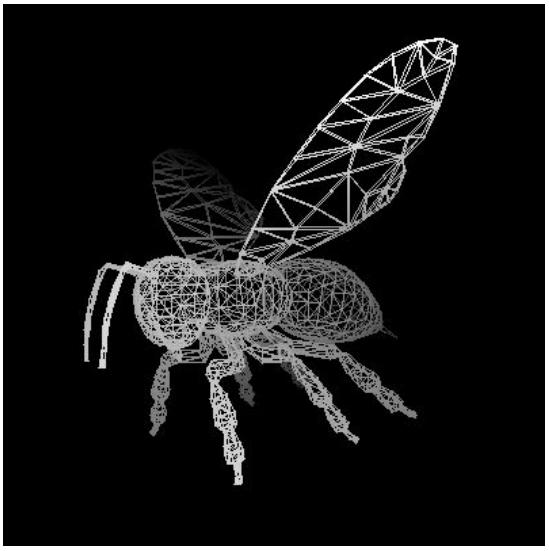
Aluno: Eric Löw Schmidt
Disciplina: INFO7072 - TÓPICOS ESPECIAIS III

O trabalho desenvolvido foi um algoritmo para ler arquivos de objetos tridimensionais e projetá-los em perspectiva em formato wireframe. O programa recebe da entrada um arquivo e pode tanto desenhar em uma janela a projeção de um objeto 3D quanto gerar uma imagem de saída no formato PPM.

A representação wireframe é uma representação visual de um objeto onde apenas as suas bordas são desenhadas, ou seja, não são representadas texturas nem cores do objeto.

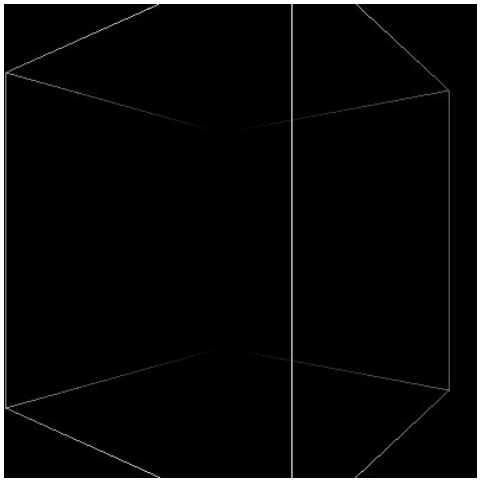

Exemplo de objeto em wireframe.

O algoritmo desenvolvido é capaz de abrir e ler arquivos em formato Wavefront OBJ, que é um formato de arquivo que armazena objetos tridimensionais por meio de um conjunto de vértices e faces descritos em um plano cartesiano de 3 dimensões. Cada vértice deste arquivo guarda uma posição X, Y e Z, representando sua localização no espaço. Cada face do arquivo é composta por um conjunto de 3 a 4 vértices interligados. A união de todas as faces descritas no arquivo gera o objeto tridimensional. Apesar deste formato de arquivo guardar outros tipos de informação(como textura, por exemplo), elas são dispensáveis para a realização do trabalho proposto.

```
 1 v 16 32 16
 2 v 16 32 -16
 3 v 16 0 16
 4 v 16 0 -16
 5 v -16 32 16
 6 v -16 32 -16
 7 v -16 0 16
 8 v -16 0 -16
 9
10 f 1 3 4 2
11 f 6 8 7 5
12 f 2 6 5 1
13 f 3 7 8 4
14 f 1 5 7 3
15 f 4 8 6 2
16
```
Exemplo de arquivo em formato Wavefront(v=vértices, f=faces)

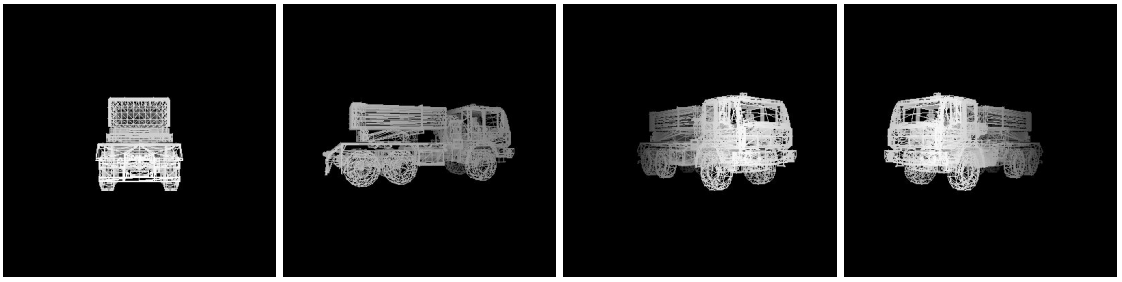
Após a leitura do arquivo é realizado um processo de ajuste dos vértices do objeto, a fim de mantê-los centralizados dentro do intervalo [-1..1] para as dimensões X, Y e Z.

