Introdução à Computação Gráfica

Emmanuella Faustino Albuquerque
20170002239

Atividade Prática 3: Implementação de Modelos Locais de Iluminação

08 de novembro de 2021

VISÃO GERAL

Nesta atividade foram implementadas, com o Modelo de Iluminação de Phong, as interpolações de Gouraud e de Phong utilizando GLSL Shaders no Three JS.

ESTRATÉGIAS

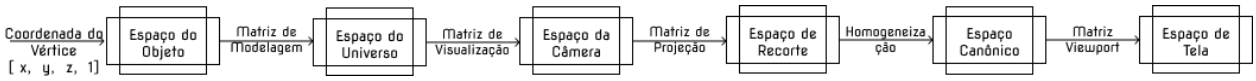
Primeiramente, foi preciso analisar a equação do modelo de Phong abaixo, com isso foi possível perceber que de todas as variáveis já declaradas era preciso encontrar o valor de v: (que é o vetor normalizado que aponta para a câmera).

$$I = I_a\kappa_a + I_p\kappa_d\,(\mathbf{n}\cdot\mathbf{l}) + I_p\kappa_s\,(\mathbf{r}\cdot\mathbf{v})^n \quad \text{(i)}$$

Definindo o vetor V

Sabendo que o valor de v, pode ser definido pela equação abaixo, podemos realizar o cálculo para encontrá-lo e então, obter a equação completa I(i).

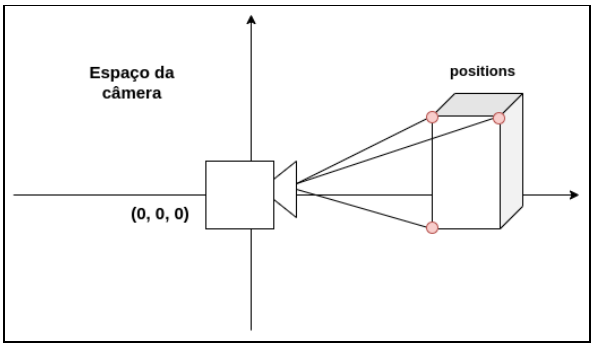
$$V = \left|Camera_{(position)} - Vertice_{(position)}\right| \quad \text{(ii)}$$



Logo, como é possível observar na Imagem 1 e na equação (ii) apresentada para calcular o valor de V, é preciso saber a posição da câmera, a posição do vértice analisado e então realizar a normalização do vetor.

```
// 'position_cam_spc' : Posicao da Camera no Espaco do Camera. (vec4)
vec4 position_cam_spc = viewMatrix * vec4(cameraPosition, 1.0);

// Esp.Camera: (CameraPosition − VerticePosition)
v = normalize(position_cam_spc.xyz − P_cam_spc.xyz);
```



. Imagem 1: Análise do vetor V

Com isso, utilizando a posição da câmera (i.e position_cam_spc) e a variável que contém o vértice transformado para o Espaço de Câmera (i.e P_cam_spc), foi possível calcular o vetor V.

Negando o vetor L (vetor da fonte luz)

Após encontrar o valor de V foi possível gerar o I, isto é, a intensidade final calculada para o vértice ou fragmento, e assim gerar a estrutura básica para a interpolação de Gouraud e de Phong. Porém, como visto em aula, é possível que aconteça a situação da imagem abaixo, na qual o vetor L está na direção contrária da que queremos (case b):

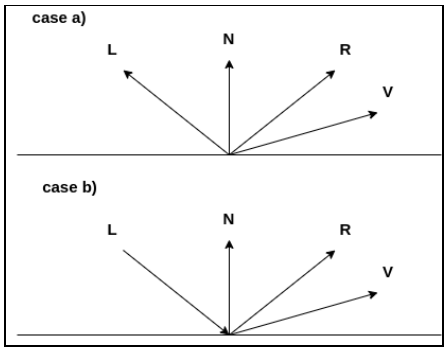


Imagem 2: Análise do vetor L

Assim, foi preciso negar a direção do vetor L, (i.e vetor que aponta para a fonte de luz pontual/direcional).

