Resumo de Estudo

Capítulo 4 — Startup

Carolina Rocio, Enzo Ofrante, Isabela Pinheiro, Kauã Araujo e Lucas Cunha

Maio de 2026

Resumo

Este material apresenta um resumo de estudo sobre a Seção 4.5 do Capítulo 4, que trata da etapa de inicialização do motor do *Wolfenstein 3D*. O foco está em entender como o jogo preparava o computador antes da execução, verificava as condições do sistema, configurava o modo gráfico VGA e contornava limitações importantes do hardware da época. O objetivo não é decorar detalhes de código, mas compreender o processo geral: primeiro o jogo identifica os recursos disponíveis, depois ajusta a memória e o vídeo, e por fim prepara a tela para iniciar a experiência do jogador.

Sumário

1 Visão geral da matéria

A Seção 4.5 apresenta a fase de *startup*, ou seja, a inicialização do motor do jogo. Antes de o jogador acessar menus ou entrar na fase 3D, o programa precisava preparar o ambiente de execução.

Nos computadores pessoais do início dos anos 1990, não havia uma padronização tão simples como nos sistemas atuais. Cada máquina podia ter uma quantidade diferente de memória, drivers carregados, mouse, joystick, placa de som ou apenas o alto-falante interno do PC. Além disso, o sistema operacional DOS impunha várias limitações.

Por isso, o *Wolfenstein 3D* precisava realizar algumas tarefas logo no início:

- verificar os recursos disponíveis na máquina;
- identificar memória, dispositivos de entrada e placas de som;
- preparar a tela inicial de diagnóstico;
- configurar o VGA para o modo gráfico usado pelo jogo;
- resolver o problema da ausência de *double buffering*;
- preparar a troca de páginas de vídeo sem causar distorções na imagem.

A ideia principal da seção é mostrar que o jogo só funcionava bem porque seus programadores conheciam profundamente o hardware. Eles não dependiam apenas de comandos prontos: manipulavam diretamente registradores e memória de vídeo para extrair desempenho da máquina.

2 Startup

2.1 O que significa Startup

No contexto do capítulo, *startup* é a etapa inicial do programa. É o momento em que o jogo ainda não começou de verdade, mas o motor já está executando rotinas essenciais para deixar o sistema pronto.

Essa etapa funciona como uma preparação. O programa confere se há memória suficiente, identifica dispositivos, carrega informações básicas e configura o vídeo. Sem essa fase, o jogo poderia travar, apresentar erros visuais ou simplesmente não iniciar em muitas máquinas.

2.2 Por que essa etapa era necessária

Atualmente, boa parte das diferenças entre computadores é escondida pelo sistema operacional. Na época do DOS, isso não acontecia da mesma forma. O jogo precisava lidar diretamente com limitações como:

- pouca memória convencional disponível;
- presença ou ausência de drivers;
- diferentes placas de som;
- diferentes dispositivos de entrada;
- arquitetura complexa da memória VGA;
- impossibilidade de usar *double buffering* diretamente no modo gráfico padrão.

Portanto, o *startup* não era apenas uma tela de abertura. Ele fazia parte da estratégia técnica para garantir que o jogo pudesse rodar em computadores diferentes.

3 Signon

3.1 Função da tela Signon

A primeira parte importante da inicialização é a tela chamada *Signon*. Essa tela funcionava como um diagnóstico do sistema. Ela mostrava quais recursos o jogo conseguiu identificar antes de iniciar.

Entre as informações verificadas estavam:

- memória principal disponível;
- memória EMS;
- memória XMS;
- mouse;
- joystick;
- dispositivos de som.

O item mais importante era a memória principal, chamada de *MAIN*. Como o DOS trabalhava com memória convencional limitada, o jogo precisava saber se havia espaço suficiente para continuar.

3.2 A importância da memória principal

O *Wolfenstein 3D* precisava de uma quantidade mínima de memória convencional para funcionar. O problema era que, mesmo que o computador tivesse mais memória física instalada, o DOS e os drivers podiam reduzir bastante a memória realmente disponível para o programa.

Se não houvesse memória suficiente, o usuário poderia receber uma mensagem de erro indicando falta de memória. Por isso, a tela *Signon* era útil tanto para o jogador quanto para a própria id Software, pois ajudava a explicar problemas comuns de execução.

3.3 Por que a imagem ficava dentro do executável

Um detalhe importante é que, nesse momento inicial, vários sistemas do motor ainda não estavam carregados. O jogo ainda não podia depender normalmente do sistema de arquivos e dos gerenciadores internos.

Por isso, a imagem da tela *Signon* e a paleta de cores eram compiladas junto com o executável. Assim, quando o programa era carregado pelo DOS, esses dados já estavam disponíveis na memória.

Depois que a tela era exibida, o espaço ocupado por ela era liberado para que o jogo pudesse usar essa memória durante a execução.

3.4 Resumo da Signon

A tela *Signon* não era apenas visual. Ela tinha função técnica. Servia para verificar o computador, mostrar recursos detectados e ajudar a identificar se o jogo conseguiria rodar. Também mostra como a memória era um recurso crítico na época.

4 O problema do VGA

4.1 O problema inicial

Depois da tela *Signon*, o jogo precisava configurar o vídeo. O modo gráfico mais atraente era o modo 13h, pois oferecia resolução de 320x200 pixels com 256 cores. Para jogos da época, isso era uma boa combinação entre qualidade visual e desempenho.

O problema é que o modo 13h não resolvia bem a questão do *double buffering*. Sem isso, a imagem poderia sofrer *tearing*, que ocorre quando o monitor exibe parte de um quadro antigo e parte de um quadro novo ao mesmo tempo.

