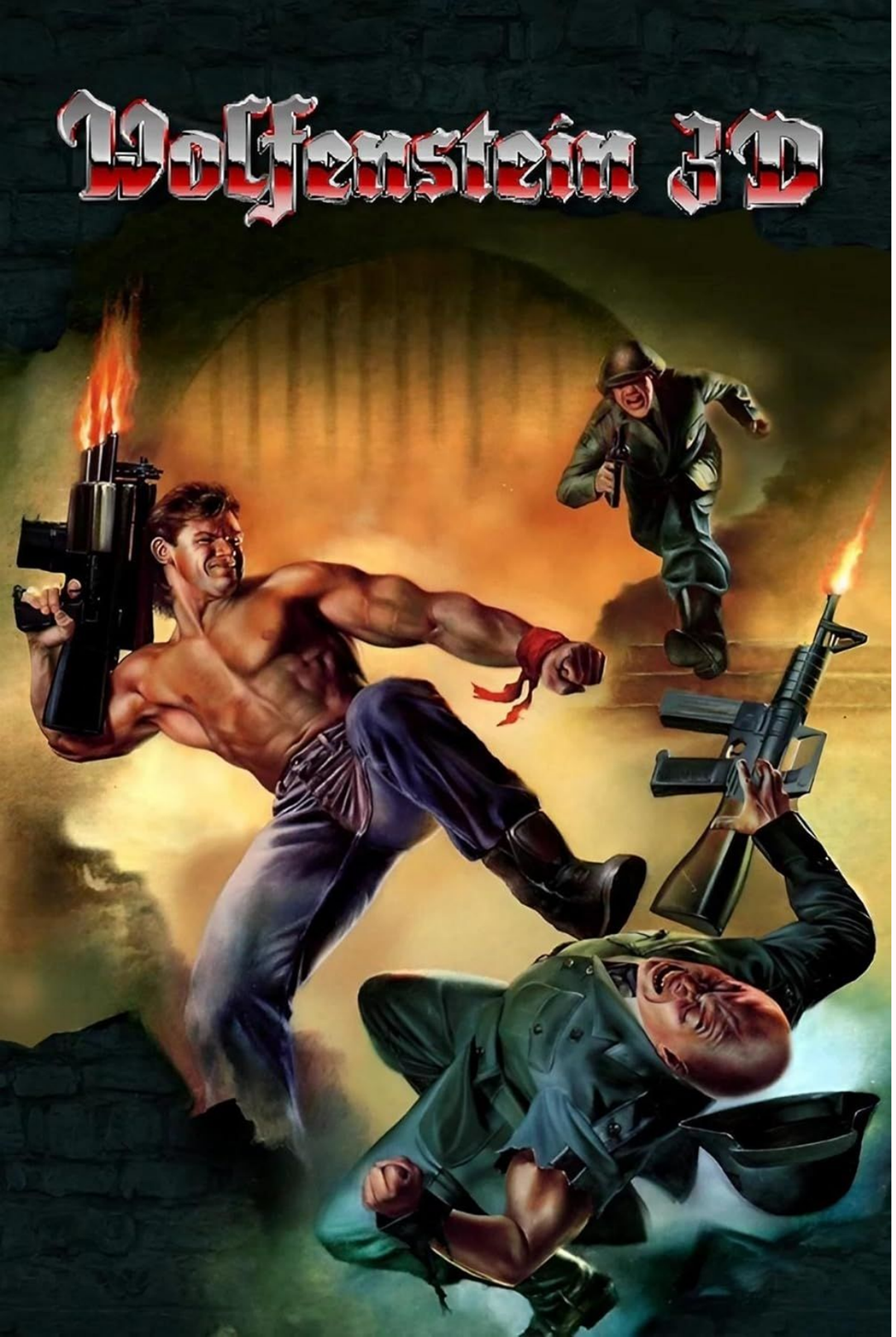


Como Wolfenstein 3D Renderizava seus Gráficos

Entendendo o motor gráfico que definiu o FPS moderno.

Iremos do menu 2D ao raycasting 3D, passando pelas otimizações da placa VGA.

SEMINÁRIO DE ARQUITETURA DE COMPUTADORES II

Arthur Schwambach | Elissa Brunelli | Christiano Rangel

Eduardo Rodrigues | Rhayssa dos Santos

Os Dois Renderizadores

Wolfenstein 3D operava em duas fases gráficas distintas, cada uma com desafios e técnicas próprias.

