Game Engine Black Book: Wolfenstein 3D

Arquitetura de Computadores II

Lucas Pizzol, Erlon Felipe, Guilherme Gasperazzo, Pablo, Gabriel Araújo

Turma: CC5M

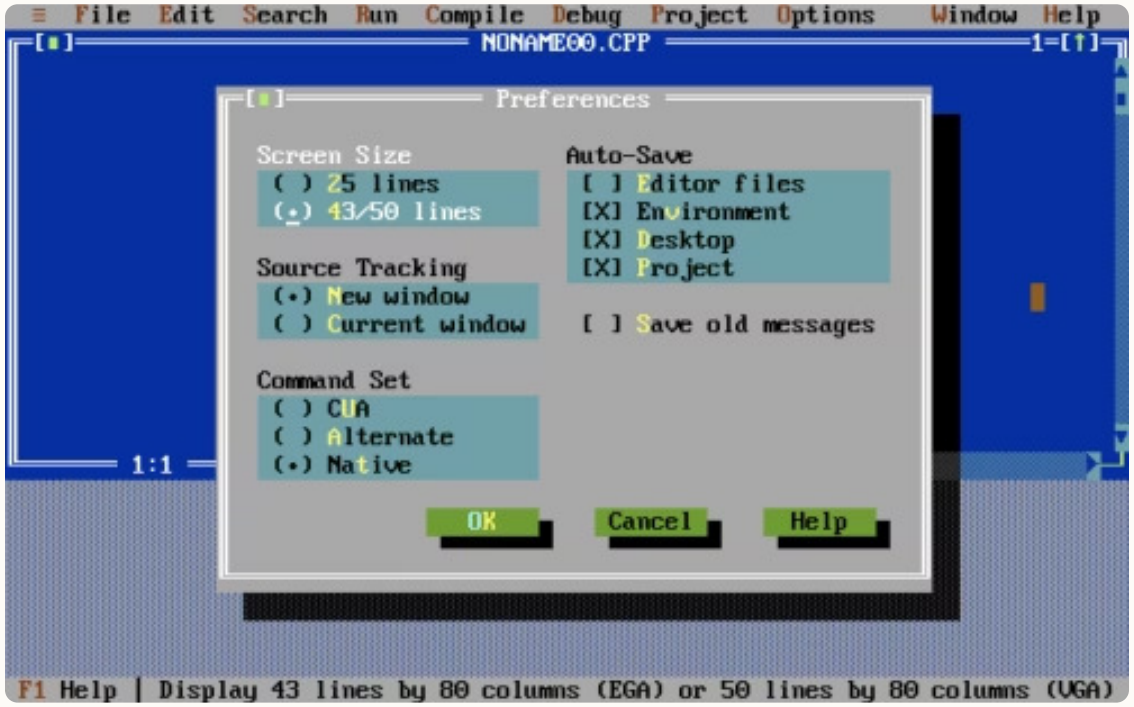
Programação e Assets no Wolfenstein 3D

Uma visão técnica das seções 3.2, 3.3 e 3.4

Programação

Gráficos

Pipeline de Assets



3.2 — Divisão do Trabalho na Programação

John Carmack

runtime code (núcleo do jogo)

John Romero

ferramentas de desenvolvimento

Jason Blochowiak

sistemas de gerenciamento

Desenvolvimento focado em eficiência

Organização direta e funcional

A divisão do trabalho destacava funções bem delimitadas entre programação principal, ferramentas e sistemas de suporte, com foco em eficiência e organização prática.

3.2 — Ferramentas de Desenvolvimento

TED5

editor de mapas (níveis e estruturas)