4.2 O que é o Chain-4

O modo 13h usava um mecanismo chamado *Chain-4*. Ele facilitava a programação porque fazia a memória de vídeo parecer linear para o programador. Em outras palavras, o desenvolvedor podia escrever na VRAM de forma mais simples, sem se preocupar tanto com os bancos internos.

Porém, essa facilidade tinha um custo: o *Chain-4* desperdiçava grande parte da memória de vídeo. A placa VGA possuía 256 KiB de VRAM, mas o modo 13h com *Chain-4* não permitia aproveitar essa memória da melhor forma.

4.3 A solução: desativar o Chain-4

Para resolver o problema, o motor do jogo desativava o *Chain-4*. Com isso, os programadores passavam a ter acesso mais direto aos bancos da VRAM.

Essa técnica permitia dividir a memória de vídeo em várias páginas. Na prática, o jogo conseguia usar três áreas diferentes para desenhar e exibir imagens. Isso possibilitava uma forma de *triple buffering*, evitando que a imagem fosse alterada enquanto ainda estava sendo exibida pelo monitor.

4.4 Mode-X e Mode-Y

A técnica de desativar o *Chain-4* foi popularizada por Michael Abrash com o chamado *Mode-X*. Esse modo permitia trabalhar com 320x240 pixels e pixels quadrados.

O *Wolfenstein 3D*, porém, usou uma variação conhecida como *Mode-Y*. Em vez de usar 320x240, o jogo manteve 320x200. Essa escolha tinha dois motivos principais:

- 320x200 possui menos pixels, então exige menos processamento;
- os artistas criavam os gráficos nessa mesma resolução, evitando problemas no fluxo de produção.

Assim, o jogo abriu mão dos pixels perfeitamente proporcionais para ganhar desempenho e manter compatibilidade com o processo de criação dos recursos gráficos.

5 Desenho vertical

5.1 Por que desenhar na vertical

Ao desativar o *Chain-4*, surgiu outro problema: o programador precisava selecionar manualmente o banco de memória VGA onde iria escrever. Essa seleção era feita por comandos

de entrada e saída, que eram lentos.

Se o jogo desenhasse pixel por pixel na horizontal, teria que trocar de banco muitas vezes. Isso reduziria muito o desempenho.

A solução foi desenhar verticalmente. Ao desenhar uma coluna inteira antes de mudar de banco, o motor reduzia a quantidade de trocas necessárias e aumentava a velocidade de renderização.

5.2 Impacto no motor gráfico

Essa decisão teve impacto direto na arquitetura do jogo. O *Wolfenstein 3D* já era baseado em *raycasting*, técnica que desenha a cena 3D coluna por coluna. Portanto, o desenho vertical combinava muito bem com a forma como o motor calculava paredes e objetos.

Essa é uma das ideias mais importantes da seção: a limitação do hardware não foi apenas contornada, ela influenciou o próprio projeto do motor gráfico.

6 Troca de páginas de vídeo

6.1 O problema da troca de tela

Com várias páginas de vídeo disponíveis, o jogo precisava alternar qual página seria exibida pelo monitor. Essa troca era feita configurando o controlador CRT para começar a leitura em outro endereço da VRAM.

O problema é que esse endereço tinha 16 bits, mas era atualizado em duas partes de 8 bits. Se o monitor começasse a ler a tela no meio dessa atualização, poderia pegar um endereço incompleto ou incorreto.

O resultado seria uma imagem distorcida, misturando partes de duas páginas diferentes.

6.2 A solução com alinhamento

A solução foi criar um pequeno espaçamento entre as páginas de vídeo. Embora a tela tivesse 200 linhas de altura, o jogo considerava 208 linhas ao organizar as páginas na memória.

Esse preenchimento fazia com que os endereços iniciais das páginas ficassem alinhados em múltiplos de 256. Assim, ao trocar de página, era necessário alterar apenas a parte alta do endereço.

Com isso, a troca de tela se tornava mais segura e evitava falhas visuais.

7 Profound Carnage

Depois da configuração inicial, o jogo exibia uma tela de classificação chamada *PC-13: Profound Carnage*. Ela imitava de forma bem-humorada a classificação PG-13 dos filmes.

Na época, ainda não existia o sistema ESRB de classificação de jogos, que só surgiria depois. Mesmo assim, a id Software colocou essa tela como uma brincadeira e também como aviso sobre o conteúdo violento do jogo.

Essa parte mostra o estilo irreverente da empresa, mas também marca o fim da etapa de inicialização. Depois disso, o jogo estava pronto para entrar nas próximas fases do motor.

8 Síntese final da matéria

A Seção 4.5 mostra que o início do *Wolfenstein 3D* envolvia muito mais do que carregar uma tela de abertura. O jogo precisava preparar a máquina, verificar memória, identificar dispositivos e configurar o vídeo de maneira muito específica.

Os principais pontos para estudar são:

- a tela *Signon* funcionava como diagnóstico do sistema;
- a memória convencional era limitada e precisava ser cuidadosamente controlada;
- o modo 13h era atraente por usar 320x200 e 256 cores;
- o *Chain-4* facilitava a programação, mas desperdiçava VRAM;
- desativar o *Chain-4* permitiu usar múltiplas páginas de vídeo;
- o jogo usou uma variação chamada *Mode-Y*;
- o desenho vertical reduzia trocas lentas entre bancos de memória;
- o alinhamento das páginas evitava distorções durante a troca de tela;
- a tela *Profound Carnage* encerrava a fase inicial com o estilo característico da id Software.

A principal conclusão é que o desempenho do jogo dependia diretamente do conhecimento do hardware. O motor não era rápido por acaso: ele foi construído para transformar limitações do VGA, da memória e do DOS em soluções práticas de programação.