Aluno: Renilton Ribeiro de Oliveira Matrícula: 11328395

Atividade 3

Introdução

Esse é um relatório sobre a terceira atividade de computação gráfica, implementação de dois modelos de iluminação, essa disciplina é ministrada pelo professor Christian Pagot, na UFPB, no período 2020.2. Para esse trabalho foi utilizada a linguagem de programação Javascript juntamente com o framework disponibilizado pelo professor, o programa template renderiza um torus vermelho na tela, como ilustrado na figura 1.

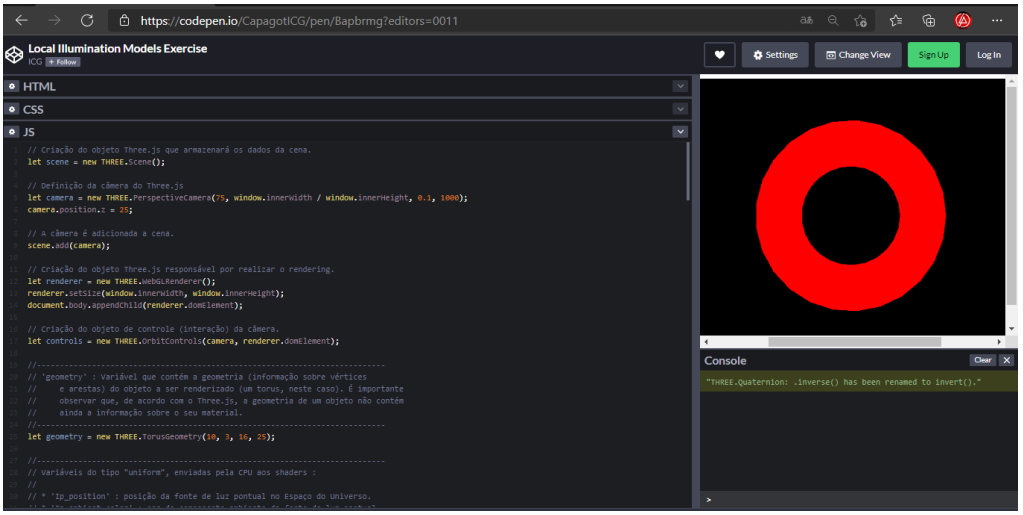

figura 1. Renderização do torus utilizando a cor vermelha.

Bom, os modelos de Iluminação são técnicas usadas para calcular a intensidade da cor de um ponto a ser exibido. Também chamados modelos de reflexão, eles utilizam a cor do objeto, a cor da luz, a posição da luz, a posição do ponto e a posição do observador.

Neste trabalho vamos utilizar o modelo de iluminação de Phong, ele é composto pela adição de três modelos distintos de iluminação local: os modelos ambiente, difuso e especular, abaixo podemos observar a equação que descreve o modelo.

$$I = I_a K_a + I_p K_d (\mathbf{n} \cdot \mathbf{l}) + I_p K_s (\mathbf{r} \cdot \mathbf{v})^n$$

Onde:

- I : intensidade (cor) final calculada para o vértice ou fragmento.
- Ia : intensidade da luz ambiente.
- Ka : coeficiente de reflectância ambiente.
- Ip : intensidade da luz pontual/direcional.
- Kd : coeficiente de reflectância difusa.
- n : vetor normal.
- l : vetor que aponta para a fonte de luz pontual/direcional.

- Ks : coeficiente de reflectância especular.
- r : reflexão de l sobre n.
- v : vetor que aponta para a câmera.
- n : tamanho do brilho especular.
- IaKa : termo ambiente.
- IpKd (n · l) : termo difuso.
- IpKs (r · v) n : termo especular.

