Software

CAPÍTULO 4 — WOLFENSTEIN 3D

Daniel Frigini Rosa
Victor Samora Nunes
Samuel
Alysson

4.1 Pegando o Código Fonte

Lançamento

O código foi liberado em **21 de julho de 1995** nos servidores FTP da id Software — e ainda é acessível pelo mesmo link após mais de 20 anos.

GitHub

Por volta de **2012**, a id Software migrou todo o seu código open source para o GitHub, onde os arquivos permanecem disponíveis.

```
ftp://ftp.idsoftware.com/idstuff/source/wolfsrc.zip
```

```
$ git clone git@github.com:id-Software/wolf3d.git
```

4.2 Primeiro Contato

```
$unzip WOLFSRC.1
```

wolfscr.zip

O pacote contém um arquivo **autoextraível (.exe SFX)** criado por PKZIP — prático na época, mas obsoleto hoje.

PKZIP: ferramenta pioneira de compactação desenvolvida por Phil Katz em 1989 para MS-DOS. Introduziu o formato .zip. Arquivos SFX descompactam automaticamente sem softwares externos.

Analisando o Código com cloc

A ferramenta **cloc** (Count Lines of Code) analisa arquivos de uma pasta e coleta estatísticas: linhas de código, comentários e linhas em branco por linguagem.

```
$ cloc-1.64.pl WOLF3D

96 text files.
94 unique files.
27 files ignored.

-------------------------------------------------------------------
Language                files          blank        comment           code
-------------------------------------------------------------------
C++                        26           5750           6201          21169
C/C++ Header               42            802            660           3900
Assembly                   10            669            732           2150
DOS Batch                   1              1              0              4
-------------------------------------------------------------------
SUM:                       79           7222           7593          27223
-------------------------------------------------------------------
```

Linguagens e Métricas

O código de **Wolfenstein 3D** é composto em cerca de **90% por C**, com uso de **assembly** para otimizações em gargalos de desempenho e para tarefas de baixo nível, como **vídeo** e **áudio**.

C

Linguagem principal do jogo

C/C++ Header

Arquivos de cabeçalho

(.h)

Assembly

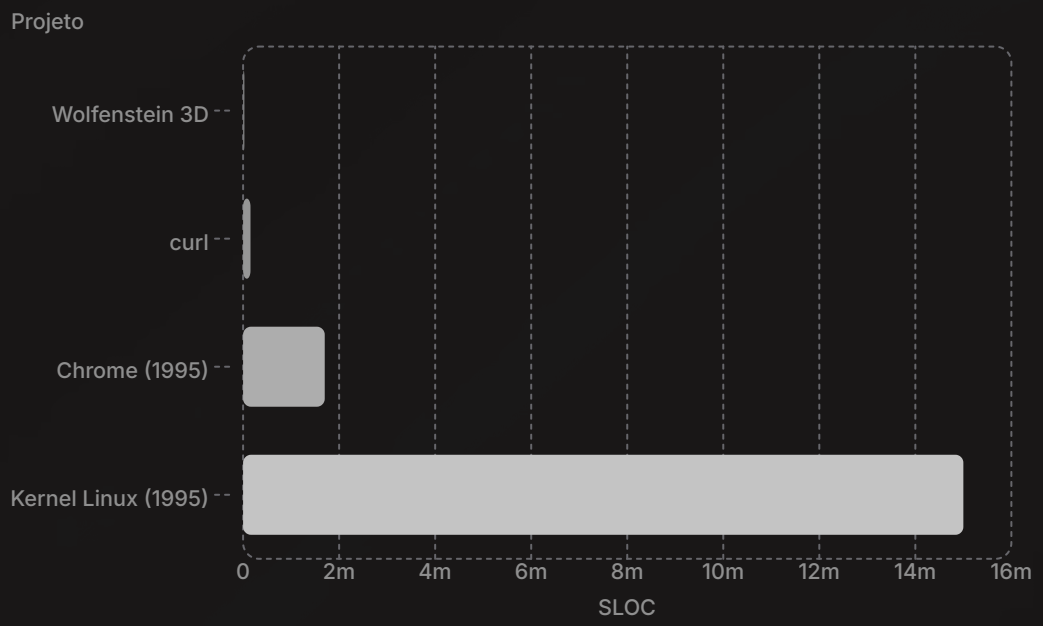
Otimizações críticas e acesso de baixo nível. Arquivos .asm, etc.

