Análise Técnica de Wolfenstein 3D

André Luiz Gomes dos Santos Filho
Daniel Carvalho Climaco
Davi André de Carvalho Pereira
Miguel Büge Paganini
Miguel Saiter Padovan

Universidade Vila Velha (UVV)

07 de Maio de 2026

Resumo

Este artigo apresenta um resumo analítico das técnicas de renderização abordadas nas seções 4.6 a 4.7.4 sobre o funcionamento interno do jogo Wolfenstein 3D. O foco central é demonstrar como o motor do jogo transita de um menu em 2D altamente otimizado para a fase de ação em 3D baseada em raycasting. A partir da análise do uso inteligente da memória VGA e do ciclo de vida dos quadros, discute-se como os desenvolvedores contornaram as severas limitações de hardware da época para criar uma experiência fluida.

Sumário

1 Introdução

O desenvolvimento de jogos no início dos anos 90 exigia criatividade para superar gargalos de processamento e memória. Em Wolfenstein 3D, a execução do jogo é dividida em duas fases principais: a configuração inicial através de um renderizador 2D e a jogabilidade em si, controlada pelo renderizador 3D.

Compreender essas etapas é fundamental para estudantes de computação, pois elas ilustram a aplicação prática de manipulação de memória de vídeo e algoritmos de otimização de renderização que formaram a base dos motores gráficos modernos.

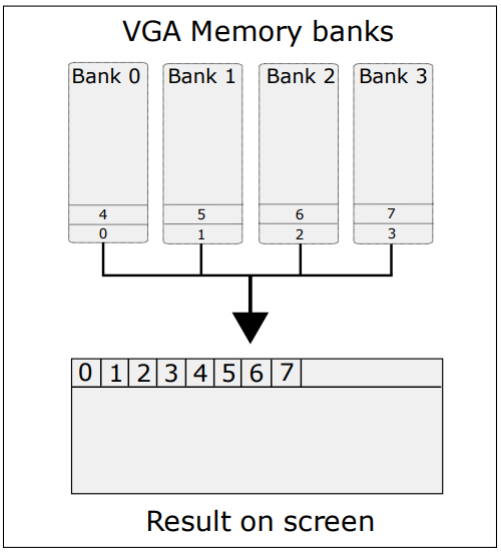
2 A Fase de Menu: Renderizador 2D

A fase inicial do jogo foca na interface de usuário. O maior gargalo técnico era a limpeza e preenchimento da tela, que possui 64.000 pixels.

2.1 O Truque da Máscara de Bancos VGA

Para otimizar a escrita, o motor utiliza o controle de máscara de bancos (*bank mask*). Ao configurar a máscara para o valor 15 (8+4+2+1), o sistema escreve em todos os quatro bancos da VRAM simultaneamente.

Figura 1: Esquema de Bancos VGA: Escrita paralela em múltiplos bancos.



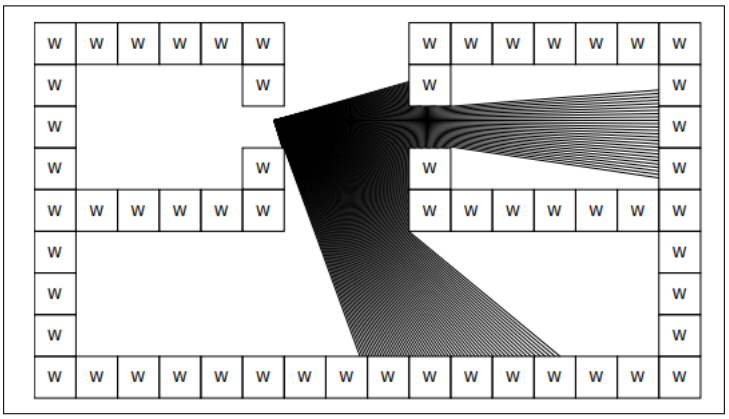
Isso permite que uma única operação de escrita na RAM preencha até 4 pixels na VRAM. Quando combinado com registradores de 16 bits, o motor consegue limpar a tela inteira com apenas 8.000 operações, otimizando drasticamente o processo.

