# Atividade Prática 3 - Implementação de Modelos Locais de Iluminação

**GDSCO0051 - Introdução à Computação Gráfica - 2021.1**

*Data de entrega: 08/11/2020, 23h59min.*

## 1 Atividade

Nesta atividade os alunos implementarão o *Modelo de Iluminação de Phong*. O modelo será implementado de duas formas diferentes: uma vez utilizando *Gouraud shading* (*i.e.* sombreamento por vértice) e outra utilizando *Phong shading* (*i.e.* sombreamento por fragmento).

## 2 Objetivo

O objetivo deste trabalho é familiarizar os alunos com os conceitos de modelo de iluminação local e de interpolação tradicionalmente utilizados em computação gráfica.

## 3 O *Framework*

O trabalho deve ser desenvolvido em JavaScript utilizando-se, como ponto de partida, o *framework* disponível no endereço:

`https://codepen.io/ICG-UFPB/pen/LYygXgq`

## 4 Fundamentação Teórica

### 4.1 Modelo de Iluminação de *Phong*

O modelo de iluminação de Phong, como visto em aula, é composto pela adição de três modelos distintos de iluminação local: os modelos ambiente, difuso e especular. A Equação 1 descreve o modelo de Phong.

$$I = I_a \kappa_a + I_p \kappa_d \left(\mathbf{n} \cdot \mathbf{l}\right) + I_p \kappa_s \left(\mathbf{r} \cdot \mathbf{v}\right)^n \tag{1}$$

onde

- $I$ : intensidade (cor) final calculada para o vértice ou fragmento.
- $I_a$ : intensidade (cor) da luz ambiente.
- $\kappa_a$ : coeficiente de reflectância ambiente.
- $I_p$ : intensidade (cor) da luz pontual/direcional.
- $\kappa_d$ : coeficiente de reflectância difusa.
- $\mathbf{n}$ : vetor normal no ponto onde se está avaliando a iluminação.
- $\mathbf{l}$ : vetor normalizado que aponta para a fonte de luz pontual/direcional.

- $\kappa_s$ : coeficiente de reflectância especular.
- $\mathbf{r}$ : reflexão de $\mathbf{l}$ em relação à $\mathbf{n}$.
- $\mathbf{v}$ : vetor normalizado que aponta para a câmera.
- $n$ : tamanho do brilho especular.
- $I_a \kappa_a$ : termo ambiente.
- $I_p \kappa_d \, (\mathbf{n} \cdot \mathbf{l})$ : termo difuso.
- $I_p \kappa_s \, (\mathbf{r} \cdot \mathbf{v})^n$ : termo especular.

### 4.2 *Vertex* e *Fragment Shaders*

Desde o início dos anos 2000 as placas de vídeo contam com unidades programáveis conhecidas como processadores de vértice e de fragmento, ou, em inglês, *vertex* e *fragment shaders*. O *vertex shader* é uma unidade que pode ser programada para alterar atributos de vértices (*e.g.* cor, posição, coordenadas de textura, etc.). O *fragment shader* é uma unidade que pode ser programada para alterar atributos de fragmentos gerados pelo processo de rasterização (*e.g.* cor, valor de profundidade, descarte, etc.).

Quando o modelo de interpolação utilizado é o de Gouraud, normalmente o *vertex shader* fica encarregado de transformar o vértice, originalmente no espaço do objeto, para o espaço de recorte e de avaliar o modelo de iluminação para determinar a cor final do vértice. O *fragment shader*, neste caso, fica encarregado apenas de atribuir ao fragmento o valor de cor interpolado.

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*.

## 5 Desenvolvimento

O programa template fornecido com este exercício renderiza um torus vermelho na tela, como ilustrado na Figura 1.

Figure 1: Renderização do torus utilizando a cor vermelha.

### 5.1 Código Template

O programa template fornecido com o exercício, em seu estado original, executa as seguintes operações:

1. Gera a malha de triângulos de um torus.
2. O *vertex shader* transforma os vértices da malha para o espaço de recorte e os colore com a cor vermelha.
3. O *fragment shader* recebe os fragmentos gerados pela rasterização e aplica a cor interpolada a cada um.

## 5.2 Exercício 1: Implementação do Modelo de Phong usando Interpolação Gouraud

Para a realização da primeira parte do exercício, ou seja, a implementação do modelo de Phong utilizando interpolação Gouraud, os alunos devem alterar o *vertex shader* de forma que este avalie o modelo de Phong (apresentado na Equação 1) para cada vértice. De forma a facilitar a realização do exercício, o *vertex shader* presente no template já contém o cálculo de algumas variáveis necessárias à avaliação do modelo de Phong, a saber:

- `Ia` : intensidade (cor) da fonte de luz ambiente. Equivale ao termo $I_a$ da Equação 1.
- `Ip_position` : posição da fonte de luz no Espaço do Universo.
- `Ip_diffuse_color` : cor do componente difuso da fonte de luz pontual. Equivale ao termo $I_p$ da Equação 1.
- `k_a` : coeficiente de reflectância ambiente do objeto. Equivale ao termo $\kappa_a$ da Equação 1.
- `k_d` : coeficiente de reflectância difusa do objeto. Equivale ao termo $\kappa_d$ da Equação 1.
- `k_s` : coeficiente de reflectância especular do objeto. Equivale ao termo $\kappa_s$ da Equação 1.
- `N_cam_spc` : vetor normal no espaço da câmera. Equivale ao termo $\mathbf{n}$ da Equação 1.
- `L_cam_spc` : vetor da fonte luz no espaço da câmera. Equivale ao termo $\mathbf{l}$ da Equação 1.
- `R_cam_spc` : reflexão do vetor `L_cam_spc` sobre o vetor `N_cam_spc`. Equivale ao termo $\mathbf{r}$ da Equação 1.

