Documentação da Implementação

Autores: Caio Ribeiro, João Vitor Ceppo, Kevin Rehbein, Marlon Weber

Aplicação Escolhida

O projeto implementa uma simulação 2D de partículas que demonstra a integração entre as linguagens Python e C++. A aplicação consiste em uma janela com fundo preto onde 500 partículas, representadas por círculos de cor ciano, se movem e colidem elasticamente. Esta abordagem de desenvolvimento conjunto foi escolhida para alavancar as vocações distintas de cada linguagem: C++ para os cálculos de alta performance e Python para a rápida prototipagem da interface gráfica.

Divisão de Responsabilidades por Linguagem

A divisão de responsabilidades é clara. O C++ é inteiramente responsável pela lógica computacionalmente intensiva da simulação. Isso inclui a inicialização das partículas com posições e velocidades aleatórias, a atualização contínua de suas posições com base no tempo decorrido e a complexa tarefa de detectar e resolver colisões tanto com as paredes da caixa quanto entre as próprias partículas. Por outro lado, o Python gerencia toda a interface com o usuário e o controle da aplicação. Utilizando a biblioteca Tkinter, o Python cria a janela, renderiza as partículas como formas no canvas e controla o loop de animação, garantindo que a tela seja atualizada a uma taxa de aproximadamente 60 quadros por segundo.

Método de Interface Entre as Linguagens

A comunicação entre as duas linguagens é viabilizada por uma Interface de Função Externa, ou FFI, através da biblioteca padrão ctypes do Python. O código C++ é compilado como uma biblioteca compartilhada, com extensão .so em Linux ou .dll em Windows, um processo automatizado pelo Makefile fornecido. Para garantir que o Python possa encontrar as funções C++ por seus nomes originais, elas são declaradas com extern "C". Isso impede o compilador C++ de alterar os nomes das funções, um processo conhecido como name mangling.

No lado do Python, o script main.py carrega essa biblioteca em tempo de execução usando a função ctypes.CDLL. Para garantir a compatibilidade de dados, uma classe Python é criada herdando de ctypes.Structure, espelhando exatamente o layout da struct Particula definida em C++. Além disso, os tipos de argumentos e de retorno de cada função C++ são explicitamente definidos no código Python, assegurando a correta passagem de dados como inteiros, ponteiros e números de ponto flutuante.

O Makefile automatiza todo processo de compilação. Ele irá compilar o código C++ em uma biblioteca compartilhada (libsimulation.so em Linux ou libsimulation.dll em Windows).

O fluxo de dados durante a execução é projetado para ser altamente eficiente. Inicialmente, o Python chama a função simulation_init para que o C++ aloque memória e configure o estado inicial das partículas. Em seguida, dentro de um loop de animação contínuo, o Python primeiro chama simulation_update, delegando todos os cálculos de física para o C++. Imediatamente depois, ele chama get_particulas_data, que retorna um ponteiro direto para o array de partículas na memória do C++. O Python então lê as coordenadas de cada partícula diretamente desse ponteiro e atualiza os objetos gráficos no canvas do Tkinter. Essa abordagem de acesso direto à memória é fundamental para a performance do projeto, pois evita a sobrecarga de copiar todo o conjunto de dados entre as linguagens a cada quadro.