

Arquitetura e Organização de Computadores II

Wolfenstein 3D



Grupo 1:
Caio Eduardo Zanotelli Facini de Athayde
Arthur Trevizani Buback
Gian Valério Zanatelli
Marcos Daniel Guasti Machado
Matheus Braga Cetrangolo

Capítulo 4.4 - Arquitetura

A arquitetura do código-fonte de Wolfenstein 3D é dividida de forma clara para facilitar a interação com o hardware:

- Enquanto os arquivos que começam com o prefixo WL_ cuidam dos aspectos específicos do jogo
- Os subsistemas de baixo nível, identificados pelo prefixo ID_, abstraem a complexidade do hardware em "Managers" dedicados.
- Essa modularização foi essencial para que o motor fosse robusto e eficiente em um sistema operacional (DOS) que não oferecia suporte nativo a processos ou threads.

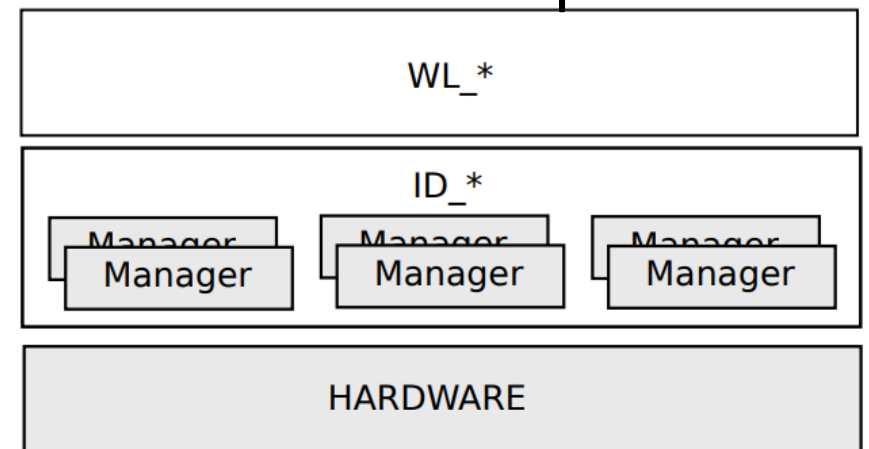


Figure 4.3: Wolfenstein 3D source code layers.



Figure 4.4: Architecture with engine and sub-systems (in white) connected to I/O (in gray).

Next to the hard drives (HDD) you can see the assets packed as described in Chapter 3 Team.

Capítulo 4.4.1 – Memory Manager - MM

O motor não utiliza a função padrão malloc do C, pois ela causa fragmentação excessiva na memória convencional. Em vez disso, a id Software implementou seu próprio MM, que gerencia blocos de RAM através de uma lista encadeada.

- Atributos de Bloco: Os blocos podem ser marcados como LOCKBIT (impedindo que sejam movidos durante a compactação) ou PURGEBITS (indicando prioridade de exclusão quando falta espaço).
- Sistema de Três Passos: Para alocar memória, o MM tenta primeiro encontrar um espaço vazio; se falhar, exclui blocos purgáveis; e em última instância, realiza a compactação total da RAM, movendo blocos e atualizando os ponteiros dos "donos" através do campo useptr.

Capítulo 4.4.2 – Page Manager - PM

Dedicado ao motor 3D, o PM gerencia a disponibilidade de texturas, sprites e sons na RAM. Inspirado no conceito de "paging" do Unix, ele utiliza páginas de tamanho fixo (4KiB) e um sistema de identificação por IDs de recursos.

- Hierarquia de Cache: O PM utiliza a RAM Convencional e a EMS como cache de Nível 1 (L1), e a memória XMS como Nível 2 (L2). O disco rígido (HDD) serve como o armazenamento final de onde as páginas são carregadas em caso de "miss".
- Política de Expulsão: Utiliza uma tag de número de quadro (frame number) para identificar e remover os recursos menos utilizados recentemente (LRU), evitando o fenômeno de thrashing (excesso de trocas de página que derruba a performance).

Capítulo 4.4.3 - 4.4.5 – Outros Subsistemas

- Video Manager (VL & VH): A camada VL abstrai a manipulação dos registros VGA via Assembly, enquanto a VH foca na renderização dos menus 2D.
- Cache Manager (CA): Responsável por carregar e descomprimir mapas e áudios, utilizando algoritmos como a compressão de Huffman.
- User Manager (US): Gerencia entradas do usuário e elementos da interface, embora contenha muito código legado de Catacomb 3-D que não é utilizado.

Metodologia

A abordagem metodológica central da arquitetura de Wolfenstein 3D é o **pragmatismo técnico**. A decisão de criar gerentes de memória dedicados foi justificada pela necessidade de controle total sobre a RAM fragmentada do PC. No caso do áudio, por exemplo, como as placas de som eram alimentadas por interrupções, o motor não podia permitir falhas de página durante a reprodução; por isso, todos os sons são carregados obrigatoriamente na memória convencional antes do início da ação.

Conclusão

A seção 4.4 do "Game Engine Black Book" revela que o sucesso de Wolfenstein 3D não dependeu apenas de gráficos inovadores, mas de uma engenharia de software sofisticada para a época. Ao implementar sistemas de paginação e gerenciamento dinâmico de memória em um ambiente de 16 bits, a id Software criou uma fundação sólida que permitiu ao jogo rodar em máquinas com apenas 640KiB de memória base, definindo padrões para o que viria a ser a arquitetura moderna de motores de jogos.