Wolfenstein 3D - Capítulo 4.5: Inicialização

Erlon Felipe Castiglioni Tristão Pinheiro
Gabriel Araújo Pieckhardt Romão
Guilherme Gasperazzo Zamprogno
Lucas Pizzol de Oliveira Morandi
Pablo Barcellos Soares

Universidade de Vila Velha

Abril de 2026

Resumo

Este trabalho aborda o processo de inicialização do motor do Wolfenstein 3D, destacando os desafios impostos pelo hardware limitado e heterogêneo dos PCs do início dos anos 1990. A tela de sign-on funcionava como um sistema de autodiagnóstico, e a inicialização envolvia configurar o modo gráfico VGA, carregar a paleta de cores e exibir a imagem inicial diretamente da memória. Além disso, são discutidas as limitações do modo gráfico padrão, a implementação de técnicas como o triple buffering, soluções engenhosas para evitar o tearing causado por limitações de escrita em registradores, e a criação de uma classificação indicativa própria (PC-13).

Sumário

1 Introdução

O processo de inicialização do motor de jogo representa um dos momentos mais críticos da execução, especialmente considerando as limitações impostas pelo hardware da época. Conforme discutido no Capítulo 2 (“Hardware”), os computadores pessoais do início dos anos 1990 apresentavam um ambiente altamente heterogêneo, com variações significativas em termos de drivers carregados, dispositivos instalados e disponibilidade de memória. Nesse contexto, o motor precisava realizar uma série de verificações antes mesmo de iniciar a execução propriamente dita.

2 Tela de Inicialização

A chamada signon screen (tela de inicialização) desempenhava o papel de um sistema de autodiagnóstico. Sua função principal era identificar os recursos disponíveis no sistema do usuário e determinar se o jogo poderia ser executado adequadamente. Isso incluía a detecção de dispositivos como mouse, joystick e placas de som, bem como a verificação da quantidade de memória disponível.

Figura 1: Interface da Tela de Inicialização

A métrica mais relevante exibida nessa tela era o valor identificado como **“MAIN”**, que representava a quantidade de memória convencional disponível. Em sistemas baseados em DOS, os programas tinham acesso a apenas **640 KiB de RAM convencional**, uma limitação estrutural da arquitetura herdada dos primeiros PCs. Cada driver carregado pelo sistema consumia parte desse espaço, reduzindo a memória disponível para o jogo.
Caso o sistema não dispusesse de memória suficiente para carregar o executável, o usuário recebia a mensagem:

Out of memory

Para funcionar corretamente, **Wolfenstein 3D** exigia no mínimo **320 KiB de memória convencional livre**. Diante das frequentes dificuldades enfrentadas pelos usuários, John Romero publicou notas explicativas, como o texto “The 640KB Barrier”, com o objetivo de esclarecer essas limitações e reduzir a necessidade de suporte técnico.

2.1 Implementação Técnica

Durante a exibição da tela de signon, o motor ainda não dispõe de um sistema de arquivos ativo. O único componente funcional é o gerenciador de memória. Por essa razão, tanto a paleta de cores quanto a imagem da tela são incorporadas diretamente ao executável. Essa abordagem garante que os dados estejam carregados na memória assim que o programa é iniciado pelo DOS. A rotina de inicialização executa então os seguintes passos:

- Configuração do modo gráfico VGA;
- Carregamento da paleta de cores;
- Transferência da imagem da tela de signon da RAM para a memória de vídeo (VRAM);
- Exibição de indicadores visuais baseados nos recursos detectados.

Após a exibição, a memória ocupada pela imagem, aproximadamente **64.000 bytes**, é explicitamente liberada, permitindo sua reutilização durante a execução do jogo. Essa prática evidencia o nível de otimização necessário para operar dentro das restrições severas de memória da plataforma.

3 Resolvendo o Problema do VGA

O problema principal do VGA na época era a ausência de double buffering, ou seja, existia apenas um framebuffer disponível. Isso causava efeitos visuais indesejados, como o tearing (quebra de imagem), além de limitar o desempenho dos jogos. No modo padrão utilizado, conhecido como **Mode 13h**, a resolução era de 320x200 com 256 cores, e ele utilizava um sistema chamado **Chain-4**. Esse modo era fácil de programar por usar memória linear, porém apresentava uma grande desvantagem: desperdiçava cerca de 75% da memória de vídeo (VRAM) e não permitia o uso de técnicas mais avançadas, como múltiplos buffers.