O processo de projeção de um objeto 3D em um plano 2D consiste basicamente em converter as coordenadas dos vértices de cada face do objeto em um ponto [X,Y] da tela e desenhar linhas interligando-os consecutivamente. O objeto é sempre projetado em torno do ponto [0,0,0] do espaço dimensional. Uma câmera com posição [0,0,3] é utilizada para gerar a projeção e visualização do objeto em relação a sua posição no espaço. As proporções do objeto e os cálculos de perspectiva levam em conta sempre a posição da câmera. A projeção do objeto leva em conta também a profundidade de cada vértice em relação a câmera para gerar as arestas de cada face do objeto, pois foi adicionado um efeito de gradiente na função que desenha as arestas do objeto. Quanto mais afastado da câmera uma aresta estiver, mais escura ela será, e quanto mais próximo, mais clara.


O efeito de gradiente pode facilmente ser visto na projeção do cubo.

No algoritmo sequencial, todos os vértices passam pelo processo de projeção para o ambiente 2D e só então são desenhados na tela. Já no algoritmo paralelo, cada thread da GPU recebe apenas uma aresta para lidar, com isso, cada thread se encarregará de projetar e desenhar apenas uma linha do objeto no plano 2D, não havendo a necessidade de pré-processar todos os vértices antes de serem desenhados.

O grau de complexidade da projeção do objeto gira em torno do número de faces que ele possui. Quanto mais faces o objeto tiver, maior o tempo de processamento. O algoritmo possui boa escalabilidade, já que os processos de projeção dos vértices de cada face não são dependentes entre si.



Ao executar o programa, o objeto é rotacionado horizontalmente em N diferentes posições consecutivas dentro do intervalo de 360 graus. Cada uma das diferentes posições é projetada e utilizada para gerar um arquivo de saída ou a imagem na tela. Cada nova posição difere (360 / N) graus de seu antecessor.

A divisão dos blocos da GPU foi feita em 1 dimensão, de modo que existem N blocos de tamanho 4, onde N é o número total de faces do objeto. Desta forma, cada thread se encarrega de projetar apenas a aresta de número ‘threadIdx.x’ da face de número ‘blockIdx.x’ do objeto. Uma segunda abordagem foi experimentada para o programa, utilizando duas dimensões de blocos da GPU, onde nela haveriam N blocos de tamanho 4 na dimensão Y, entretanto, esta abordagem apresentou um desempenho menor em relação a abordagem com apenas 1 dimensão, por isso a primeira abordagem foi mantida.

A janela para exibir a projeção dos objetos é criada por meio da biblioteca Xlib. Nenhuma biblioteca externa as padrões do sistema Linux foi utilizada para desenvolver esse trabalho. As funções que desenham as linhas na tela foram também desenvolvidas manualmente. Uma função calcula os pixels a serem percorridos por uma linha e aplica sobre eles a cor com maior intensidade a ser desenhada(para isso foi utilizada a função atomicMax no algoritmo paralelo).

O programa pode ser compilado por meio do comando ‘make all’. Ao executar o programa deve-se passar ao menos o nome do arquivo de entrada por meio de argumento, caso contrário o programa avisará da falta de argumentos. A execução do algoritmo conta com as seguintes opções de argumentos:

-s: Executa o algoritmo sequencial.

-f <file>: Especifica o algoritmo de entrada.

-r <N>: Especifica a quantidade de posições a serem geradas durante a rotação do objeto."

-w: Abre uma janela para exibir a projeção criada.

-x <N>: Altera a largura da tela.

-y <N>: Altera a altura da tela.

-p: Nome da pasta para salvar as imagens geradas.

-t: Define o número de vezes que o algoritmo será executado consecutivamente(1 por padrão).

-h: Abre o menu de ajuda.