🖥️ Renderizador 2D — Menus

Telas de configuração, carregamento e opções. Fundo vermelho, imagens estáticas e textos carregados do disco.

🎮 Renderizador 3D — Ação

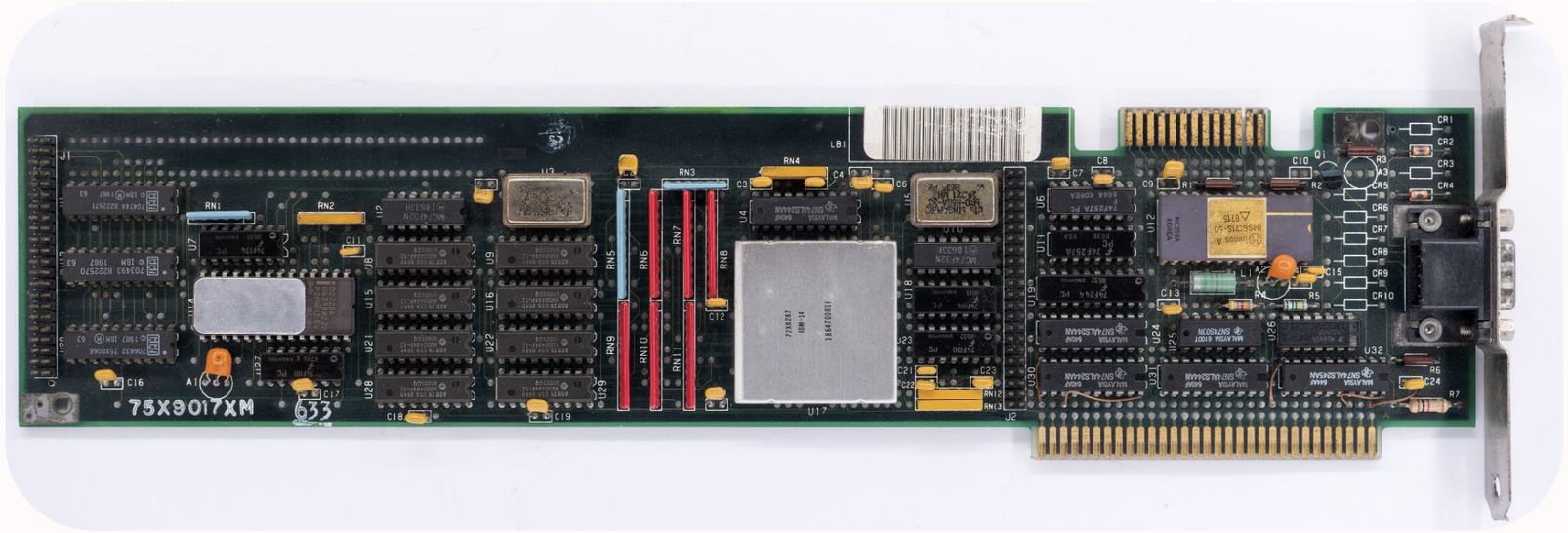
Corredores, paredes, inimigos e a arma do jogador. Raycasting em tempo real com ilusão de profundidade.

O Truque dos Bancos de Memória VGA

O Problema

A tela tinha **320×200 pixels** — 64.000 pixels.

Preencher o fundo vermelho do menu pixel por pixel exigiria 64 mil escritas na memória.



A Solução

A VGA tinha **quatro bancos de memória**.

Configurando uma máscara, o jogo escrevia em todos simultaneamente.

Com registradores de 16 bits:

8 pixels por operação.