IGRAB

empacotamento de assets gráficos

MUSE

processamento de áudio

Geração de arquivos .h

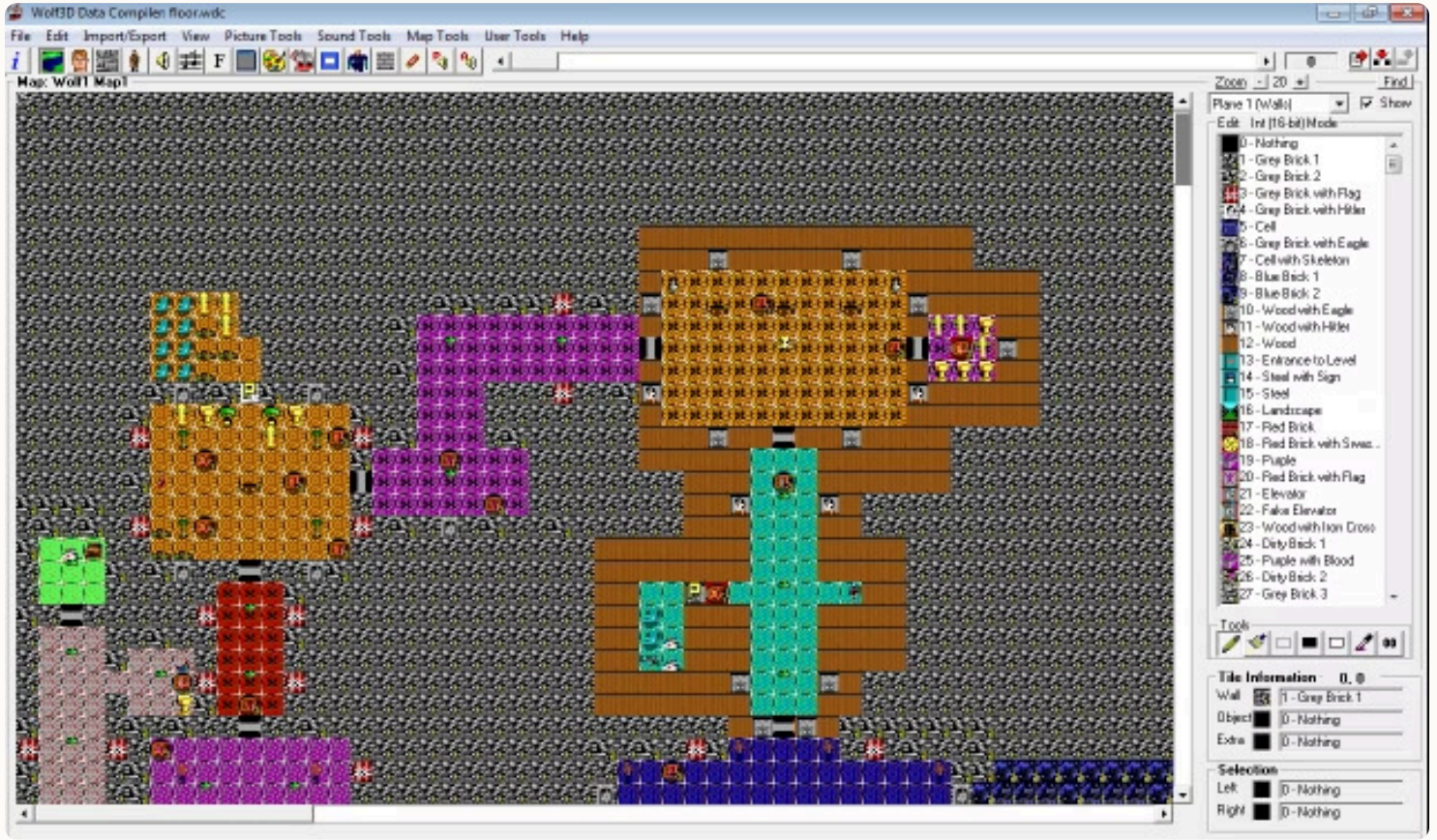
para integração com código

Managers

Input Manager, Sound Manager, Page Manager, User Manager

As ferramentas foram criadas internamente para dar suporte ao desenvolvimento e contornar limitações técnicas do hardware.