Os testes foram realizados no servidor Orval do departamento de informática da UFPR, com uma GPU GTX750 ti. Para avaliar o desempenho do algoritmo proposto, foi utilizado como parâmetro seu tempo de execução. Um contador é iniciado imediatamente antes da execução da função de projeção e parado imediatamente após(tanto para a execução sequencial quanto para a paralela). Os tempos das projeções de todas as N diferentes posições do objeto são somados para mostrar o tempo total de processamento do algoritmo de projeção. A relação entre os tempos do algoritmo sequencial e do algoritmo paralelo em GPU serão usados como parâmetro de desempenho.

Para a realização dos testes foi criado um parâmetro no programa que permite a repetição das projeções consecutivamente(-t <N>). Basicamente o programa se repete N vezes e calcula a média de todas as repetições.

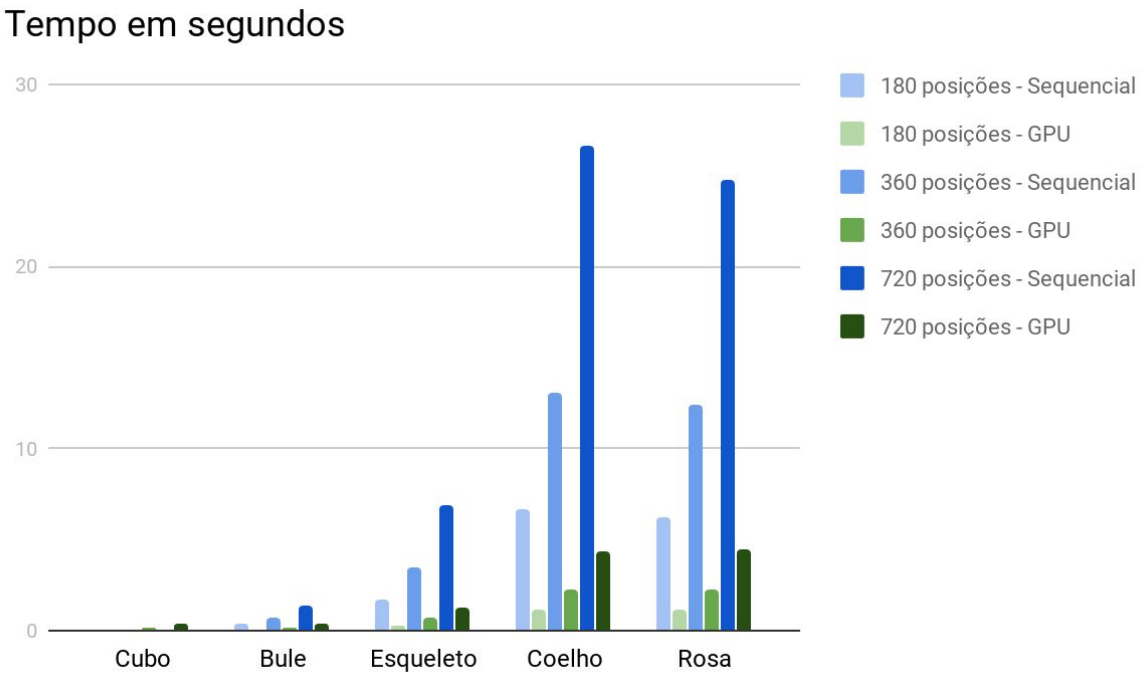
Objeto	Vértices	Faces
Cubo	8	6
Bule	822	1600
Esqueleto	8338	16034
Coelho	34834	69451
Rosa	83272	83346

Para realizar os testes, foram escolhidos 5 objetos de diferentes tipos: Cubo(8 vértices e 6 faces), Bule(822 vértices e 1600 faces), Esqueleto(8338 vértices e 16034 faces), Coelho(34834 vértices e 69451 faces) e Rosa(83272 vértices e 83346 faces). Cada objeto foi submetido aos algoritmos sequencial e paralelo com 5 diferentes posições de rotação(16 posições, 64 posições, 180 posições, 360 posições e 720 posições).

