Wolfenstein 3D: Arquitetura da Engine

Grupo: Manassés Paterline, Davi Corradi, Daniel Murad,
Arthur da Matta, Guilherme Altoé

1 Arquitetura da Engine e Divisão de Camadas

A arquitetura da engine do Wolfenstein 3D, detalhada no Capítulo 4.4 do Game Engine Black Book, foi projetada com um forte foco em modularidade e domínio do hardware da época. O código-fonte foi inteligentemente dividido em duas camadas principais, separando o que o jogo de fato faz de como ele executa as operações na máquina.

A primeira é a camada de lógica da engine (WL_*), que é estritamente responsável pela lógica do próprio jogo. A segunda é a camada de hardware (ID_*), que lida com os gerenciadores do sistema e realiza a interface direta com os componentes físicos da máquina.

2 Os Sete Gerenciadores do Sistema

Para resolver os gargalos de hardware e integrar os módulos de forma eficiente, a arquitetura dividiu suas responsabilidades em sete gerenciadores principais: Memory, Page, Video, Cache, Sound, User e Input.

2.1 Memory Manager e Page Manager

O *Memory Manager* foi desenvolvido para lidar com a severa restrição de apenas 640Kb de memória RAM usável no sistema. Para contornar problemas de fragmentação de memória, a equipe substituiu a função padrão de alocação (`malloc`). Em seu lugar, implementaram um controle total da RAM que utilizava blocos de memória organizados manualmente por meio de uma lista encadeada.

Trabalhando em conjunto, o *Page Manager* resolvia a impossibilidade de manter tudo na memória ao mesmo tempo (como mapas, texturas, sprites e sons), o que causava carregamentos muito lentos e travamentos durante o *gameplay*. Suas principais características incluem:

- Implementação de um sistema de paginação com carregamento sob demanda.
- Manutenção inteligente de dados, mantendo apenas o necessário carregado na RAM.
- Carregamento de dados de fase (precaching) em segundo plano enquanto a tela "Get Psyched!"é exibida ao jogador.

2.2 Video Manager e Cache Manager

Lidando com os gráficos limitados da época, o *Video Manager* realizava acesso direto à memória VGA do computador. Ele fazia uso da linguagem Assembly para garantir uma manipulação de pixels extremamente eficiente. Essa renderização era subdividida em uma camada de baixo nível (VL_*, construída em C e Assembly) e uma de alto nível (VH_*), dedicada a menus 2D.

Como a leitura de dados do disco rígido era um processo muito lento, o *Cache Manager* atuava carregando e descompactando os assets do jogo. Ele mantinha esses dados (comprimidos por métodos como Huffman e RLEW) prontos para o uso imediato pela engine.

2.3 Sound, Input e User Manager

A diversidade de componentes nos PCs da época exigia soluções robustas de compatibilidade. O *Sound Manager* fornecia uma camada de abstração que unificava o suporte a diferentes hardwares de áudio, como PC Speaker, AdLib e Sound Blaster. Assim, um único gerenciador garantia o funcionamento do jogo em múltiplos dispositivos sonoros.

Seguindo o mesmo princípio, o *Input Manager* ficava encarregado das entradas do jogador. Ele convertia os sinais de diferentes periféricos — teclado, mouse e joystick — em uma interface única de entrada. Isso facilitava imensamente a compatibilidade entre hardwares e simplificava a lógica interna do jogo.

Por fim, o *User Manager* era responsável pela interface interativa. Gerenciado principalmente pelo módulo WL_MENU, ele cuidava da renderização de textos, dos menus e do HUD (interface durante o jogo), garantindo a separação de responsabilidades entre a interface visual do jogador e o motor principal do jogo.