Relatório de Arquitetura – Editor SVG

Autor: Gustavo do Nascimento Pereira

Matrícula: 2024001644

Visão Geral

Para este projeto, minha principal preocupação foi criar uma base que pudesse crescer. Sabendo que precisaríamos de uma versão em terminal (Etapa 2) e depois uma gráfica (Etapa 3), a arquitetura

Model-View-Controller (MVC) foi a escolha natural.

Essa abordagem me permitiu separar o projeto em três partes claras:

- **Modelo:** A lógica central do editor. É aqui que vivem as classes `Canvas`, a `Shape` abstrata e as formas concretas como o `Polygon`.
- **Visão (View):** A interface com o usuário. Por enquanto, temos a `CLIView` que imprime no terminal, mas a estrutura está pronta para receber uma `GUIView` no futuro.
- **Controlador (Controller):** O `EditorController` é o cérebro que conecta tudo. Ele responde às ações, atualiza o modelo e diz à visão o que mostrar.

Principais Decisões de Design

- **Polimorfismo:** Para gerenciar diferentes formas (polígonos, e futuramente círculos, etc.) em uma única tela, criei uma classe base abstrata `Shape`. Ela funciona como um "contrato", garantindo que qualquer forma que criarmos saiba como se mover (`move()`) e como se transformar em texto SVG (`toSVGString()`). Isso permite que o `Canvas` gerencie uma lista de `std::vector<std::unique_ptr<Shape>>`, tratando todas as formas da mesma maneira, sem precisar saber o tipo específico de cada uma.
- **Composição:** Eu usei composição para modelar relações do tipo "tem-um", o que deixou o design mais claro e flexível. Por exemplo, um `Canvas` *tem uma* coleção de `Shapes`, e um `Polygon` *tem uma* lista de `Points` que formam seus vértices. Essa abordagem é mais robusta do que tentar forçar uma relação de herança onde não faz sentido.
- **Gerenciamento de Memória:** Para evitar vazamentos de memória, uma preocupação central em C++, eu usei `std::unique_ptr` para gerenciar o ciclo de vida das formas. O `Canvas` é o "dono" das `Shapes` que estão na tela. Ao usar `unique_ptr`, garantimos que quando uma forma for removida ou o programa fechar, sua memória será liberada automaticamente. Isso demonstra o princípio RAII na prática.