Documentação da Implementação: Integração C e Lua via Ray Tracing

Gerson Clara e Henrique Valeza

Fevereiro de 2026

1 Aplicação Escolhida

A aplicação gráfica escolhida para demonstrar o uso conjunto de duas linguagens foi um simulador de *Ray Tracing* básico. Trata-se de um gerador de imagens estáticas que calcula a cor dos pixels simulando raios de luz virtuais em um ambiente tridimensional, processando a intersecção geométrica desses raios contra objetos matemáticos (como esferas) para criar efeitos simples de sombreamento difuso.

2 Divisão e Vocação das Linguagens

Para cumprir o requisito de utilizar ferramentas com vocações distintas, a arquitetura da aplicação foi dividida da seguinte forma:

- **Linguagem C (Serviço de Cálculo):** A linguagem C foi encarregada da implementação do motor de Ray Tracing. Toda a álgebra linear (cálculos vetoriais, produto escalar) e o processamento intenso dos loops de varredura de pixels ocorrem nativamente no código C.
- **Linguagem Lua (Interface e Apresentação):** Lua atuou como a linguagem de interface com o usuário. O script de apresentação define os parâmetros de resolução da imagem, o nome do arquivo de saída e aciona o motor de cálculo.

3 Método de Interface entre as Linguagens

A integração entre as linguagens não ocorre por via de arquivos intermediários ou chamadas de sistema (*system calls*), mas sim em nível de memória utilizando a **API C do Lua**.

O processo de comunicação segue as seguintes etapas técnicas:

1. **Embutimento e Inicialização:** O programa principal em C inicializa a máquina virtual do Lua através da função `luaL_newstate()`. Em seguida, a função de cálculo em C (`l_executar_raytracing`) é registrada e exposta para o ambiente virtual do Lua sob o nome `meuraytracer.gerar_imagem`.

2. **Execução da Interface:** O C chama a função `luaL_dofile` para executar o script de interface do usuário, transferindo o controle do fluxo lógico para a linguagem interpretada.

3. **A Pilha Virtual (Stack):** Quando o script Lua invoca a função para gerar a imagem, ele empilha os argumentos (largura, altura e nome do arquivo) em uma Pilha Virtual LIFO (*Last In, First Out*) gerenciada pela API.

4. **Desempilhamento e Processamento:** O motor C recupera o controle e acessa essa pilha utilizando rotinas de checagem estrita de tipos (como `luaL_checkinteger` e `luaL_checkstring`). Com os dados em mãos, executa a renderização física em C, salva o arquivo, empilha um valor booleano confirmando o sucesso da operação, e devolve o controle final para o script Lua apresentar a mensagem de conclusão ao usuário.