. Sistema de Diálogos com Árvores de Decisão

Crie um sistema de diálogos para NPCs.

- O diálogo deve ter múltiplas escolhas que levam a respostas diferentes.
- Use árvores de decisão para determinar como a conversa se desenrola.

15. Algoritmo de IA para Inimigos (State Pattern)

Crie uma inteligência artificial para NPCs inimigos usando o padrão State.

- O inimigo pode alternar entre estados como "Patrulhando", "Perseguindo o Jogador" e "Atacando".
- Se o jogador se esconder, o inimigo deve retornar ao modo "Patrulhando".