1. Descrição

Uma aplicação gráfica interativa construída em C++ e Lua que exibe um cubo tridimensional com seis faces independentes sobre um fundo de partículas. O usuário pode rotacionar o cubo com as teclas WASD, misturar cores aditivamente nas faces do cubo, carregar imagens para qualquer face, controlar zoom e rotação da textura aplicada. Um monitor de performance pode ser ativado em tempo de compilação, abrindo uma segunda janela com métricas de FPS, tempo de frame, uso de memória e estatísticas de cache de texturas.

2. Divisão de Responsabilidades entre as Linguagens

C++ é responsável pela inicialização e criação das janelas, renderização do cubo (vértices, quads, projeção de perspectiva), carregamento de texturas com stb_image, gerenciamento do estado Lua e pela ponte que conecta as duas linguagens.

Lua controla toda a lógica que se beneficia de ser modificável sem recompilação, como geração e animação das estrelas do fundo, mapeamento de teclas WASD a incrementos de rotação, a lógica de mistura aditiva de cores e o controle de padrão e foto de cada face do cubo.

3. Por que Lua

Lua foi projetada desde o início para ser embutida em aplicações escritas em C e C++. Sua API em C++ é estável e bem documentada, e a própria implementação de referência é desenvolvida em C padrão. Isso facilita a integração direta, sem a necessidade de mecanismos complexos para interligar as duas linguagens.

Outro fator é sua eficiência em tarefas executadas em tempo de execução. Lua oferece flexibilidade e dinamismo para scripts, tornando-se um bom complemento para C++, que pode permanecer responsável pelas partes de maior desempenho e controle estrutural do sistema.

4. Integração

A integração é feita inteiramente pela API C do Lua (liblua), sem bibliotecas de binding de terceiros. A classe LuaBridge encapsula um ponteiro opaco para o lua_State, seguindo o padrão PIMPL para que os cabeçalhos Lua não precisem ser expostos fora do módulo bridge. Todos os arquivos do projeto que dependem do bridge incluem apenas lua_bridge.h, que não declara nada do Lua diretamente.

O ciclo de uma chamada segue sempre o mesmo protocolo de pilha Lua: empurrar a função global com lua_getglobal, empurrar os argumentos com lua_push*, chamar com lua_pcall especificando número de argumentos e retornos esperados, ler os retornos com lua_to* e limpar a pilha com lua_pop. Erros de pcall são impressos em stderr e a pilha é corrigida, garantindo que o estado Lua permaneça válido mesmo diante de falhas em scripts.

Nenhum recurso do C++ ou do Lua precisou ser modificado para viabilizar a integração pois a API pública do Lua já oferece tudo o que é necessário. O único cuidado arquitetural relevante foi isolar o lua_State atrás de PIMPL para evitar que o include path do Lua vazasse para o restante do projeto, mantendo os headers de interface limpos.