3 O Renderizador 3D

O motor de Wolfenstein 3D utiliza um sistema de *Raycasting*. A ideia central é lançar um raio a partir do ponto de vista para cada coluna de pixels visível na tela.

A altura da coluna (h) desenhada na tela é baseada na distância (d) até a parede atingida pelo raio, calculada pela fórmula $h = \frac{X}{d}$, onde X é um fator de escala simples.

Figura 2: Processo de Raycasting: Lançamento de 320 raios para uma tela 320x200.



4 O Ciclo de Vida do Quadro

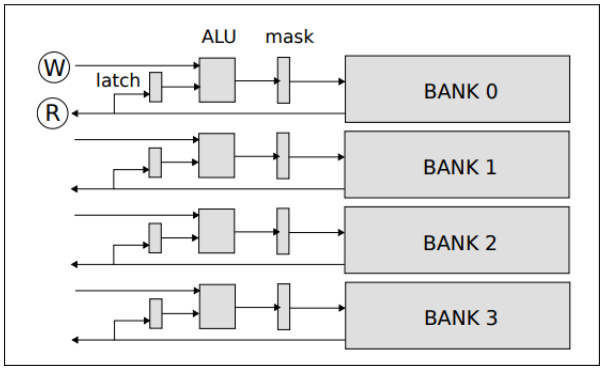
O motor original possuía uma falha onde cada fatia de tempo do jogo tinha uma duração diferente dependendo do tempo de renderização e atualização, o que tornava o jogo não-determinístico. Para desenhar uma cena 3D, o sistema envolve cinco estágios precisos:

1. **Limpeza:** O framebuffer é limpo desenhando o teto e o chão com cores sólidas.
2. **Paredes:** Para cada coluna, um raio é lançado e uma coluna texturizada é desenhada com altura inversamente proporcional à distância.
3. **Sprites:** Os sprites (inimigos, lâmpadas, barris) são desenhados.
4. **Arma:** A arma é desenhada em primeiro plano na tela.
5. **Flip de Buffers:** O controlador CRT é instruído a usar o framebuffer composto na próxima sincronização vertical.

5 Configuração 3D e HUD (Seção 4.7.3)

Antes de começar a desenhar os quadros, o renderizador 3D configura a VRAM com elementos estáticos formando o HUD. Para as atualizações dinâmicas na interface, os programadores exploraram os *latches* presentes no Modo 12h da placa VGA.

Figura 3: Arquitetura de Latches: Reutilização de circuitos para transferência rápida VRAM-VRAM.



A arquitetura da placa VGA possuía um *latch* na frente de uma ALU configurável, permitindo preencher quatro *latches* de uma vez e escrever quatro bytes com uma única operação na RAM. Esse sistema possibilita transferências da VRAM para a VRAM de 4 bytes por vez, resultando em um aumento geral de velocidade de renderização em 30%.

6 Limpando a Tela (Seção 4.7.4)

No início de um quadro, o motor limpa a área 3D com as cores do teto e do chão. O código utiliza a mesma técnica da máscara de bancos da fase 2D, permitindo escrever 8 pixels com uma única instrução ao usar registradores de 16 bits.

```
void VGAClearScreen(void) {
    asm mov dx, SC_INDEX
    asm mov ax, SC_MAPMASK + 15 * 256
    asm out dx, ax
    asm mov es, [screenseg]
    asm mov di, [bufferofs]
    asm mov ax, [ceiling]
    toploop:
        asm mov cl, bl
        asm rep stosw
        asm add di, dx
        asm dec bh
        asm jnz toploop
}
```

O uso muito bem arquitetado da instrução de assembly `REP STOSW` faz com que a limpeza total do quadro 3D (composto por 304x152 pixels) requeira apenas 779 instruções do processador.

7 Conclusão

O desenvolvimento de Wolfenstein 3D exigiu superar limitações drásticas de hardware através do uso criativo da memória VRAM e de otimizações de baixo nível em Assembly. A união das técnicas de *Raycasting* com o controle absoluto de ferramentas como a máscara de bancos e a arquitetura de *latches* garantiu a taxa de quadros necessária para criar e popularizar o paradigma moderno de jogos de tiro em primeira pessoa