3.2 — Ferramentas de Desenvolvimento



Ferramenta TED5 para criação de Mapas

3.2 — Debugging em Baixo Nível

Uso de dois monitores (VGA + MDA)

Execução simultânea (jogo + debugger)

Turbo Debugger 386

Acesso direto à memória

Debug em tempo real

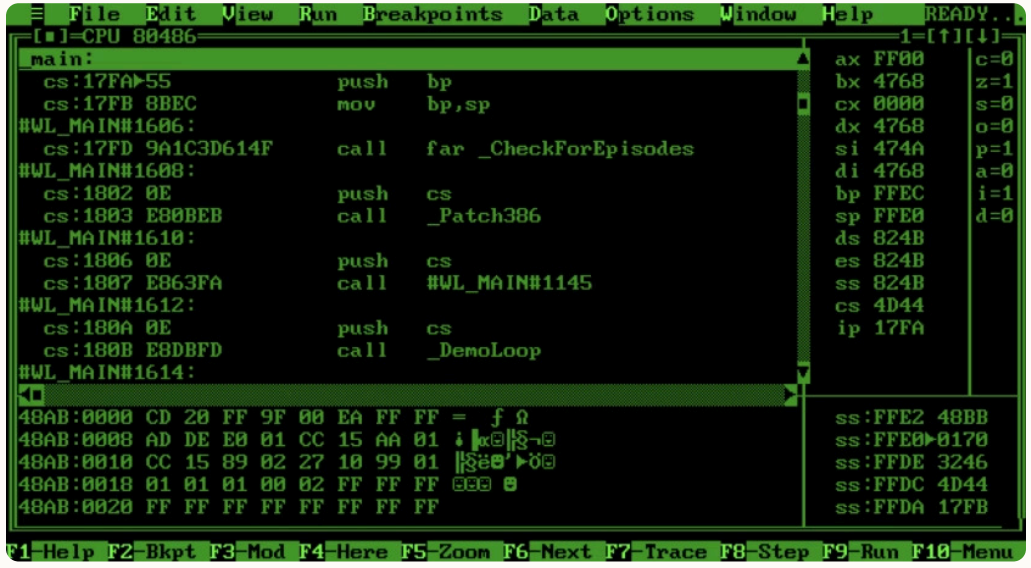
Destaque especial com endereços de memória:

VGA → 0xA0000

MDA → 0xB8000

O debugging foi feito diretamente no hardware, permitindo análise em tempo real do funcionamento do jogo.

3.2 — Debugging em Dois Monitores



3.2 — Filosofia de Programação

Baixo nível (low-level)

A programação em Wolfenstein 3D era feita de forma diretamente próxima ao hardware.

Sem abstrações modernas

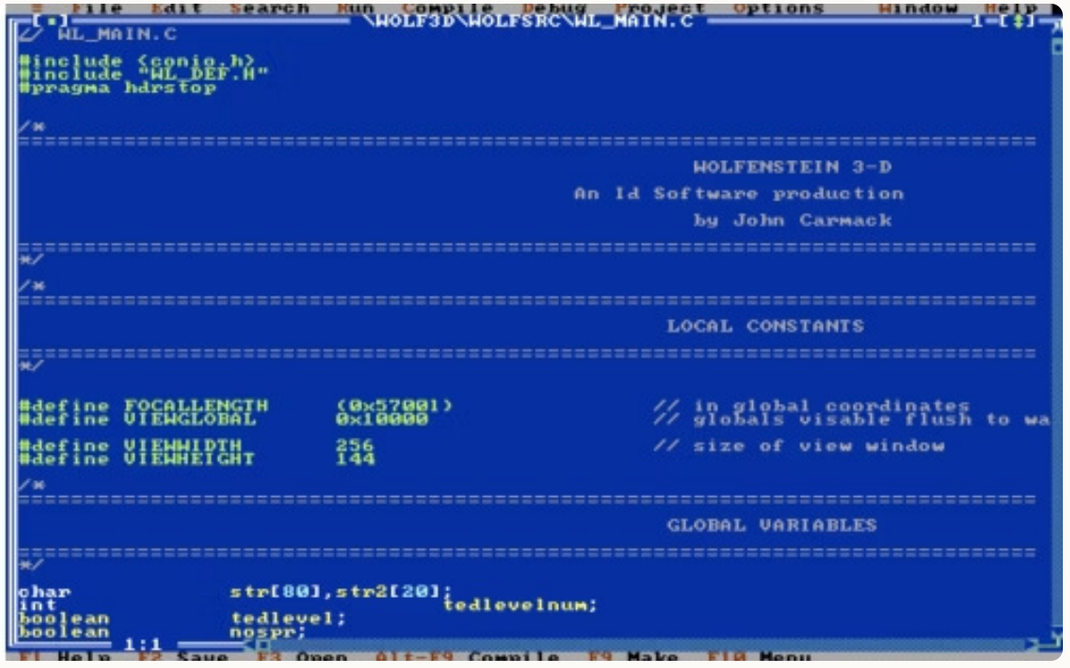
O desenvolvimento não dependia de abstrações de alto nível ou bibliotecas externas.