DOS Batch

Scripts .bat usados no ambiente DOS para automatizar compilação, montagem e etapas do build do projeto.

SLOC (*Source Lines of Code*) é a métrica de linhas de código-fonte usada para prever esforço de desenvolvimento, estimar produtividade e avaliar mantenabilidade.

Wolfenstein em Números



Source Lines of Code (SLOC)

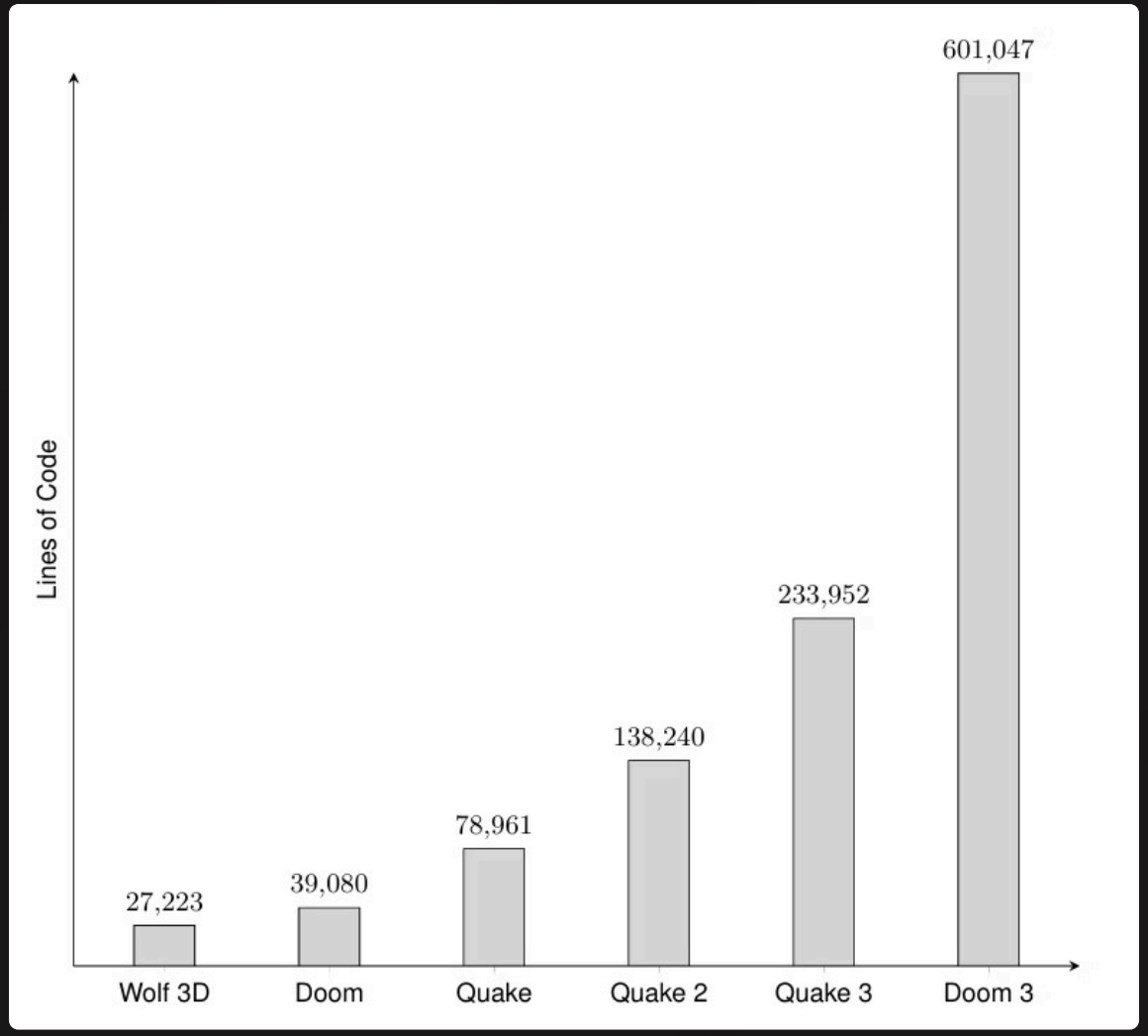
Com **27.223 SLOC**, o Wolfenstein 3D é minúsculo comparado a softwares modernos — o Chrome e o Kernel Linux ultrapassam 40 milhões de SLOC hoje.