Estratégia

Optei pela por dedicar tempo a entender a sintaxe utilizada ao manipular shaders, na primeira parte da tarefa optei por ir fazendo separadamente cada modelo de iluminação(ambiente, difuso e especular) para observar se os resultados eram os desejados por mim, após isso fui para o modelo de iluminação de Phong em si, após o término da primeira parte da atividade, foi relativamente fácil fazer a parte dois, já que era havia sido bem explicada na aula do dia 20 de maio de 2021, tive apenas que manipular os dados da parte 1 de forma a mover os cálculos de iluminação do vertex shader para o fragment shader.

Resultado(Parte 1)

Na figura 2 podemos observar a aplicação do modelo de iluminação ambiente sobre a figura, representado pelo trecho de código:

*vec3 T_diffuse = Ip_diffuse_color * k_d * max(dot(N_cam_spc, L_cam_spc), 0.0);*

Esse modelo é bem simples, segundo esse modelo a cor final do pixel é exclusivamente determinada pela intensidade da luz ambiente e pelo coeficiente de refletância ambiente.

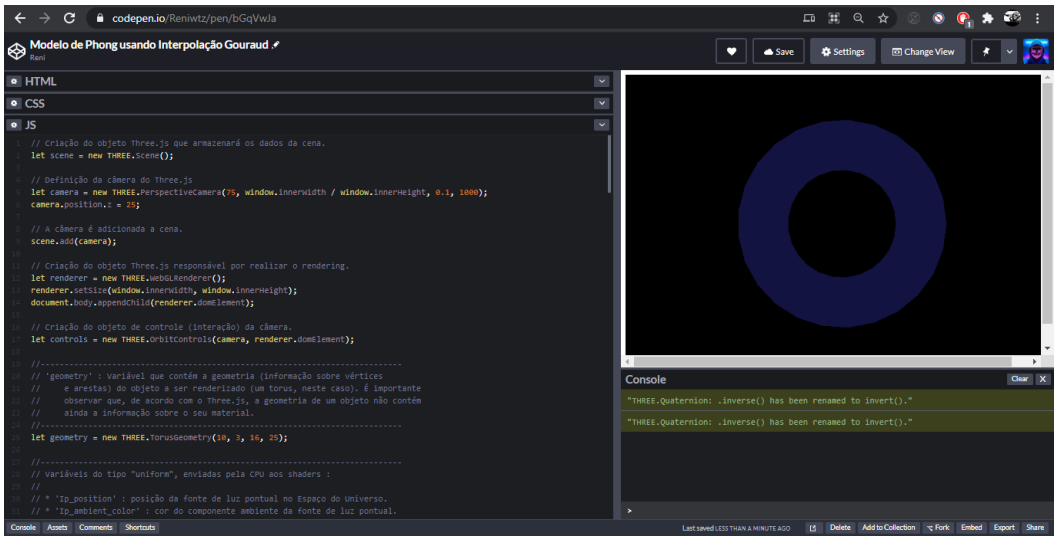

FIGURA 2. Torus renderizado utilizando o modelo de iluminação ambiente.

Na figura 3 podemos observar a aplicação do modelo de reflexão difuso sobre a figura, representado pelo trecho de código abaixo.

*vec3 T_diffuse = Ip_diffuse_color * k_d * max(dot(N_cam_spc, L_cam_spc), 0.0);*

Segundo esse modelo a cor final do pixel é determinada pela a intensidade da fonte de luz pontual, vezes coeficiente de refletância e vezes o produto interno entre o vetor da luz e o vetor normal, produto esse que representa o cosseno de teta, o produto interno pode gerar números negativos caso o ângulo seja maior que 90 graus, então durante a implementação a fim de evitar isso calculamos o max entre zero e o produto interno, quando for maior ou igual a zero ficamos com o produto interno, quando for menor ficamos com zero.