Controle direto do hardware

Os programadores lidavam diretamente com componentes como CPU e memória.

Trade Off da Visibilidade

Uso do dobro da resolução vertical do monitor prejudicava a legibilidade mas permitia enxergar o código de forma mais ampla.



3.3 — Criação dos Gráficos

Pixel art manual

Os gráficos eram desenhados pixel a pixel.

Deluxe Paint

O trabalho era feito com o Deluxe Paint.

Sem automação

O processo não contava com automação.

Trabalho artesanal

A criação exigia cuidado manual e atenção aos detalhes.

3.3 — Formato de Imagem

ILBM

O formato ILBM (InterLeaved BitMap) era um formato comum para gráficos em computadores Amiga, com suporte a paletas de cores e estrutura adequada para imagens indexadas.

Conversão de formatos

Os gráficos criados em ferramentas como o Deluxe Paint precisavam ser convertidos para um formato interno que o motor de Wolfenstein 3D pudesse ler e renderizar com rapidez.

Estrutura específica

Para obter desempenho, os dados eram organizados em uma estrutura própria, com cores indexadas e disposição adequada ao acesso direto pela rotina de raycasting.

Preparação para o jogo

Essa etapa finalizava os assets para uso no jogo, incluindo a organização dos dados para leitura eficiente do disco e carregamento em memória.

3.3 — Limitações de Cores

256 cores

O jogo operava com uma paleta limitada a 256 cores, uma restrição comum para a época, que exigia criatividade e eficiência no uso de cada tom disponível.

Paleta fixa

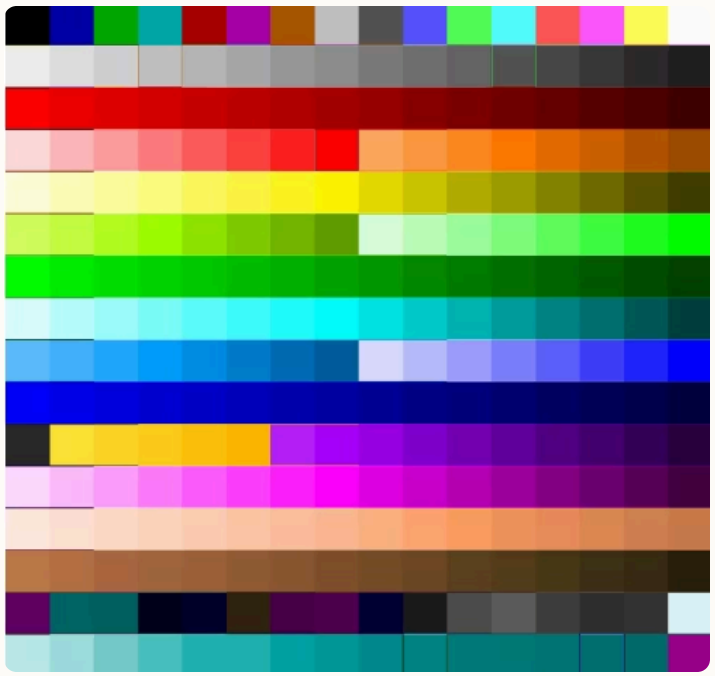
A paleta de cores era fixa e global para todo o jogo, o que significava que todos os gráficos e elementos visuais tinham que compartilhar o mesmo conjunto de 256 cores, sem possibilidade de paletas dinâmicas por objeto ou cena.

Color key e dithering

Para lidar com transparência e variações visuais, o livro descreve o uso de uma cor específica como *color key* (chave de cor) e técnicas como pontilhismo/dithering, que ajudavam a simular combinações de cores e efeitos visuais dentro da paleta limitada.

