**TESTE DE DESEMPENHO GPU/CPU**

**COM OPENGL**

Trabalho apresentado para a disciplina de Computação Gráfica

**Caio Bertoldo Bezerra**

28 de Outubro de 2025

# **1. Introdução**

Neste trabalho, eu fiz um programa em C++ usando OpenGL para testar o desempenho da minha placa de vídeo. A ideia era simples: desenhar triângulos coloridos girando na tela e ver como o FPS (frames por segundo) mudava quando eu aumentava a quantidade de triângulos ou adicionava efeitos como iluminação e texturas.

O trabalho foi bem interessante porque dá pra ver na prática como diferentes efeitos gráficos impactam o desempenho. Não é só teoria - você vê os números caindo quando adiciona mais complexidade ao programa.

## **1.1 O que eu queria descobrir**

Meus objetivos eram:

• Ver como o FPS muda quando aumento o número de triângulos

• Testar o impacto da iluminação (luz normal e spotlight)

• Ver o que acontece quando adiciono texturas

• Comparar todas essas configurações e entender onde está o gargalo

## **1.2 Por que isso é importante**

Quando a gente está desenvolvendo jogos ou qualquer aplicação gráfica, é super importante saber o que deixa o programa mais lento. Não adianta fazer um jogo bonito se ele fica travando toda hora. Este teste ajuda a entender quais efeitos custam mais caro em termos de processamento.

# **2. Como eu fiz o teste**

## **2.1 Setup do programa**

Usei C++ com as bibliotecas GLUT, GL e GLU. Escolhi o GLUT porque é mais simples que o GLFW - não precisa mexer com shaders e é mais fácil de instalar. A janela ficou em 1280x720 (resolução HD padrão).

O programa cria triângulos coloridos (vermelho, verde e azul) que ficam girando em círculo. É bem visual e dá pra ver claramente quando o FPS cai.

## **2.2 Configurações testadas**

Testei 7 quantidades diferentes de triângulos: 1, 10, 50, 100, 500, 1000 e 5000 triângulos. Para cada quantidade, rodei 6 configurações diferentes:

1. Sem nada (baseline) - só os triângulos coloridos

2. Com luz omnidirecional (luz que ilumina pra todos os lados)

3. Com luz spot (tipo um holofote)

4. Com textura (um padrão de tabuleiro de xadrez)

5. Textura + luz omnidirecional

6. Textura + luz spot (tudo junto!)

No total, foram 42 testes (7 quantidades × 6 configurações). Cada teste rodou por 3 segundos pra dar tempo do FPS estabilizar.

## **2.3 Medição do FPS**

O programa contava quantos frames eram renderizados a cada segundo e mostrava na tela. Simples e direto. Todos os resultados foram salvos num arquivo CSV pra depois eu poder fazer os gráficos.

# **3. Resultados**

## **3.1 Visão geral dos números**

Depois de rodar todos os 42 testes, aqui está o resumo:

**FPS Médio Geral:** 1344 FPS

**FPS Máximo:** 2502 FPS (com 10 triângulos e textura, sem luz)

**FPS Mínimo:** 79 FPS (com 5000 triângulos e luz spot)

Ou seja, a diferença entre o melhor e o pior caso foi enorme! Quase 32 vezes mais rápido no melhor cenário.

## **3.2 O que eu descobri sobre cada configuração**

**Teste básico (sem nada):** Média de 1356 FPS. Esse foi meu baseline - quando o programa está no mais simples possível.

**Com textura (sem luz):** Média de 1471 FPS. Curiosamente, ficou até um pouco mais rápido! Provavelmente porque a textura que eu usei era bem simples (um tabuleiro de xadrez gerado automaticamente).

**Com iluminação (sem textura):** Média de 1309 FPS. Aqui deu pra ver o impacto - perdeu uns 3.5% de desempenho comparado ao básico. A iluminação realmente pesa um pouco.

**Com tudo junto (luz + textura):** Média de 1310 FPS. Praticamente igual a só ter iluminação. Parece que quando você já tem iluminação, adicionar textura não pesa muito mais.

## **3.3 Gráfico 1: FPS sem efeitos (baseline)**

Gráfico, Gráfico de linhas

O conteúdo gerado por IA pode estar incorreto.

*Figura 1: FPS sem efeitos (baseline)*

## **3.4 Gráfico 2: Como a iluminação afeta o FPS**

Gráfico, Gráfico de linhas

O conteúdo gerado por IA pode estar incorreto.

*Figura 2: Como a iluminação afeta o FPS*

## **3.5 Gráfico 3: Impacto da textura**

Gráfico, Gráfico de linhas

O conteúdo gerado por IA pode estar incorreto.

*Figura 3: Impacto da textura*

## **3.6 Gráfico 4: Comparação de tudo**

Gráfico, Gráfico de linhas

O conteúdo gerado por IA pode estar incorreto.

*Figura 4: Comparação de tudo*

## **3.7 Gráfico 5: Quanto cada efeito diminui o desempenho**

Gráfico, Gráfico de cascata

O conteúdo gerado por IA pode estar incorreto.

