Arquitetura de Software

Capítulo 4 — Análise do Código Fonte

Integrantes do Grupo

Felipe Mavigno Stein

Kaiky Viglioni Tavares Moura

Marco Antonio Faustini Pessoa

Nícolas Medeiros de Pinho

Sérgio Alexandre Gobbi Rebello

20 de abril de 2026

Sumário

1 Obtenção do Código Fonte

O código-fonte do motor de jogo de *Wolfenstein 3D* foi disponibilizado originalmente no servidor FTP da id Software em 21 de julho de 1995. Notavelmente, o arquivo permanece no mesmo URL original décadas depois, o que é um fato raro na web. Atualmente, a forma mais recomendada e rápida de acesso é através do GitHub, onde a id Software migrou seus projetos de código aberto por volta de 2012.

2 Primeiro Contato e Estatísticas

Após a descompressão do arquivo original, o código revela uma estrutura predominantemente escrita em C, com porções críticas em **Assembly**. O uso de Assembly era destinado a otimizações de gargalos e operações de I/O (Input/Output) de baixo nível para vídeo e áudio.

Utilizando ferramentas de análise de código (`cloc`), observam-se as seguintes proporções:

- **C++:** 21.169 linhas
- **C/C++ Headers:** 3.900 linhas
- **Assembly (TASM):** 2.150 linhas

Apesar de não ser sempre um fator de comparação válido, analisar o número de linhas do programa é um indicador ótimo de seu tamanho e proporção, com um total de aproximadamente 27.223 linhas de código (*SLOC - Source Lines Of Code*), o motor do Wolfenstein 3D é considerado pequeno para os padrões modernos. Em comparação, o navegador Google Chrome possui cerca de 1,7 milhão de linhas, e o kernel Linux ultrapassa as 15 milhões.

Tabela 1: Crescimento da complexidade nos motores da id Software

Motor de Jogo	**Linhas de Código (SLOC)**
Wolfenstein 3D	27.223
Doom	39.080
Quake	78.961
Quake 2	138.240
Quake 3	233.952
Doom 3	601.047

Uma curiosidade é que John Carmack, um dos programadores do jogo, disse que os editores de texto da época não tinham verificação ortográfica, então ocasionalmente

ocorriam erros de digitação e soletração. John disse que tinha uma relativa dificuldade nessa parte, então muitas vezes ele escrevia a palavra "Column"como "collumn"(coluna em inglês), e que isso foi inconveniente ao ponto de um usuário mandar um email para eles dizendo:
'É "COLUMN", burro do C%&#!!'

3 A Visão Geral (Big Picture)

A arquitetura do motor é dividida em três blocos principais que se comunicam via memória compartilhada:

1. **Menu 2D:** Responsável pela interface de configuração.
2. **Motor 3D (Renderer):** Onde o processamento principal ocorre.
3. **Sistema de Som:** Executado de forma concorrente aos outros dois.

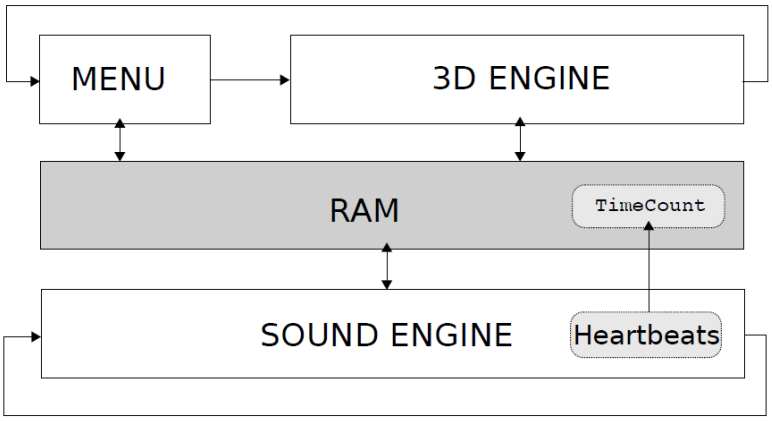

Figura 1: Diagrama dos três blocos principais do motor de jogo.

O sistema de som é orientado por interrupções, agindo como o "batimento cardíaco"do motor, controlando o tempo global através da variável `TimeCount`.

4 O Loop Principal

O ponto de entrada do programa em `void main()` inicializa os gerenciadores e entra no loop de demonstração. Um detalhe técnico importante da época é que, por rodar em **Modo Real**, os tipos de dados diferem do padrão de 32 bits: `int` e `word` possuem 16 bits, enquanto `long` possui 32 bits.

Listing 1: Estrutura de inicialização no Wolf3D

```
void InitGame() {
    MM_Startup();    // Gerenciador de Memoria
    VW_Startup();    // Gerenciador de Video
    IN_Startup();    // Gerenciador de Entrada
    SD_Startup();    // Gerenciador de Som
    CA_Startup();    // Gerenciador de Cache
    BuildTables();   // Tabelas de lookup (sin/cos)
}
```

4.1 Otimização para 386

Embora compilado para 16 bits, o jogo detecta se a CPU é um Intel 386 através da função `Patch386()`. Se detectado, o motor "patcheia"seu próprio código em tempo de execução, substituindo divisões de 32 bits da biblioteca do Borland C por instruções nativas que utilizam os registradores `eax` e `edx`, resultando em um ganho significativo de performance.

5 O Sistema de Áudio e Interrupções

Como o MS-DOS não oferecia suporte a processos ou *threads*, a única forma de executar áudio simultaneamente ao processamento gráfico era através de **ISR** (*Interrupt Service Routines*). O motor instala uma rotina na Tabela de Vetores de Interrupção, que pode ser disparada em frequências de 140Hz a 7000Hz, dependendo do hardware de som utilizado (PC Speaker ou Sound Blaster).

Listing 2: Configuração da velocidade do Timer para Áudio

```
static void SDL_SetTimerSpeed(void) {
    word rate;
    void interrupt (*isr)(void);

    if ((DigiMode == sds_PC) && DigiPlaying) {
        rate = TickBase * 100; // 7000 Hz
        isr = SDL_t0ExtremeAsmService;
    } else {
        rate = TickBase * 2;   // 140 Hz
        isr = SDL_t0SlowAsmService;
    }
    setvect(8, isr); // Instala a interrupcao
}
```

6 Conclusão

A arquitetura de *Wolfenstein 3D* reflete os desafios de programação do início dos anos 90: a necessidade de lidar com limitações do modo real, a falta de multitarefa nativa do sistema operacional e a dependência de otimizações manuais em Assembly para garantir a fluidez em hardware limitado.