Ajustes manuais

Devido às severas limitações, a criação de cores e sombreamento era um processo altamente manual, exigindo que os artistas fizessem ajustes pixel a pixel para garantir que os gráficos se apresentassem da melhor forma dentro da paleta restrita.



3.3 — Sprites e Texturas

Sprites de inimigos

Sprites usados para representar inimigos no jogo.

Texturas de parede

Texturas aplicadas às paredes e superfícies do cenário.

HUD

Elementos da interface do jogador, como vida, armas e munição.

Organização visual

Estruturação dos assets visuais para manter a apresentação do jogo clara.

3.4 — Pipeline de Assets

O processo de transformar arte bruta em elementos prontos para o jogo seguia um fluxo organizado.

1

Criação

Definição e design dos assets visuais e sonoros.

2

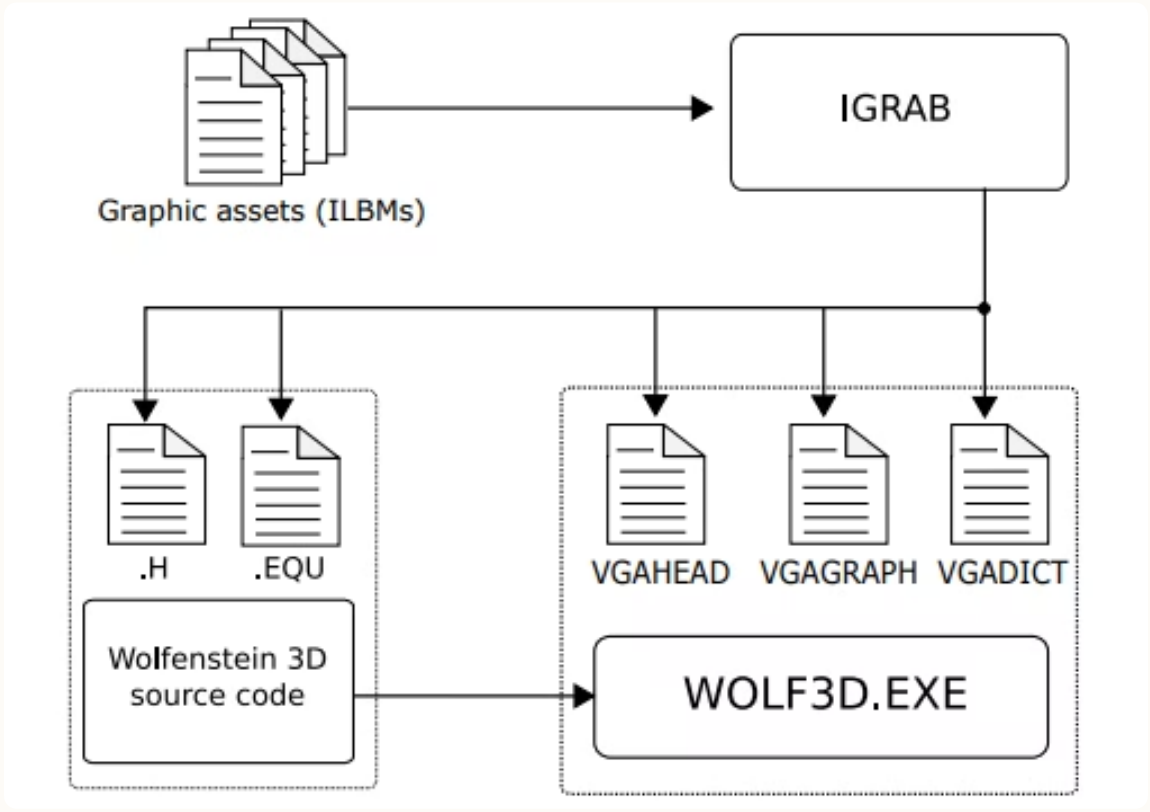
Conversão

Processamento prévio dos assets para otimização e adequação ao motor do jogo.

3

Uso

Integração e renderização dos assets dentro do ambiente de jogo.