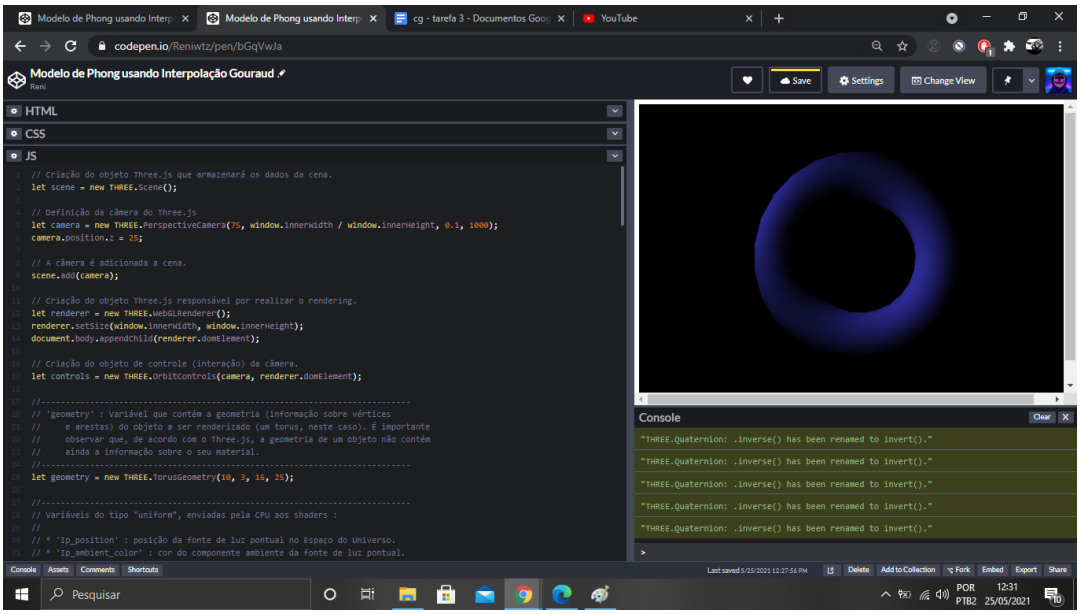

figura 3. Torus renderizado utilizando o modelo de reflexão difuso.

Na figura 4 podemos observar a aplicação do modelo de iluminação Phong sobre a figura, representado pelo trecho de código abaixo.

*vec3 T_speculate = Ip_diffuse_color * k_s * pow(max(dot(R_cam_spc, V_cam_spc), 0.0), 16.0);*

Segundo esse modelo a cor final do pixel é determinada pela a intensidade da fonte de luz pontual, vezes coeficiente de refletância e vezes o produto interno entre o vetor R(que representa a reflexão do vetor de luz sobre o vetor normal) e o vetor que aponta para câmera, produto esse que representa o cosseno de teta, o produto interno pode gerar números negativos caso o ângulo seja maior que 90 graus assim como o modelo difuso, então durante a implementação novamente a fim de evitar isso calculamos o max entre zero e o produto interno, quando for maior ou igual a zero ficamos com o produto interno, quando for menor ficamos com zero.

A fim de obtermos um brilho especular maior, temos que fazer com que o cosseno de teta chegue mais devagar de um até zero, para isso multiplicamos ele por uma potência(representado pelo n na equação), no caso da atividade o valor dessa potência é 16.

