



cub3D

Meu primeiro RayCaster com a miniLibX

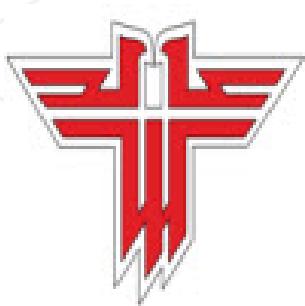
Preâmbulo: Este projeto é inspirado no jogo mundialmente famoso Wolfenstein 3D, que é considerado o primeiro FPS já criado. Ele permite que você explore ray-casting. Seu objetivo será fazer uma visão dinâmica dentro de um labirinto, onde você precisa encontrar seu caminho.



Versão: 11.0

Sumário

I	Prólogo	2
II	Objetivos	3
III	Instruções Comuns	4
IV	Instruções para IA	6
V	Parte obrigatória - cub3D	9
VI	Parte bônus	14
VII	Examples	15
VIII	Submissão e avaliação por pares	18



Capítulo I

Prólogo

Desenvolvido pela **Id Software**, liderado pelos mundialmente renomados John Carmack and John Romero, e publicado em 1992 pela **Apogee Software**, **Wolfenstein 3D** é o primeiro jogo “First Person Shooter” de verdade na história dos video games.



Figura I.1: John Romero (esquerda) and John Carmack (direira) posando para a posteridade.

Wolfenstein 3D é o ancestral de jogos como **Doom** (**Id Software**, 1993), **Doom II** (**Id Software**, 1994), **Duke Nukem 3D** (**3D Realm**, 1996) e **Quake** (**Id Software**, 1996), que são marcos eternos adicionais no mundo dos video games.

Agora, é sua vez de reviver a História...



O jogo **Wolfenstein 3D** originalmente é situado na Alemanha nazista, o que pode ser potencialmente perturbador. Fotos e a história deste jogo são trazidas aqui apenas por razões técnicas e de cultura pop/geek, já que o jogo foi considerado uma obra de arte para ambos.

Capítulo II

Objetivos

Os objetivos deste projeto são similares a todos os objetivos deste primeiro ano: rigor, uso de algoritmos básicos em C, pesquisa por informação, etc.

Como um projeto de design gráfico, cub3D permitirá que você melhore suas habilidades nestas áreas: janelas, cores, eventos, preencher formas, etc.

Concluindo, cub3D é um playground notável para explorar aplicações práticas e lúdicas da matemática sem precisar entender os detalhes.

Com a ajuda de vários documentos disponíveis na internet, você usará matemática como uma ferramenta para criar algoritmos elegantes e eficientes.



Se fizer sentido para você, você pode testar o jogo original antes de começar este projeto: <http://users.atw.hu/wolf3d/>

Capítulo III

Instruções Comuns

- Seu projeto deve ser escrito em C.
- Seu projeto deve ser escrito de acordo com a Norma. Se você tiver arquivos/funções bônus, eles serão incluídos na verificação da norma, e você receberá uma nota 0 se houver algum erro de norma.
- Suas funções não devem sair inesperadamente (segmentation fault, bus error, double free, etc.) exceto por comportamento indefinido. Se isso ocorrer, seu projeto será considerado não funcional e receberá uma nota 0 durante a avaliação.
- Toda memória alocada na heap deve ser liberada corretamente quando necessário. Vazamentos de memória não serão tolerados.
- Se o enunciado exigir, você deve submeter um `Makefile` que compile seus arquivos fonte para a saída requerida com as flags `-Wall`, `-Wextra`, e `-Werror`, usando `cc`. Adicionalmente, seu `Makefile` não deve realizar re-links desnecessários.
- Seu `Makefile` deve conter pelo menos as regras `$(NAME)`, `all`, `clean`, `fclean` e `re`.
- Para submeter bônus para seu projeto, você deve incluir uma regra `bonus` em seu `Makefile`, que adicionará todos os diversos headers, bibliotecas, ou funções que não são permitidas na parte principal do projeto. Os bônus devem ser colocados em arquivos `_bonus.{c/h}`, a menos que o enunciado especifique o contrário. A avaliação das partes obrigatórias e bônus é conduzida separadamente.
- Se seu projeto permitir que você use sua `libft`, você deve copiar suas fontes e seu `Makefile` associado para uma pasta `libft`. O `Makefile` do seu projeto deve compilar a biblioteca usando seu `Makefile`, e então compilar o projeto.
- Nós encorajamos você a criar programas de teste para seu projeto, mesmo que este trabalho **não precise ser submetido e não será avaliado**. Isso lhe dará a oportunidade de testar facilmente seu trabalho e o trabalho de seus colegas. Você achará esses testes especialmente úteis durante sua defesa. De fato, durante a defesa, você está livre para usar seus testes e/ou os testes do colega que você está avaliando.