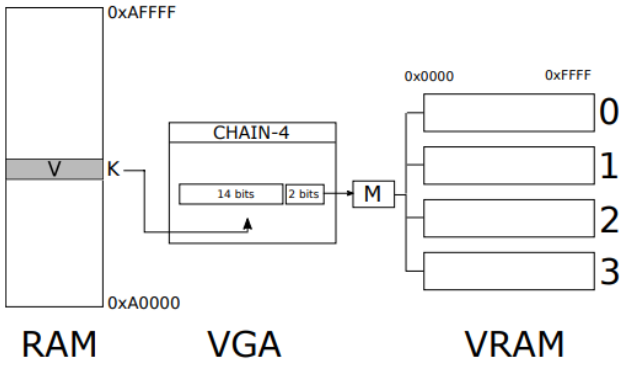

Figura 2: Esquema de operações entre RAM e VRAM do Chain-4

Vantagem:

- Fácil de programar (memória linear)

Problema:

- Desperdiça 75% da VRAM
- Não permite múltiplos buffers

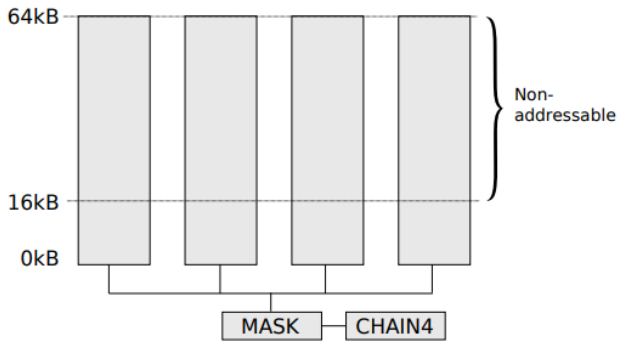

Figura 3: Desperdício de VRAM no Chain-4

Por isso, o time utilizou a técnica **"Mode-Y"** que era parecida com o **"Mode-X"** que desativava o chip Chain-4 e permitia acesso total aos 256Kbit de ram aos programadores, a diferença é que ela não obrigava eles a usarem a resolução de 320x240px que aumentava a quantidade de pixels que tinha como consequência o aumento do trabalho dos designers e processamento. Essas escolhas resolveram um grande problema, mas introduziu 2 menores.

3.1 Problemas Introduzidos

3.1.1 Problema de Memória

Sem o chip Chain-4 os desenvolvedores tinham que selecionar o banco para escreve, na função eles podiam escolher entre 4 planos, isso por que a CPU só suportava até 64KB de memória, então os 256KB do VGA era divido entre os 4. O problema acontecia quando se usava essa função quando tinham que limpar a tela por exemplo onde tinham que ser feita centenas de alterações do banco utilizado tornando o processo extremamente lento, já que mudaram algo que era feito em hardware para algo feito em software.
A solução para isso foi desenhar verticalmente primeiro para minimizar o número de trocas de banco, assim deixando o código duas vezes mais rápido, tudo no jogo foi desenhado dessa forma (paredes, sprites, menus, itens, etc).

3.1.2 Problema de Tearing

Outro problema gerado foi o Tearing que causa cortes na imagem, isso acontecia pois o monitor desenhava mais rápido que a CPU escrevia, então quando o processador estava desenhando uma imagem nova enquanto o monitor está lendo a antiga, com isso o jogador via metade do frame 1 e metade do frame 2, parecendo que "rasgou" a imagem no meio.
Para solucionar o problema os devs usaram a sincronização vertical (vsinc): o computador esperava o monitor terminar de desenhar para só então escrever o novo mas para sprites pesados causava lentidão, mas por causo do "Mode-Y" eles conseguiam usar o **Triple Buffering**, ou seja: 3 páginas na memória da VGA, onde o motor fica exibindo uma delas, outra pronta, esperando sua vez e a última sendo escrita pela CPU para o próximo frame.

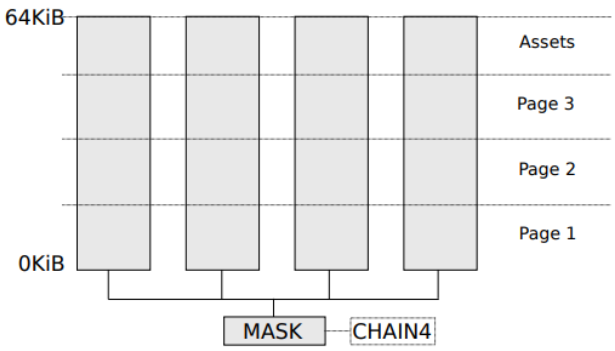

Figura 4: Esquema de operações entre RAM e VRAM do Triple Buffering

3.1.3 CRT Controller

Esse problema acontecia durante a troca de quadros de imagem na tela, onde duas imagens diferentes e desalinhadas eram apresentadas simultaneamente. Isso ocorria porque o endereço inicial do controlador de vídeo (CRTC) era composto por 16 bits, enquanto a instrução *out* da linguagem *Assembly*, utilizada para a comunicação com o hardware, era capaz de escrever apenas 8 bits de cada vez.

Figura 5: Esquema de operações entre RAM e VRAM do Chain-4

Sendo assim, a atualização do endereço era dividido em dois passos:

- Gravação do byte superior no registrador;
- Gravação do byte inferior no registrador.