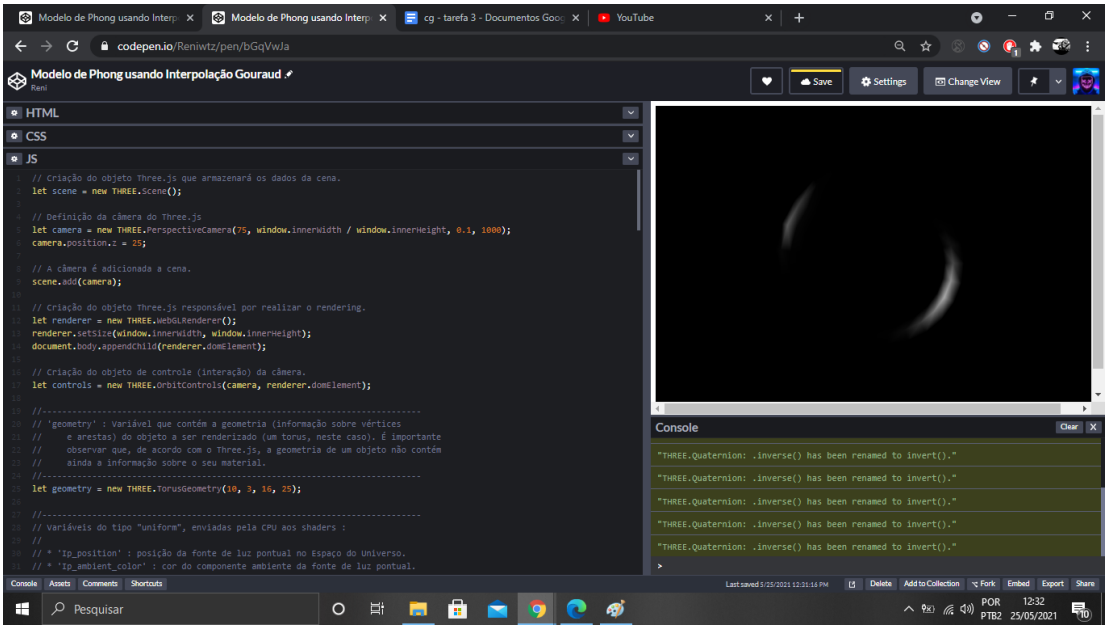

figura 4. Torus renderizado utilizando o modelo de reflexão especular.

Na figura 4 podemos observar o resultado final do modelo de Phong usando interpolação Gourand, representado pela equação do trecho de código abaixo.

I = vec4(T_ambient + T_diffuse + T_speculate, 0);

Segundo esse modelo de Phong, temos que somar os termos calculados anteriormente(ambiente, difuso, especular). onde cada termo vai suprir uma deficiência do outro com o objetivo de conseguir um resultado mais realista como podemos observar abaixo.

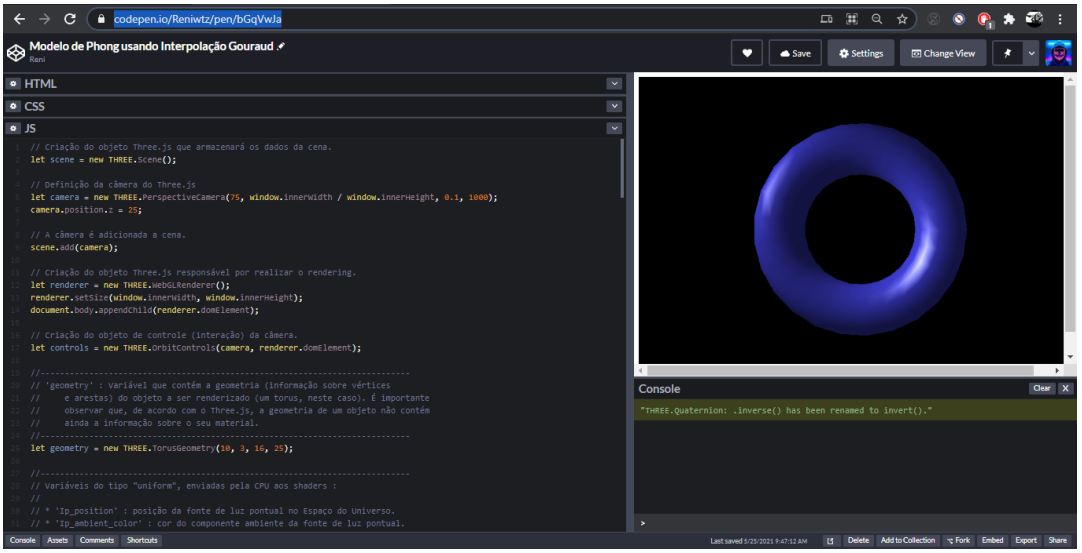

figura 5. Torus renderizado utilizando o modelo de iluminação de Phong e a interpolação de Gouraud.