SLOC mede o tamanho de um programa contando linhas do código-fonte. É mais útil para **comparar proporções** do que avaliar uma base isolada.

Uma Lembrança Inesquecível

> "We didn't have spell checkers in our editors back then, and I always had poor spelling. The word 'collumn' appears in the source code dozens of times. After I released the source code, one of the emails that stands out in memory read: It's 'COLUMN', you dumb FUCK!"
>
> — **John Carmack**, Programador

O lançamento do código-fonte foi recebido de forma extremamente positiva, gerando elogios — e comentários memoráveis como este.



Linhas de código dos motores de jogos da id Software

Além do Código-Fonte

O pacote contém mais do que apenas código:

GOODSTUF.TXT

Dois e-mails de fãs — um ex-prisioneiro de guerra e um funcionário da Microsoft — demonstrando o sucesso do jogo.

SIGNON.OBJ

Tela de inicialização com specs do sistema (RAM, EMS, XMS, joystick, som) incorporada ao binário.

GAMEPAL.OBJ

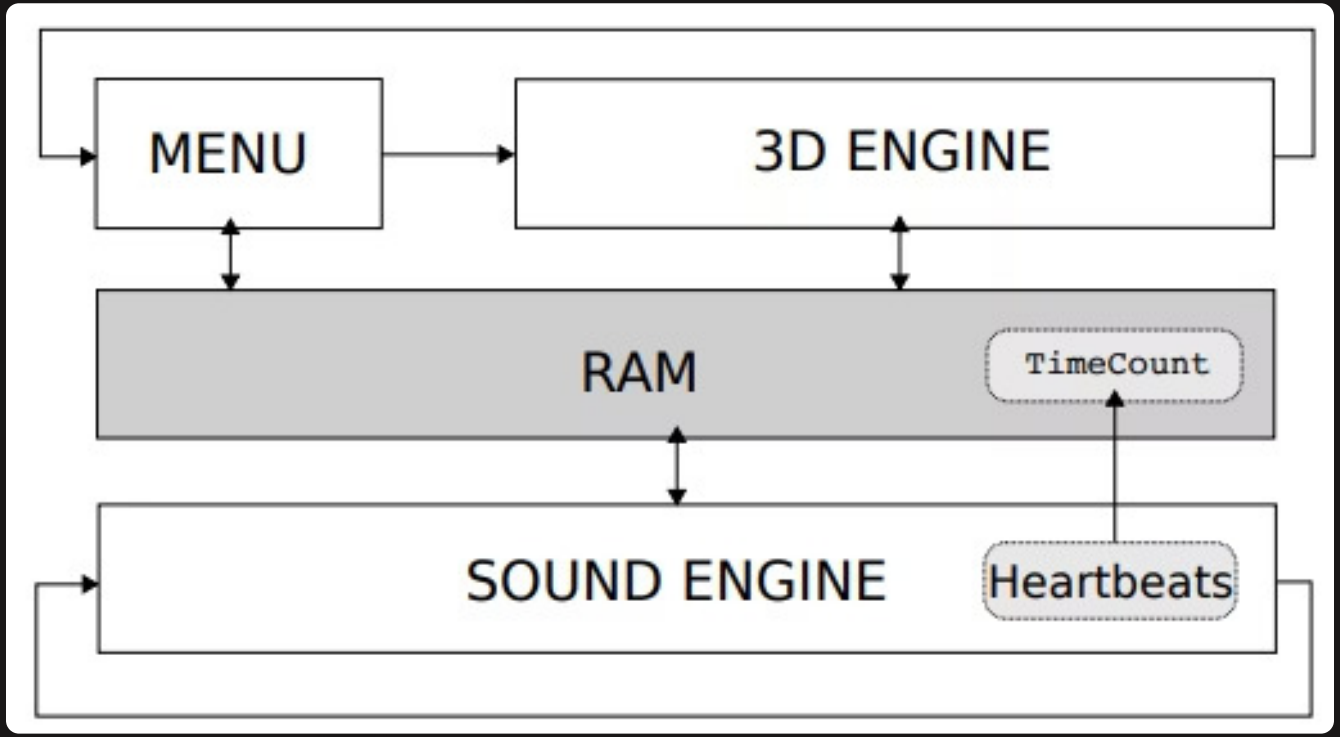
Paleta do jogo, codificada e vinculada ao executável pelo mesmo motivo que SIGNON.OBJ.