Na VRAM, se o endereço da página 1 fosse 0x0000, e o da página 2 fosse 0x3E80, para realizar a operação de troca da página 1 para a página 2, por exemplo, seria necessário atualizar o *high byte* da página de 0x00 para 0x3E, e o *low byte* de 0x00 para 0x80. Como essa atualização era dividida em duas etapas, o endereço lido em algum instante poderia ser uma mistura das duas páginas, como 0x3E00, apresentando uma imagem desalinhada.
Para contornar essa situação, os desenvolvedores definiram **208** como o valor da altura do framebuffer no lugar de usar 200. Como a imagem continha 200 pixels de altura, restava um preenchimento intencional entre as páginas.

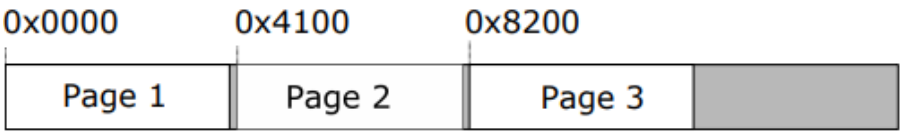

Figura 6: Padding entre as páginas para evitar o glitch causado pela atualização do CRTC

Com essas 8 linhas “invisíveis” o endereço inicial de todas as páginas se tornava um múltiplo exato de 256, apresentando a seguinte estrutura:

- **Página 0**: 0x0000
- **Página 1**: 0x4100
- **Página 2**: 0x8200

Dessa forma apenas o *high byte* do endereço das páginas eram diferentes, sendo possível atualizar as páginas utilizando somente 8 bits, transformando a troca de páginas em uma operação atômica.

4 Carnificina Profunda

O jogo foi lançado em 1991, quando ainda não existia o **Entertainment Software Rating Board (ESRB)**, órgão responsável pela classificação indicativa para jogos, que veio a surgir alguns anos depois por conta das críticas a violência excessiva em jogos como Doom. Nesse contexto, os desenvolvedores decidiram criar e incluir a própria tela de classificação indicativa, exibida após a tela inicial do jogo, que continha a classificação **PC-13**, uma cópia satirizada da sigla **PG-13**, utilizada em filmes. O jogo se declarava como voluntariamente classificado e atribuía à sigla PC-13 o significado **”Profound Carnage”** (Carnificina Profunda), brincando com a violência extrema presente em seu jogo.

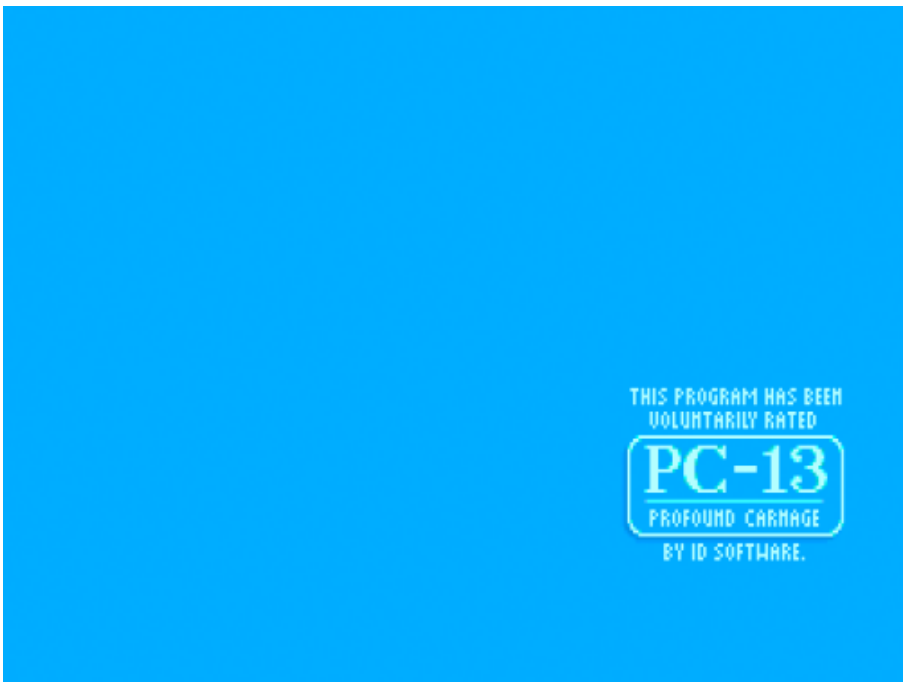

Figura 7: Classificação indicativa PC-13 exibida após a tela de inicialização

5 Conclusão

O processo de inicialização do motor demonstra o alto nível de criatividade e otimização exigido dos desenvolvedores da época. Diante de limitações severas de hardware e software, soluções inovadoras, como

o uso do Mode-Y, triple buffering e ajustes de memória, foram fundamentais para garantir desempenho e qualidade visual. Isso evidencia como o desenvolvimento de jogos nesse período era profundamente ligado ao entendimento técnico do hardware, exigindo soluções de baixo nível altamente eficientes.

6 Referências Bibliográficas

SANGLARD, Fabien. **Game Engine Black Book Wolfenstein 3D**: v2.2. [S. l.: s. n.], 2022.