Resultado(Parte 2)

Quando o modelo de interpolação utilizado é o de Phong, normalmente o vertex shader fica encarregado apenas de transformar o vértice, originalmente no espaço do objeto, para o espaço de recorte, enquanto que a avaliação do modelo de iluminação passa a ser função do fragment shader.

Após o término da parte um os cálculos de iluminação foram movidos do vertex shader para o fragment shader para execução da segunda parte da atividade, alguns vetores foram mantidos no vertex shader como sugerido no exercício, vetores esses citados abaixo.

• N_cam_spc : vetor normal no espaço da câmera.
• L_cam_spc : vetor da fonte luz no espaço da câmera..
• R_cam_spc : reflexão do vetor L_cam_spc sobre o vetor N_cam_spc.
• V_cam_spc : vetor que aponta para a câmera(vetor calculado pelo Aluno).

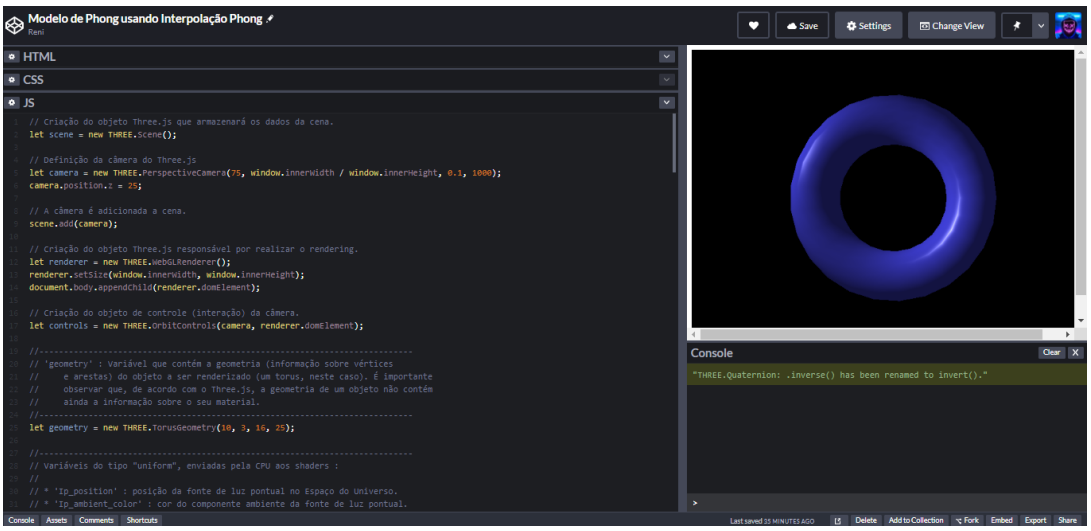

figura 6. Torus renderizado utilizando o modelo de iluminação de Phong e a interpolação de Phong.

Dificuldade e Melhorias Possíveis

A grande dificuldade foi em entender a sintaxe utilizada na atividade, mas com as aulas síncronas muitas dúvidas foram respondidas a respeito da mesma, além disso por incrível que pareça foi em achar o “v” da equação.

O Trabalho poderia ser refeito em outra linguagem que o deixasse mais otimizado, poderia ter sido feito o desenvolvimento de funções que permitisse que fosse rotacionando o cubo automaticamente, e conforme isso, ele fosse interpolando a cores, além disso, no meu código com certeza deve haver coisas que poderiam ter sido feitas de forma mais eficaz, optei por fazer de um jeito que ficasse mais didático em algumas partes.

Referências

1. Notas e vídeos de aulas do Prof. Christian Azambuja Pagot.
2. https://threejs.org/docs/index.html#api/en/

Link para o código

Parte 1.
https://codepen.io/Reniwtz/pen/bGqVwJa

Parte 2.
https://codepen.io/Reniwtz/pen/WNpOrxj?editors=1011