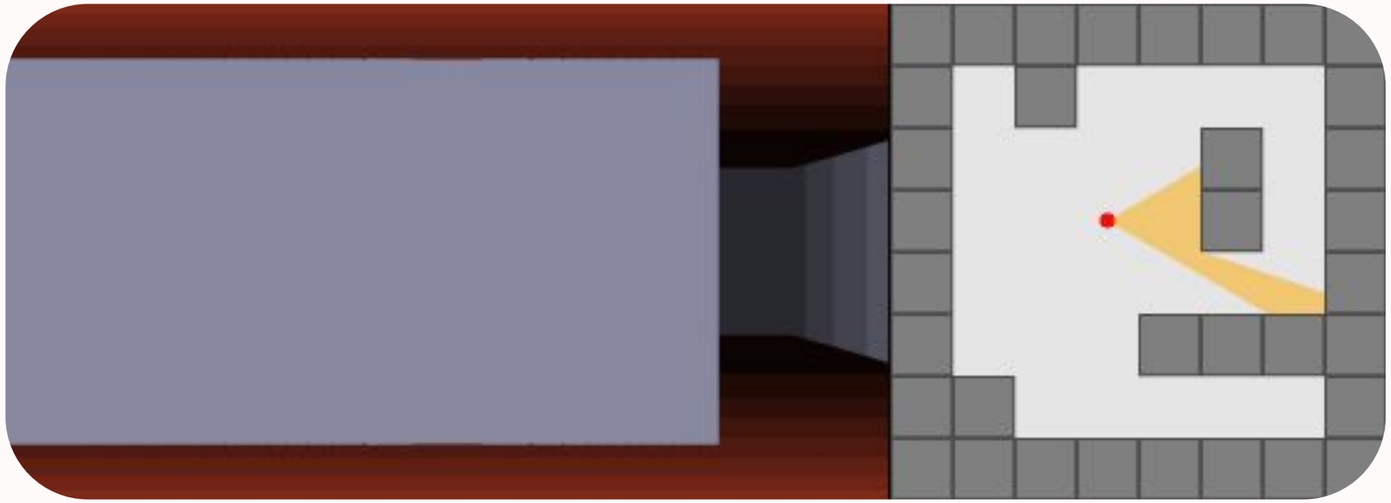
Resultado: de 64.000 escritas para apenas **8.000 operações**

Uma redução de 87,5%.

Raycasting: A Ilusão do 3D

O motor não renderizava polígonos 3D.

Ele lançava **~320 raios** — um para cada coluna da tela — a partir do ponto de vista do jogador. Cada raio avançava até encontrar uma parede no mapa 2D.



Eficiência

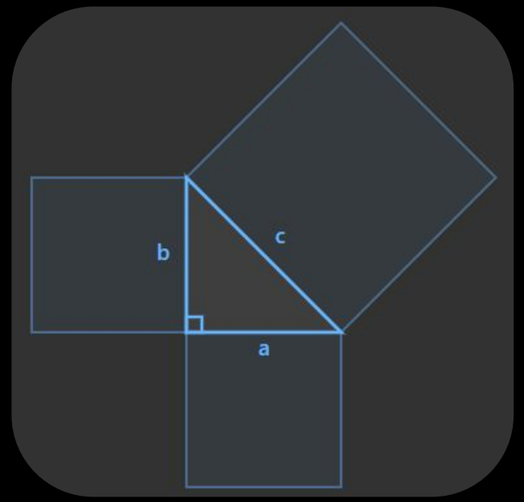
Sem polígonos complexos. Apenas interseções de raios com paredes em um mapa 2D, gerando aparência 3D convincente.

A Distância

Quanto mais perto a parede, maior a coluna desenhada.

A fórmula:

altura = escala ÷ distância.



