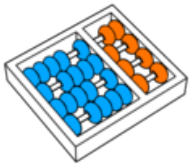

Universidade Estadual de Campinas
Instituto de Computação

Prof. Dr. Bruno Barbieri de Pontes Cafeo
cafeo@ic.unicamp.br
https://ic.unicamp.br/~cafeo/

UNICAMP

INF1900 - Projeto Final

Descrição do Projeto

Neste projeto final, você deve criar um jogo de Truco em C++ que incorpora e demonstra de forma abrangente os princípios da Programação Orientada a Objetos (OO). O jogo de Truco deve ser totalmente funcional, com suporte para dois jogadores humanos e implementar todas as regras do jogo de Truco.

Conceitos de OO

1. **Classes e Herança:**
   - Organize o projeto em classes que representem os elementos do jogo, como jogadores, cartas, rodadas, entre outros. Utilize herança quando aplicável para modelar relações de especialização entre as classes, como classes para diferentes tipos de cartas.
2. **Polimorfismo:**
   - Aplique o polimorfismo para permitir o tratamento genérico de objetos, como cartas. Isso pode ser feito através de funções virtuais em classes base, permitindo diferentes implementações em classes derivadas, se necessário.
3. **Encapsulamento:**
   - Mantenha os detalhes de implementação ocultos e forneça interfaces bem definidas para interagir com as classes. Isso inclui o acesso aos membros de classe (utilize modificadores de acesso como public, private e protected) e o uso de getters e setters quando apropriado.
4. **Abstração:**
   - Abstraia os conceitos do jogo de Truco em classes que representem entidades reais, como jogadores, cartas e partidas. Isso permite que o código seja mais fácil de entender e manter.

Concorrência e Multithreading

5. **Integração de Threads:**
   - Integre threads de maneira significativa para realizar operações em paralelo, como a execução das ações dos jogadores e a contagem de pontos, demonstrando a aplicação dos conceitos de multithreading. Por exemplo, permita que os jogadores façam suas escolhas simultaneamente.
6. **Técnicas de Sincronização:**
   - Utilize técnicas de sincronização e exclusão mútua para garantir a integridade dos dados compartilhados entre threads, evitando problemas como condições de corrida. Analise cuidadosamente seu código em busca de possíveis condições de corrida, onde duas ou mais threads podem acessar os mesmos dados simultaneamente. Utilize exclusão mútua para evitar essas condições de corrida, garantindo que as operações críticas sejam feitas de forma segura e ordenada.

Tratamento de Exceção

7. **Implementação de Tratamento de Exceções:**
   - Implemente tratamento de exceções para lidar com erros e situações excepcionais durante a execução do jogo, como movimentos inválidos dos jogadores ou problemas de alocação de memória.

Interface de Usuário

8. **Criação de Interface de Usuário:**

   - Crie uma interface de usuário amigável que permita aos jogadores interagirem com o jogo de Truco de forma intuitiva. Utilize a Win32 API (ou MFC) para criar e gerenciar janelas, controles e eventos, fornecendo uma experiência de usuário agradável. Isso inclui a representação gráfica das cartas, placar e mensagens informativas.

Utilização Adequada de Smart Pointers

9. **Utilização de Smart Pointers:**

   - Utilize smart pointers para gerenciar objetos que requerem alocação dinâmica de memória, como instâncias de jogadores ou recursos dinâmicos. Garanta que a utilização dos smart pointers seja aplicada de forma apropriada para gerenciar a vida útil dos objetos, evitando vazamentos de memória e garantindo a liberação automática de recursos quando não mais necessários.

Serialização e Persistência

10. **Serialização e Persistência:**

    - Implemente a serialização de dados do jogo para permitir que os jogadores salvem e carreguem partidas de Truco em qualquer ponto do jogo. Utilize técnicas de serialização para gravar e ler dados do jogo em arquivos, garantindo a capacidade de persistência. Utilize std::filesystem para eventuais manipulações de diretórios, como salvar e carregar partidas em pastas específicas.

Padrão arquitetural

11. **Implementação de padrão arquitetural:**

    - Implemente um padrão arquitetural em sua solução de forma a organizar o código visando potencializar propriedades importantes de OO. Espera-se aqui que seja implementado ou o padrão arquitetural MVC ou MVVM.

Documentação

12. **Geração de documentação:**

    - Deve ser gerado um documento descrevendo os requisitos do sistema. O documento deve seguir o template disponibilizado no arquivo anexo a essa atividade (`Exemplo_Requisitos_ProjetoFinal.docx`)
    - Além da documentação interna do código e o documento de requisitos, vocês devem gerar diagramas e esquemáticos do projeto. Pelo menos deve ser entregue o diagrama de classes do projeto. Também é recomendada a entrega de fluxograma e/ou diagramas de sequência de partes específicas do projeto.
    - Toda a documentação deve estar identificada no `README.md` do repositório.

Grupos

- O projeto deverá ser realizado em grupo de 4 pessoas.
- Os grupos previamente definidos estão no arquivo `Grupos.pdf` também junto a essa atividade.
- Os grupos foram definidos pelo SiDi de forma a balancear a experiência da equipe.
- Os líderes de cada grupo estão marcados em verde e serão responsáveis pela criação do repositório, bem como o envio do link do projeto no Github que será usado para a entrega.

Critérios de Avaliação

Seu projeto será avaliado com base nos seguintes critérios:

1. Organização e estrutura do código, demonstrando a aplicação efetiva dos princípios da Programação Orientada a Objetos.
2. Utilização apropriada dos conceitos de C++ aprendidos no curso, com ênfase em multithreading, tratamento de exceção, interface de usuário, smart pointers e serialização.
3. Eficiência do código para garantir um desempenho significativo, demonstrando a aplicação prática dos conceitos de threads e smart pointers.
4. Comentários explicativos ao longo do código, destacando a aplicação dos tópicos aprendidos no curso.

Entrega

- Deve ser criado um repositório para o projeto no Github. A entrega final deve ser identificada por uma Github tag.
- As informações relevantes sobre o software, bem como o número do grupo e o nome dos membros que realizaram a entrega devem estar no arquivo `README.md`.
- O repositório deve estar acessível para a nossa correção.
- A planilha com o link do projeto no Github deverá ser preenchida até a data de entrega **apenas pelo líder da equipe**.
- A data da tag será checada para avaliar o momento da entrega.

Data de entrega: 26/01/2024 até 23h59.