3.4 — Ferramenta IGRAB

Empacotamento de assets

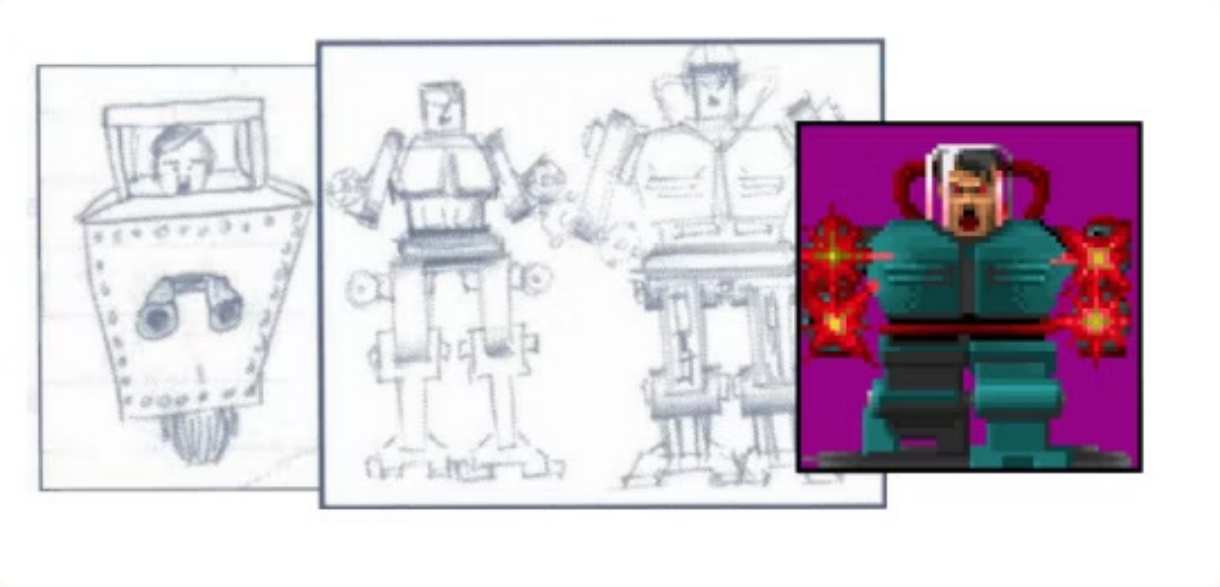
Agrupa múltiplos arquivos em um único pacote.

Conversão para formato interno

Converte os dados para o formato interno usado pelo motor.

Preparação para engine

Organiza os dados para uso pela engine.



3.4 — Sistema de Arquivos

1

VGAHEAD → índice

Arquivo de índice dos assets

2

VGAGRAPH → dados

Arquivo com os dados brutos dos gráficos

3

VGADICT → compressão

Dicionário usado na descompactação Huffman

4

Acesso por ID

Permite localizar rapidamente os assets pelo identificador

3.4 — Compressão e Performance

Huffman

A codificação Huffman era usada para comprimir dados dos ativos do jogo.

Redução de memória

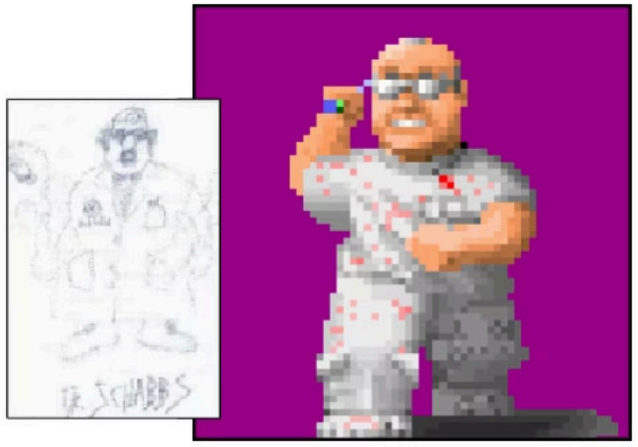
A compressão ajudava a diminuir o espaço necessário para armazenar os ativos.

Acesso rápido

O sistema permitia localizar e recuperar os dados com rapidez.

Melhor desempenho

Com menos dados para ler e processar, o jogo podia operar de forma mais eficiente.



Obrigado

Perguntas?

Limitações como catalisador para inovação

Sem GPU, sem FPU, mas com criatividade e otimização extrema, criaram o primeiro FPS da história.