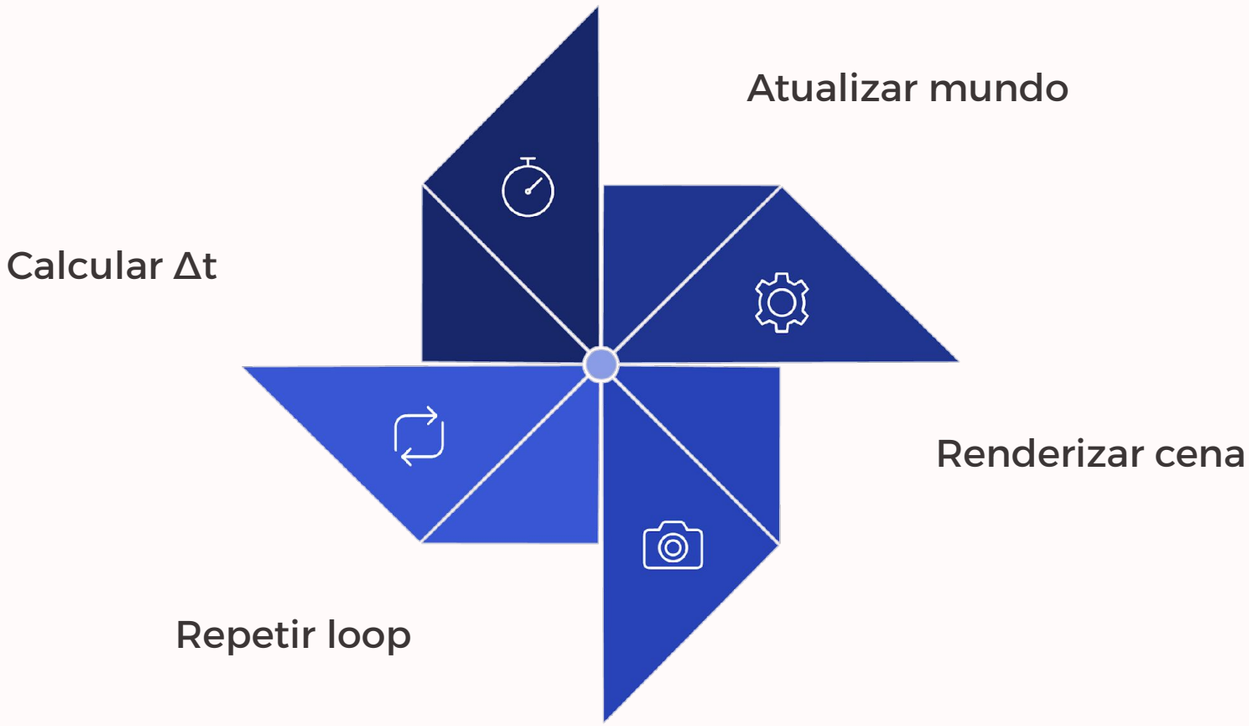
Portas no Mundo

Portas faziam parte do mundo sólido. O raycaster verificava se estavam abertas ou fechadas e decidia se o raio parava ou atravessava.

O Loop de Frames e o Problema do Tempo Variável

O loop clássico dos anos 1990 tinha um problema crítico: **o tempo de cada frame variava conforme o hardware.**

Um 486 renderizava mais rápido que um 286, tornando o jogo não determinístico.



Demos gravadas ficavam fora de sincronia em máquinas diferentes.
Os comandos do jogador podiam ser executados em momentos errados.
O Doom, lançado depois, resolveu isso separando atualização do mundo da renderização.

As 5 Etapas de Renderização 3D

01

Limpar o Framebuffer

Teto e chão desenhados com cores sólidas na área 3D.

02

Desenhar as Paredes

Raios lançados da posição do jogador; colunas texturizadas com altura proporcional à distância.

03

Desenhar os Sprites

Inimigos, barris e lâmpadas posicionados e recortados quando atrás de paredes.

04

Desenhar a Arma

Arma sobreposta no centro inferior da tela — sem depth buffer, por limitações de acesso à memória.

05

Trocar os Buffers

Controlador de vídeo instruído a exibir a nova página na próxima sincronização vertical.

HUD: Interface Desenhada Uma Vez

Antes da ação começar, o motor preparava os elementos fixos da interface: fundo verde, barra azul e textos como **LEVEL**, **SCORE**, **LIVES**, **HEALTH** e **AMMO**.

O HUD era desenhado **apenas uma vez** no início da fase, mas precisava existir nas **três páginas de vídeo** usadas pelo sistema de buffer.

Partes dinâmicas como vida, munição, rosto do personagem eram atualizadas separadamente durante a partida.