*Figura 5: Quanto cada efeito diminui o desempenho*

## **3.8 Gráfico 6: Mapa de calor mostrando FPS por configuração**

Gráfico

O conteúdo gerado por IA pode estar incorreto.

*Figura 6: Mapa de calor mostrando FPS por configuração*

# **4. Análise e Discussão**

## **4.1 O que mais me surpreendeu**

A coisa que mais me impressionou foi como o desempenho cai RÁPIDO quando você passa de 1000 triângulos. Olha só:

• 1 triângulo: ~1840 FPS (média)

• 10 triângulos: ~2392 FPS (aumentou!)

• 100 triângulos: ~1863 FPS (ainda ótimo)

• 500 triângulos: ~757 FPS (começou a cair forte)

• 1000 triângulos: ~446 FPS (caiu quase pela metade)

• 5000 triângulos: ~102 FPS (já começa a sentir lag)

É interessante que com 10 triângulos o FPS foi o mais alto de todos. Acho que com 1 triângulo só, a GPU nem "esquenta" direito, e com 10 ela já está trabalhando numa velocidade boa sem estar sobrecarregada.

## **4.2 Sobre a iluminação**

A iluminação foi o efeito que mais impactou o desempenho. Faz sentido, porque pra cada pixel na tela, o programa tem que calcular:

• A luz ambiente (iluminação básica)

• A luz difusa (quanto de luz bate direto no objeto)

• A luz especular (aquele brilho que aparece)

A luz spot foi um pouco mais pesada que a omnidirecional porque ela precisa verificar se cada ponto está dentro do cone de luz e aplicar aquela atenuação suave nas bordas.

## **4.3 Sobre as texturas**

As texturas praticamente não impactaram o desempenho. Na verdade, em alguns casos até melhoraram um pouquinho! Isso provavelmente acontece porque:

• A textura que usei é bem simples (um padrão de xadrez)

• Ela fica na memória da GPU, então é rápido acessar

• As GPUs modernas são MUITO boas em trabalhar com texturas

Mas isso seria diferente se eu usasse texturas gigantes tipo 4K. Aí sim pesaria mais.

## **4.4 GPU vs CPU**

Neste teste, a GPU foi o fator limitante, especialmente quando tinha muitos triângulos. A CPU basicamente só ficou mandando comandos pra GPU e calculando as transformações (rotação, posição, etc.).

Com poucos triângulos (1-100), tanto CPU quanto GPU estão tranquilas. Com 500+, a GPU começa a suar pra processar tudo. Com 5000 triângulos e todos os efeitos ligados, ela tá no limite.

# **5. Conclusão**

## **5.1 O que aprendi**

Este trabalho foi muito legal porque deu pra ver na prática várias coisas que a gente estuda na teoria:

1. Quantidade de geometria importa MUITO - depois de 500 triângulos o FPS cai que é uma beleza

2. Iluminação é cara computacionalmente - especialmente iluminação complexa como spotlight

3. Texturas simples praticamente não custam nada nas GPUs modernas

4. Quando você combina vários efeitos, o impacto não é sempre aditivo - às vezes a GPU consegue otimizar

## **5.2 Aplicações práticas**

Se eu fosse fazer um jogo ou aplicação gráfica, esses testes me ensinariam a:

• Não colocar geometria demais na tela ao mesmo tempo

• Usar LOD (Level of Detail) - objetos longe podem ter menos triângulos

• Limitar o número de luzes dinâmicas na cena

• Não ter medo de usar texturas (desde que não sejam gigantes)

## **5.3 Possíveis melhorias**

Se eu fosse continuar este trabalho, seria interessante testar:

• Múltiplas luzes ao mesmo tempo

• Texturas mais pesadas (4K, normal maps)

• Comparar GLUT com OpenGL moderno (com shaders)

• Ver a diferença entre GPU integrada e dedicada

## **5.4 Considerações finais**

Foi um trabalho bem bacana de fazer. Não só aprendi sobre OpenGL e desempenho gráfico, mas também pratiquei C++, análise de dados com Python e criação de gráficos.

O mais legal é que agora quando eu jogar algum game e ver as opções gráficas, vou entender melhor o que cada coisa faz e por que algumas opções pesam mais que outras no desempenho.

# **6. Referências**

KHRONOS GROUP. OpenGL Documentation. Disponível em: https://www.opengl.org/

FREEGLUT. The Free OpenGL Utility Toolkit. Disponível em: http://freeglut.sourceforge.net/

Material da disciplina de Computação Gráfica

Tutoriais online sobre OpenGL e GLUT

# **Anexo - Código do Programa**

O código completo do programa está incluído no arquivo main\_glut.cpp que acompanha este relatório.

**Principais características do código:**

• Aproximadamente 350 linhas de código C++

• Usa bibliotecas GLUT, GL e GLU

• Testes automatizados que rodam sozinhos

• Geração automática de arquivo CSV com resultados

• Textura procedural criada em código

• Sistema de iluminação usando funções do OpenGL

**Para compilar e executar:**

1. Instalar dependências: sudo apt-get install freeglut3-dev

2. Compilar: make

3. Executar: ./gpu\_test\_glut

4. Gerar gráficos: python3 generate\_graphs.py