README

Instruções de compilação e arquivos resultantes de tentativas anteriores de build.

4.3 Big Picture: Arquitetura do Motor

As três sessões se comunicam por **memória compartilhada**. O renderizador grava solicitações de som na RAM, lidas pelo loop de som. O sistema de som também grava na RAM para os renderizadores — é responsável pelo **ritmo de todo o motor**, atualizado pela variável `TimeCount`.



Desenrolando o Loop Principal

Tipos e Modo Real

Em modo real, os tipos em C diferem do esperado em 32 bits:

- `int` e `word` → 16 bits
- `long` e `dword` → 32 bits

O Patch386 detecta CPUs 386 e substitui a divisão da Borland por instruções usando os registradores `eax` e `edx`.

> `eax` e `edx` são registradores de propósito geral de 32 bits — versões estendidas de `ax` e `dx` (16 bits).

01

InitGame()

Inicializa todos os gerenciadores do motor.

02

DemoLoop()

Loop principal: chama renderizadores 2D e 3D indefinidamente.

03

PlayLoop()

Coração do jogo: input → update → render. Execução constante até o encerramento.

Desenrolando o Loop Principal

Init

```
void InitGame () {
MM_Startup () ; // Gerenciador de memória
SignonScreen () ; // Mostra sistema de configuração
VW_Startup () ; // Gerenciador de vídeo
IN_Startup () ; // Gerenciador de entrada
PM_Startup () ; // Gerenciador de memória principal
PM_UnlockMainMem () ;
SD_Startup () ; // Gerenciador de som
CA_Startup () ; // Gerenciador de cache
US_Startup () ; // Gerenciador de utilidades
InitDigiMap () ;
ReadConfig () ;
CA_CacheGrChunk ( STARTFONT ) ; // Carrega fonte para cache
MM_SetLock (& grsegs [ STARTFONT ] , true ) ; // Trava fonte na memória
LoadLatchMem () ; // Carrega memória de latch
BuildTables () ; // Constrói tabelas de ângulos, distâncias, etc
SetupWalls () ; // Configura paredes do jogo
}
```

Demoloop

```
void DemoLoop () {
StartCPMusic ( INTROSONG ) ;
PG13 () ; // Mostra tela de classificação
while (1) {
CA_CacheScreen ( TITLEPIC ) ;
CA_CacheScreen ( CREDITSPIC ) ;
DrawHighScores () ;
PlayDemo (0) ;
GameLoop () ; // Loop principal do jogo
SetupGameLevel () ;
StartMusic () ;
PM_CheckMainMem () ;
PreloadGraphics () ;
DrawLevel () ;
PlayLoop () ; // Loop de jogo
StopMusic () ;
}
Quit (" Demo loop exited ??? ") ;
}
```

PlayLoop

```
void PlayLoop () {
PollControls () ; // Pega controles do jogador
MoveDoors () ; // Move portas
MovePWalls () ; // Move paredes móveis
for ( obj = player ; obj ; obj = obj - > next )
DoActor ( obj ) ; // Move atores
ThreeDRefresh () { // Desenha tela
VGAClearScreen () ; // Limpa tela
WallRefresh () ; // Desenha paredes
DrawScaleds () ; // Desenha sprites
DrawPlayerWeapon () ; // Desenha arma do jogador
[...] // Desenhar demais elementos da tela
}
UpdateSoundLoc () ; // Atualiza posição do som (Stereo sound loc
}
```

Sistema de Som e Interrupções

SDL_SetTimerSpeed

O prefixo **SDL_** significa **SounD Low level** — nada a ver com a biblioteca Simple DirectMedia Layer (que nem existia em 1991).

ISR (Rotina de Interrupção)

Instalada na Tabela de Vetores de Interrupção, pode ser chamada em frequências de **140Hz, 700Hz ou 7000Hz** conforme a necessidade.

Por que Interrupções?

Sem suporte a processos ou threads no SO, era a única forma de executar o som **simultaneamente** ao restante do motor.