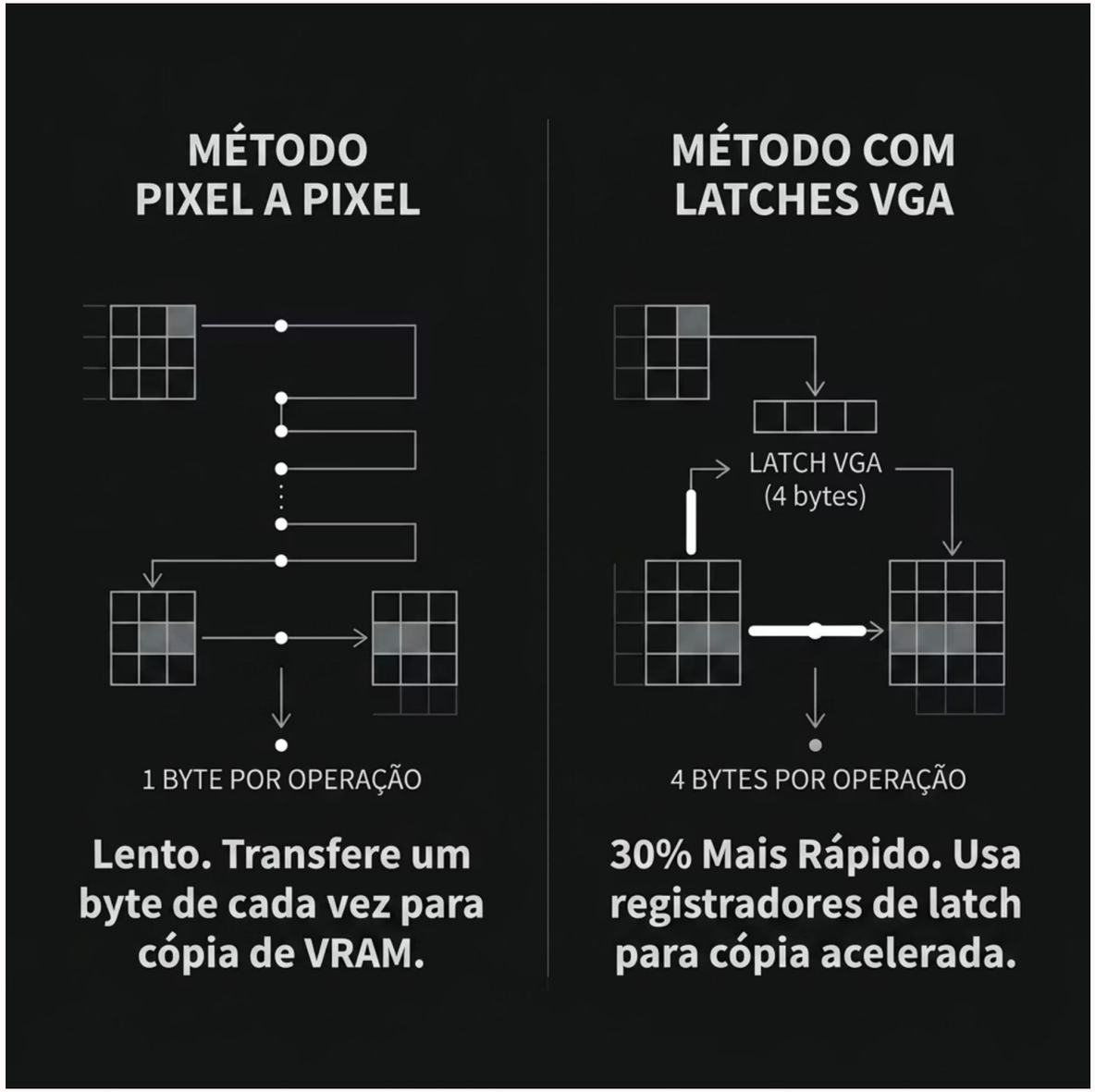
Latches VGA: Copiando Dados em Alta Velocidade

O que são Latches?

Circuitos da VGA que guardavam valores lidos da memória de vídeo. Originalmente projetados para outro modo gráfico, foram **reaproveitados no Mode-Y** pelos programadores do Wolfenstein 3D.

Como eram usados?

Permitiam copiar dados de uma região da VRAM para outra transferindo **quatro bytes de uma vez**, acelerando drasticamente a atualização dos elementos do HUD.



Imagens Entrelaçadas e Limpeza Otimizada

Armazenamento Entrelaçado

Imagens eram separadas por banco de memória.

Cada banco era carregado rapidamente com operações de cópia em bloco, evitando acessos sequenciais lentos.

Limpeza da Área 3D

A área de renderização (304×152 pixels) era limpa a cada frame com as cores de teto e chão.

A instrução **REP STOSW** e a máscara dos bancos VGA reduziam drasticamente o número de operações.

Ganho de 30%

A combinação dessas técnicas resultava em cerca de **30% de aumento de velocidade** na atualização da interface.

Era essencial para manter o framerate estável.

Conclusão: Engenharia no Limite do Hardware

Wolfenstein 3D não impressionava apenas pelo visual!

Wolfestein 3D era um **marco de otimização de baixo nível!**

→ Raycasting inteligente

Ilusão 3D convincente sem polígonos complexos, usando apenas interseções em mapa 2D.

→ Manipulação direta da VGA

Bancos de memória, latches e máscaras reduziram operações de escrita em até 87,5%.

→ Arquitetura em camadas

Separar menu 2D, renderização 3D e HUD permitiu otimizações específicas para cada fase.