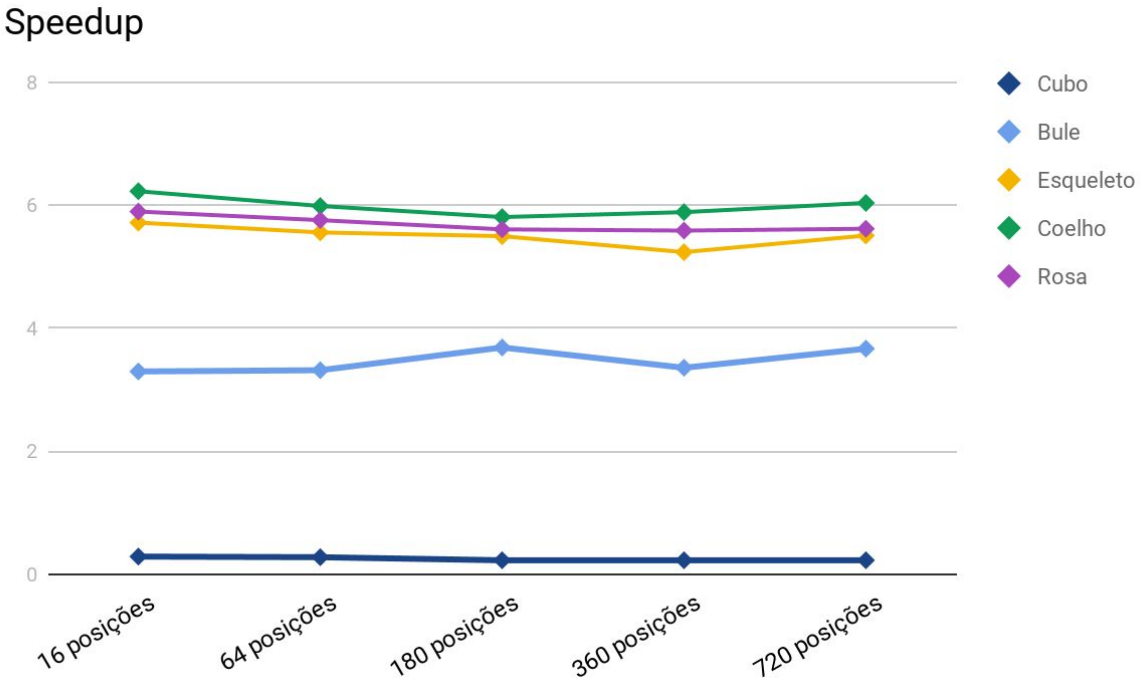
Para cada teste foi calculado seu tempo de execução médio. A relação entre a média dos tempos de execução sequencial e paralelo, bem como seu speedup, pode ser conferido na tabela abaixo:

Objeto		16 posições	32 posições	180 posições	360 posições	720 posições	Média
Cubo	CPU	0,002162	0,007280	0,019712	0,039241	0,078220	0,25
	GPU	0,007469	0,026265	0,083892	0,167420	0,334416	
	Speedup	0,29	0,28	0,23	0,23	0,23	
Bule	CPU	0,031912	0,128591	0,353791	0,681873	1,379918	3,47
	GPU	0,009678	0,038733	0,095909	0,203216	0,375535	
	Speedup	3,30	3,32	3,69	3,36	3,67	
Esqueleto	CPU	0,165667	0,640739	1,729036	3,469801	6,919671	5,51
	GPU	0,028965	0,115167	0,314385	0,661974	1,255451	
	Speedup	5,72	5,56	5,50	5,24	5,51	
Coelho	CPU	0,617637	2,370405	6,612361	13,099240	26,590607	5,99
	GPU	0,099118	0,395756	1,138831	2,222781	4,404403	
	Speedup	6,23	5,99	5,81	5,89	6,04	
Rosa	CPU	0,585055	2,282901	6,228005	12,412499	24,762575	5,70
	GPU	0,099152	0,396195	1,110844	2,220001	4,407319	
	Speedup	5,90	5,76	5,61	5,59	5,62	

Os seguinte gráfico mostra as relações entre os tempos dos algoritmos sequenciais e paralelo para 180, 360 e 720 posições de rotação:



O gráfico abaixo mostra a relação entre os diferentes Speedups de cada teste:



É possível notar uma melhora significativa de desempenho na grande maioria dos casos, com exceção do objeto Cubo, onde houve um speeddown. Objetos de poucas faces tendem a ser mais eficientes no algoritmo sequencial do que no paralelo. Entretanto, esta diferença de tempo é pouco notável nestes casos menores, pois ambos os algoritmos sequencial e paralelo executam em menos de meio segundo. Já para casos maiores, é possível ver um desempenho muito superior do algoritmo paralelo. Objetos com muitas faces, como o coelho e a rosa, já apresentam um speedup estável próximo de 6, mostrando a vantagem de se paralelizar o algoritmo para entradas com milhares de faces.