- Submeta seu trabalho para o repositório Git designado. Somente o trabalho no repositório Git será avaliado. Se o Deepthought for designado para avaliar seu trabalho, isso ocorrerá após as avaliações de seus colegas. Se um erro acontecer em qualquer seção do seu trabalho durante a avaliação do Deepthought, a avaliação será interrompida.

Capítulo IV

Instruções para IA

● Contexto

Durante sua jornada de aprendizado, a IA pode auxiliar em diversas tarefas. Dedique tempo para explorar as várias capacidades das ferramentas de IA e como elas podem apoiar seu trabalho. No entanto, sempre as utilize com cautela e avalie criticamente os resultados. Seja código, documentação, ideias ou explicações técnicas, você nunca pode ter certeza absoluta de que sua pergunta foi bem formulada ou que o conteúdo gerado é preciso. Seus colegas são um recurso valioso para ajudá-lo a evitar erros e pontos cegos.

● Mensagem principal

- 👉 Use a IA para reduzir tarefas repetitivas ou tediosas.
- 👉 Desenvolva habilidades de prompt — tanto para codificação quanto para outras tarefas — que beneficiarão sua carreira futura.
- 👉 Aprenda como os sistemas de IA funcionam para melhor antecipar e evitar riscos comuns, vieses e questões éticas.
- 👉 Continue desenvolvendo habilidades técnicas e interpessoais trabalhando com seus colegas.
- 👉 Use apenas conteúdo gerado por IA que você entenda completamente e pelo qual possa se responsabilizar.

● Regras para o aluno:

- Você deve dedicar tempo para explorar as ferramentas de IA e entender como elas funcionam, para que possa usá-las eticamente e reduzir potenciais vieses.
- Você deve refletir sobre seu problema antes de criar o prompt — isso ajuda você a escrever prompts mais claros, detalhados e relevantes, usando vocabulário preciso.

- Você deve desenvolver o hábito de verificar, revisar, questionar e testar sistematicamente tudo o que for gerado por IA.
- Você deve sempre buscar revisão por pares — não confie apenas em sua própria validação.

● Resultados da fase:

- Desenvolver habilidades de prompt tanto para uso geral quanto para áreas específicas.
- Aumentar sua produtividade com o uso eficaz de ferramentas de IA.
- Continuar fortalecendo o pensamento computacional, resolução de problemas, adaptabilidade e colaboração.

● Comentários e exemplos:

- Você encontrará regularmente situações — exames, avaliações e mais — onde você deve demonstrar compreensão real. Esteja preparado, continue desenvolvendo suas habilidades técnicas e interpessoais.
- Explicar seu raciocínio e debater com colegas frequentemente revela lacunas em sua compreensão. Priorize a aprendizagem colaborativa.
- As ferramentas de IA geralmente carecem do seu contexto específico e tendem a fornecer respostas genéricas. Seus colegas, que compartilham seu ambiente, podem oferecer insights mais relevantes e precisos.
- Onde a IA tende a gerar a resposta mais provável, seus colegas podem fornecer perspectivas alternativas e nuances valiosas. Conte com eles como um ponto de verificação de qualidade.

✓ Boa prática:

Pergunto à IA: “Como testo uma função de ordenação?” Ela me dá algumas ideias. Eu as testo e reviso os resultados com um colega. Nós refinamos a abordagem juntos.

✗ Má prática:

Peço à IA para escrever uma função inteira, copio e colo no meu projeto. Durante a avaliação por pares, não consigo explicar o que ela faz ou porquê. Perco credibilidade — e reprovo no meu projeto.

