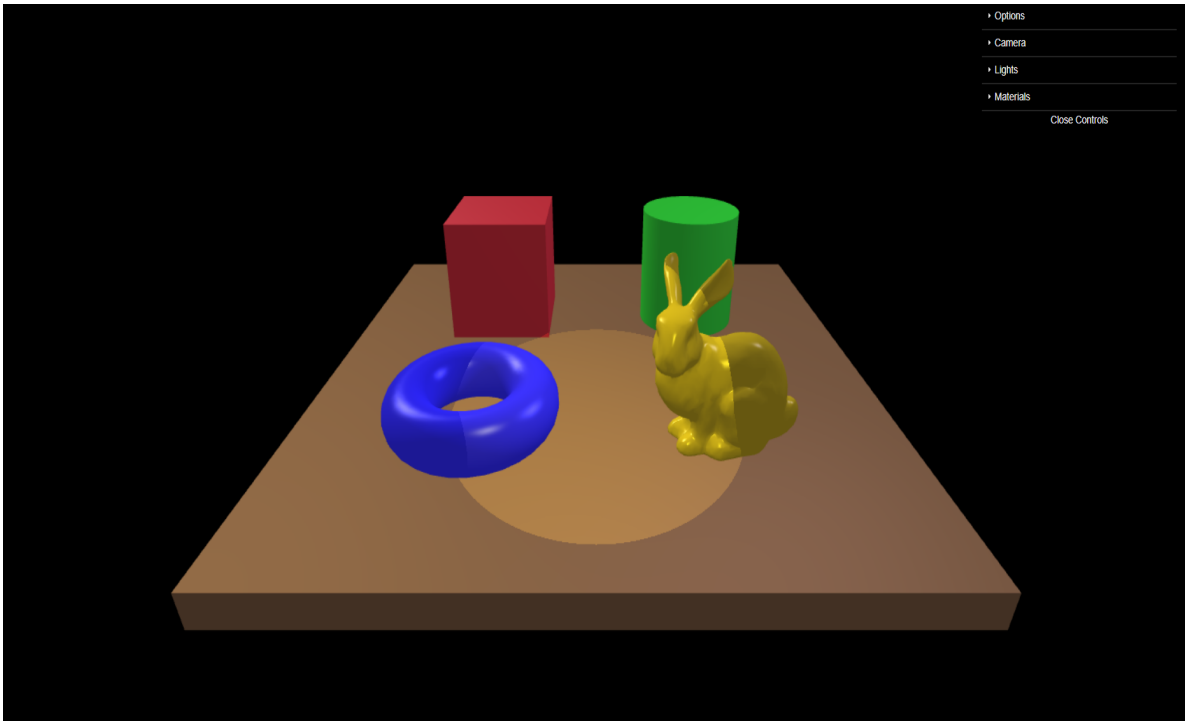
Modelo de iluminação de Phong com sombreamento de Phong.

CGI - Projeto 3



Autores (G06-08):
Bernardo Viegas 60311
Pedro Fernandes 60694

Índice

Tipos de dados das luzes

Tanto na parte javascript como nos shaders, foram usadas as seguintes atributos, com os seguintes tipos:

- **bool** active (apenas presente na parte javascript)
- **vec3** ambient* (componente ambiente da luz)
- **vec3** diffuse* (componente difusa da luz)
- **vec3** specular* (componente especular da luz)
    - *javascript: RGB 0-255
    - *shader: RGB 0.0-1.0
- **vec4** position
    - w = 0, vetor representa a direção duma luz direcional
    - w = 1, vetor representa a posição duma luz pontual
- **vec3** axis (eixo de direção do spotlight)
- **float** aperture (ângulo de abertura do spotlight)
- **float** cutoff (parâmetro de decaimento da luz spotlight em relação ao eixo)

Apenas para o javascript, foi introduzida uma variável extra que permite ativar/desativar cada luz. Quando esta é **true**, o javascript envia para o shader as componentes de luz devidas (ambient, diffuse, specular), caso contrário, envia para o shader vetores nulos para as mesmas.

Como variar o número de luzes

Para adicionar luzes novas, será necessário adicionar um objeto novo ao objeto lights no código javascript com todos os atributos mencionados anteriormente. O nome de atributo no objeto lights é o nome da luz em si, que ficará presente na interface gráfica.

Implementação da luz do tipo de spotlight

Começamos por determinar o cosseno do ângulo que a luz faz para cada fragmento em relação ao eixo do spotlight, e se este for menor que o cosseno da abertura, essa luz para esse fragmento não é somada.

```
float spotCos = dot(lightDirection, normalize(-light.axis));

if (spotCos < cos(light.aperture)) continue;
```

De seguida calculamos a luz a partir das componentes ambiente, difusa e especular da luz no ciclo atual com o material do fragmento, tendo também em conta o fator de atenuação multiplicada às componentes difusa e especular.

```
vec3 sumLight = determineLight(light, normal, lightDirection);
```

Finalmente, determina-se o cutoff efetivo para a luz calculada. Como a abertura pode variar entre 0 e 180 graus, o cosseno do spotlight também pode variar entre -1 e 1. Para impedir que o cutoff seja positivo para todos os valores, de forma suave, é transformado o valor do cosseno do spotlight para que fique entre 0 e 1.

```
float effectiveCutoff = pow((spotCos + 1.0) / 2.0, light.cutoff);
```

A luz final será então a luz multiplicada por esse cutoff efetivo.

```
finalLight += sumLight * effectiveCutoff;
```

Implementação do desafio, e as suas implicações

Para arrastar a câmera, foi necessário imaginar a posição desta como estando numa esfera centrada na posição para onde a câmera está a olhar, com raio igual à distância entre estas duas posições. Qualquer ponto numa esfera pode ser determinado por meio de equações paramétricas de dois parâmetros, θ e φ. Estes são os parâmetros que são variados quando se mexe o rato.

Para poder variar a posição da câmera, primeiro anotam-se os valores de θ e φ iniciais, tanto como o raio da esfera mencionada ao se segurar inicialmente o botão esquerdo do rato.

À medida que se vai movendo o rato, com o botão pressionado, calcula-se a variação da posição do rato em ambos os eixos, variando então o θ e o φ pelo movimento do rato no eixo dos x e y respectivamente. Esta variação é normalizada em relação ao tamanho da janela, permitindo então que o rato rode a câmera de forma consistente. Nesta fase, após determinar os novos ângulos, é então alterada a posição real da câmera. O movimento do rato atualiza também este valores visíveis na interface gráfica.

Enquanto se arrasta o rato sobre a interface gráfica, se se tiver iniciado o movimento fora desta, a câmera roda, se se tiver iniciado dentro, a câmera não roda.