Assim, foi preciso negar a direção do vetor L, (i.e vetor que aponta para a fonte de luz pontual/direcional).

```
// 'r' : reflexao de l em relacao a n. (vec3)
// 'l' : vetor normalizado que aponta para a fonte de luz pontual/direcional.
// 'n' : vetor normal no ponto onde se esta avaliando a iluminacao.

vec3 r = reflect(-l, n);
```

RESULTADOS

Gouraud Shading (interpolação por vértice)

Por meio da interpolação de Gouraud, é possível observar na Imagem 3, que o brilho gerado é deformado, pois está sendo afetado pela malha dos triângulos do Torus.

Phong Shading (interpolação por fragmento)

Já, por meio da interpolação de Phong, na Imagem 4, como o brilho é calculado individualmente para cada fragmento, ele fica mais suave, sem se deformar pela geometria utilizada.

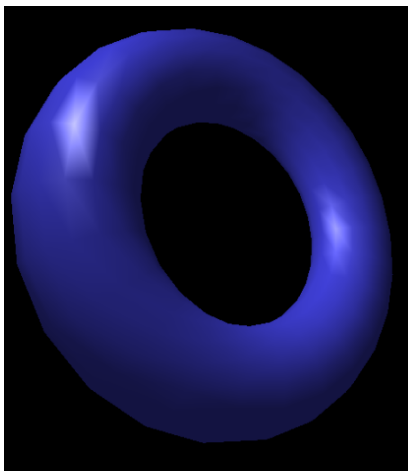


Imagem 3: Rendering do Torus. (interpolação de Gouraud)

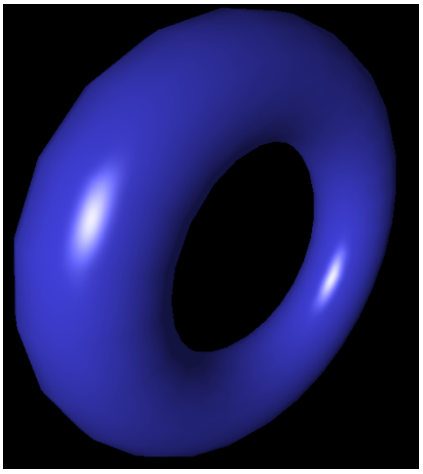


Imagem 4: Rendering do Torus. (interpolação de Phong)

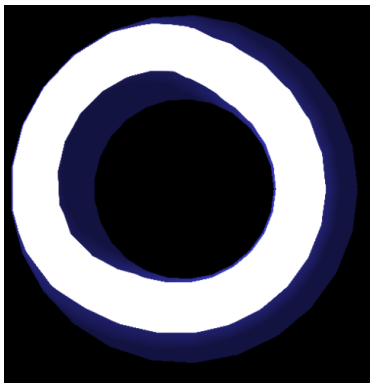


Imagem 5: Interpolação de Phong
(sem renormalização do vetor normal)

Como visto em aula, no estágio de rasterização as variáveis vão ser interpoladas e os vetores normais dos fragmentos vão ser estimados.

Como os vetores normais têm que ter comprimento igual a um, antes de realizar o cálculo I (i) para a interpolação de Phong, foi preciso renormalizar os vetores normais nesse caso, o N_cam_spc.

Com isso, foi possível obter o resultado correto do Modelo de Iluminação de Phong.

Dificuldades

Uma das dificuldades foi perceber que a variável L do termo difuso (o vetor normalizado que aponta para a fonte de luz pontual/direcional), precisava ter sua direção alterada para -L.

Possíveis melhorias

Seria interessante criar uma estrutura de visualização onde fosse possível observar ambas as interpolações de Gouraud e de Phong lado a lado.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] WebGLProgram. Threejs. Disponível em: https://threejs.org/docs/#api/en/renderers/webgl/WebGLProgram. Acesso em: 03 de novembro de 2021.

[2] ShaderMaterial. Threejs. Disponível em: https://threejs.org/docs/#api/en/materials/ShaderMaterial. Acesso em: 04 de novembro de 2021.

[3] ThreeJS - Shaders in Separate Files (with syntax highlighting). LoFi. Youtube, 13 de abril de 2021. Disponível em: https://youtu.be/GEF7OxVCP_0. Acesso em: 05 de novembro de 2021.