✓ Boa prática:

Uso a IA para ajudar a projetar um analisador sintático. Então, analiso a lógica com um colega. Detectamos dois erros e reescrevemos juntos — melhor, mais limpo e totalmente compreendido.

X Má prática:

Deixo o Copilot gerar meu código para uma parte importante do meu projeto. Ele compila, mas não consigo explicar como ele lida com pipes. Durante a avaliação, não consigo justificar e reprovo no meu projeto.

Capítulo V

Parte obrigatória - cub3D

Nome do programa	cub3D
Arquivos para entregar	Todos os seus arquivos
Makefile	all, clean, fclean, re, bonus
Argumentos	a map in format *.cub
Funções externas autorizadas	<ul style="list-style-type: none">• open, close, read, write, printf, malloc, free, perror, strerror, exit, gettimeofday.• Todas as funções da biblioteca math (-lm man man 3 math).• gettimeofday()• Todas as funções da biblioteca MinilibX.
Libft autorizada	Sim
Descrição	Você deve criar uma representação gráfica 3D "realista" do interior de um labirinto de uma perspectiva de primeira pessoa. Você deve criar esta representação usando os princípios de ray-casting mencionados anteriormente.

As restrições são as seguintes:

- Você deve usar a `miniLibX`. Seja a versão que está disponível no sistema operacional, ou a partir das fontes do projeto. Se você escolher trabalhar com as fontes, você precisará as mesmas regras usadas para a sua `libft` que estão escritas acima na parte **Instruções Comuns**.
- O gerenciamento da sua janela deve permanecer suave: mudar para outra janela,

minimizar, etc.

- Exiba texturas de paredes diferentes (a escolha é sua) que variem dependendo da direção da parede vista (Norte, Sul, Leste, Oeste).

- Seu programa deve permitir configurar as cores do chão e do teto como duas cores diferentes.
- O programa exibe a imagem numa janela e respeita as seguintes regras:
 - As setas esquerda e direita do teclado devem permitir que você olhe para a esquerda e para a direita no labirinto.
 - As teclas W, A, S, e D devem permitir que você mova o ponto de vista através do labirinto.
 - Pressionar **ESC** deve fechar a janela e encerrar o programa de forma limpa.
 - Clicar no X vermelho na borda da janela deve fechar a janela e encerrar o programa de forma limpa.
 - The use of **images** of the **minilibX** library is strongly recommended.
- Seu programa deve receber como primeiro argumento um arquivo de descrição de cena com a extensão **.cub**.
 - O mapa deve ser composto somente de 6 caracteres possíveis: **0** para um espaço vazio, **1** para uma parede, e **N,S,E** ou **W** para a posição inicial do jogador e orientação de spawn.
Este é um mapa simples válido:

```
111111
100101
101001
1100N1
111111
```
 - O mapa deve ser fechado/cercado de paredes, se não o programa deve retornar um erro.
 - Exceto pelo conteúdo do mapa, cada tipo de elemento deve ser separado por uma ou mais linhas vazias.
 - Exceto pelo conteúdo do mapa, que deve sempre ser o último, cada tipo de elemento pode ser escrito em qualquer ordem no arquivo.
 - Exceto pelo mapa, cada tipo de informação de um elemento pode ser separado por um ou mais espaços.
 - O mapa deve ser parseado como ele aparece no arquivo. Espaços são uma parte válida do mapa e cabe a você lidar com eles. Você deve conseguir parsear qualquer tipo de mapa, desde que ele respeite as regras do mapa.

- Exceto pelo mapa, cada elemento deve começar com seu identificador de tipo (composto por um ou dois caracteres), followed by its specific information in a strict order:

- * Textura Norte:

```
NO ./path_to_the_north_texture
```

- identificador: **NO**
 - caminho para a textura norte

- * Textura Sul:

```
SO ./path_to_the_south_texture
```

- identificador: **SO**
 - caminho para a textura sul

- * Textura Oeste:

```
WE ./path_to_the_west_texture
```

- identificador: **WE**
 - caminho para a textura oeste

- * Textura Leste:

```
EA ./path_to_the_east_texture
```

- identificador: **EA**
 - caminho para a textura leste

- * Cor do chão:

```
F 220,100,0
```

- identificador: **F**
 - cores R,G,B no intervalo [0,255]: **0, 255, 255**

- * Cor do teto:

```
C 225,30,0
```

- identificador: **C**
 - cores R,G,B no intervalo [0,255]: **0, 255, 255**

- Exemplo de uma parte obrigatória com uma cena **.cub** minimalista:

```
NO ./path_to_the_north_texture
SO ./path_to_the_south_texture
WE ./path_to_the_west_texture
EA ./path_to_the_east_texture

F 220,100,0
C 225,30,0

111111111111111111111111111111
10000000001100000000000001
10110000011100000000000001
10010000000000000000000001
11111111101100000111000000000001
1000000000110000011101111111111
1111011111111011100000010001
11110111111111011101010010001
1100000011010101110000010001
100000000000000001100000010001
100000000000000001101010010001
1100000111010101111011110N0111
11110111 1110101 101111010001
11111111 1111111 111111111111
```

- Se qualquer tipo de falha de configuração for encontrada no arquivo, o programa deve sair corretamente e retornar "Error\n"seguido de uma mensagem de erro explícita de sua escolha.

Capítulo VI

Parte bônus



Os bônus só serão avaliados se a sua parte obrigatória estiver perfeita. Por perfeita nós naturalmente queremos dizer que ela precisa estar completa, que não deve falhar, mesmo em caso de erros crassos como uso incorreto, etc. Isso significa que se sua parte obrigatória não obtiver TODOS os pontos durante a avaliação, seus bônus serão completamente IGNORADOS.

Lista de bônus:

- Colisões nas paredes.
- Um sistema de minimapa.
- Portas que abrem e fecham.
- Sprites animadas.
- Rotacionar o ponto de vista com o mouse.



Você poderá criar jogos melhores mais tarde, então não perca muito tempo!



É permitido que você use outras funções ou adicione símbolos no mapa para completar a parte bônus desde que seu uso seja justificado durante sua avaliação. Você também pode modificar o formato do arquivo da cena esperada conforme sua necessidade. Seja esperto!

Capítulo VII

Examples



Figura VII.1: Réplica do design original do *Wolfenstein3D*, usando RayCasting.

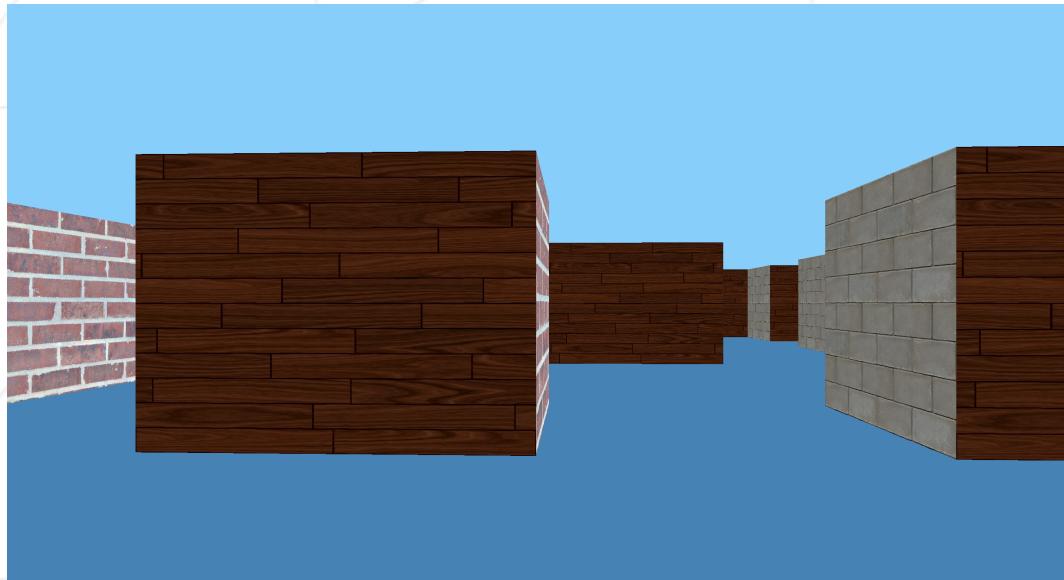


Figura VII.2: Exemplo de como seu projeto pode ficar considerando a parte obrigatória.



