Geração do Conjunto de Mandelbrot com Integração entre C++ e JavaScript via WebAssembly

Diogo Krüger Souto, Kananda Barbosa Winter, Luca Rodrigues da Silva, Maria Luiza Batista Prata, Milena Alves Ferreira.

1. Introdução

Este trabalho apresenta uma aplicação web para geração do Conjunto de Mandelbrot utilizando integração entre código C++ e JavaScript por meio de WebAssembly. O objetivo principal não é apenas a visualização do fractal, mas a demonstração de interoperabilidade entre linguagens com modelos de execução distintos, explorando o uso de código nativo no ambiente web para computação intensiva.

2. Descrição da Aplicação

O Conjunto de Mandelbrot é um fractal definido por uma iteração no plano complexo. A aplicação permite gerar e visualizar o fractal em um elemento *<canvas>* no navegador. O usuário interage com a interface web por meio de um botão que aciona o processo de cálculo e renderização da imagem.

3. Arquitetura do Sistema

A aplicação é dividida em três camadas principais:

- **C++**: responsável pelo cálculo matemático intensivo do fractal e pela escrita dos pixels em um buffer de memória linear.

- **WebAssembly (WASM)**: camada intermediária que permite executar o código C++ dentro do navegador.

- **JavaScript + HTML (Canvas)**: responsável por carregar o módulo WebAssembly, chamar as funções exportadas pelo C++ e renderizar os dados no canvas.

Fluxo simplificado:

1. O JavaScript carrega o módulo WebAssembly compilado a partir do C++.

2. O JavaScript solicita ao C++ a alocação de um buffer de memória para os pixels.

3. O C++ calcula o fractal e escreve os valores RGBA no buffer.

4. O JavaScript lê diretamente a memória linear do WASM e desenha a imagem no canvas.

4. Integração entre C++ e JavaScript

A comunicação entre C++ e JavaScript ocorre por meio da memória linear do WebAssembly. As funções *allocate_buffer* e *generate_mandelbrot* são exportadas pelo módulo WASM e chamadas diretamente pelo JavaScript. O ponteiro retornado pelo C++ é interpretado pelo JavaScript como uma região de memória (Module.HEAPU8.buffer), permitindo acesso direto aos bytes gerados. Essa abordagem evita serialização de dados e demonstra interoperabilidade real entre linguagens.

5. Tecnologias Utilizadas

- C++
- WebAssembly (via Emscripten)
- JavaScript
- HTML5 Canvas
- CMake
- Python (servidor HTTP local)

6. Conclusão

O projeto demonstra de forma prática como código C++ pode ser reutilizado no ambiente web através de WebAssembly, permitindo ganho de desempenho em

tarefas computacionalmente intensivas e integração eficiente com interfaces desenvolvidas em JavaScript. A aplicação evidencia o potencial do WASM como ponte entre linguagens de baixo nível e aplicações web modernas.