Uma vez que a implementação do modelo de iluminação esteja correto, o resultado obtido deverá ser igual ao da Figura 2.

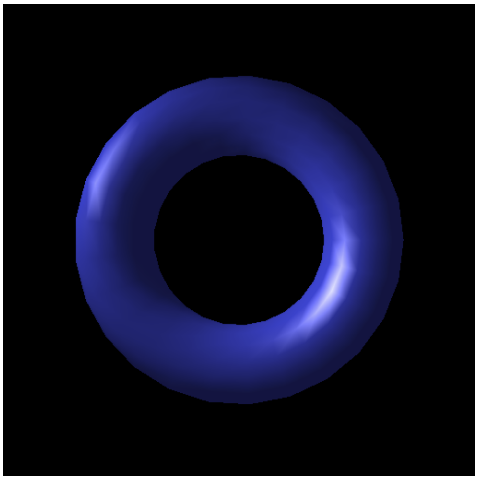

Figure 2: Torus renderizado utilizando o modelo de iluminação de Phong e a interpolação de Gouraud. Essa imagem foi gerada utilizando-se $n = 16$, onde $n$ é o expoente do termo especular.

### 5.3 Exercício 2: Implementação do Modelo de Phong usando Interpolação Phong

Para que a avaliação do modelo de iluminação de Phong seja realizada para cada fragmento, ou seja, utilizando a interpolação de Phong, os cálculos de iluminação devem ser movidos do *vertex shader* para o *fragment shader*.

Sendo assim, o template original deve ser alterado de forma que a realizar as seguintes operações, nesta ordem:

1. Gerar a malha de triângulos de um torus.
2. O *vertex shader* transforma os vértices da malha para o espaço de recorte, mas não realiza o cálculo da cor dos vértices.
3. O *fragment shader* recebe os fragmentos gerados pela rasterização e determina a suas cores finais através da avaliação do modelo de iluminação de Phong.

Uma vez que a implementação do modelo de iluminação esteja correto, o resultado obtido deverá ser igual ao da Figura 3.

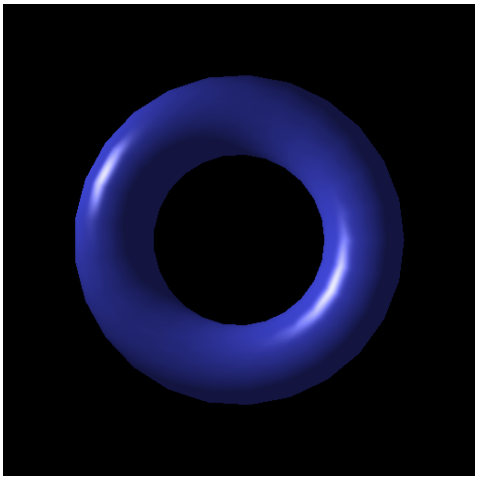

Figure 3: Torus renderizado utilizando o modelo de iluminação de Phong e a interpolação de Phong. Essa imagem foi gerada utilizando-se $n = 16$, onde $n$ é o expoente do termo especular.

## 6 Entrega

Os trabalhos devem ser entregues, via atividade específica do SIGAA, até as **23 horas e 59 minutos** do dia **08/11/2021**. A entrega consistirá em um arquivo compactado (*i.e.* ZIP) contendo um relatório e o código fonte:

1. Relatório no formato PDF, contendo:
   (a) Nome e número de matrícula do(s) alunos(s).
   (b) Um parágrafo que descreva a atividade desenvolvida.
   (c) Breve explicação das estratégias adotadas pelo aluno na resolução da atividade.
   (d) *Printscreens* e discussão dos resultados gerados, dificuldades e possíveis melhoras.
   (e) Referências bibliográficas.

(f) O aluno pode, se assim desejar, disponibilizar seu código fonte também em um repositório *online* (*e.g.* `codepen.io`, `jsfiddle.net`, etc.) e incluir o *link* para este respositório em seu relatório. Entretanto, observa-se que esta disponibilização do código fonte em sites é opcional, não vale nota, e não substitui o envio do código fonte pelo SIGAA.

2. Arquivo contendo o código fonte em JavaScript.

**Não serão aceitos trabalhos enviados por outro meio que não o SIGAA.**

Este trabalho pode ser desenvolvido em duplas.

# Importante:

1. Não serão aceitos trabalhos enviados por outro meio que não o SIGAA.
2. Trabalhos entregues até às **23h e 59min** do dia **08/11/2021** serão contabilizados como presença em aula. Trabalhos que **não forem entregues** até às **23h e 59min** do dia **08/11/2021** serão contabilizados como **falta em aula**, e receberão **nota zero**.