Figura VII.3: Exemplo de parte bônus com um minimapa, texturas de teto e chão e uma sprite animada de um porco-espinho famoso.



Figura VII.4: Outro exemplo de bônus com um HUD, barra de saúde, efeito de sombra e arma capaz de atirar

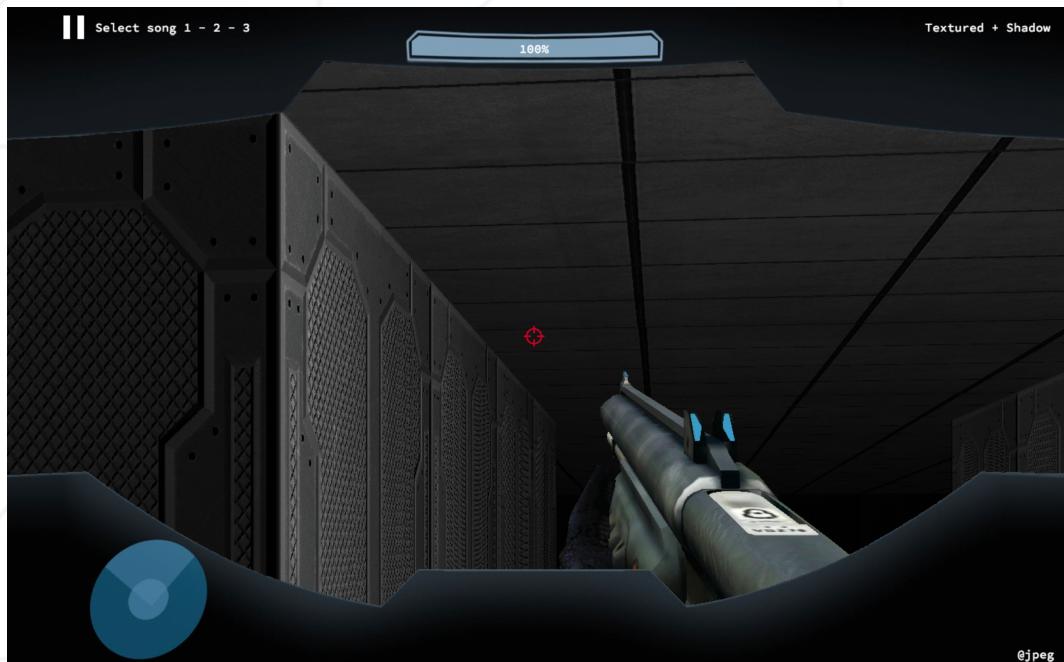


Figura VII.5: Outro exemplo de um jogo com bônus com a arma de sua escolha e o jogador olhando para o teto

Nota: este documento contém imagens protegidas por direitos autorais. Nós consideramos que este subject, criado para fins educacionais, enquadra-se nas diretrizes de Uso Justo.

Capítulo VIII

Submissão e avaliação por pares

Envie sua tarefa em seu repositório `Git` como de costume. Apenas o trabalho dentro do seu repositório será avaliado durante a defesa. Não hesite em verificar duas vezes os nomes de seus arquivos para garantir que estejam corretos.

Durante a avaliação, uma breve **modificação do projeto** pode ser ocasionalmente solicitada. Isso pode envolver uma pequena mudança de comportamento, algumas linhas de código para escrever ou reescrever, ou um recurso fácil de adicionar.

Embora esta etapa possa **não ser aplicável a todos os projetos**, você deve estar preparado para ela se for mencionada nas diretrizes de avaliação.

Esta etapa visa verificar sua compreensão real de uma parte específica do projeto. A modificação pode ser realizada em qualquer ambiente de desenvolvimento que você escolher (por exemplo, sua configuração usual), e deve ser viável em poucos minutos — a menos que um prazo específico seja definido como parte da avaliação.

Você pode, por exemplo, ser solicitado a fazer uma pequena atualização em uma função ou script, modificar uma exibição ou ajustar uma estrutura de dados para armazenar novas informações, etc.

Os detalhes (escopo, alvo, etc.) serão especificados nas **diretrizes de avaliação** e podem variar de uma avaliação para outra para o mesmo projeto.



????????????? XXXXXXXXXX = \$3\$\$796ba5a53df1352e06cc7b0f3ad2a41d