

# Project 3 - Illumination and Shading

## 1. Identificação do grupo:

- Henrique José Maceiro Martins - 67985
- Eduardo Ferreira de Melo - 75157

## 2. Análise dos Shaders

Abaixo descrevem-se as variáveis de entrada e saída dos shaders utilizados para implementar os modelos de iluminação de Phong (fragment) e Gouraud (vertex).

### 2.1. Vertex Shader (`shader.vert`)

#### Variáveis de Entrada (in - Attributes):

- `a_position` (vec4): A posição de cada vértice do objeto no referencial do modelo.
- `a_normal` (vec4): O vetor normal de cada vértice.

#### Variáveis Uniformes (uniform):

- `u_model_view` (mat4): Matriz de transformação ModelView.
- `u_projection` (mat4): Matriz de projeção perspetiva.
- `u_normals` (mat4): Matriz para transformação correta das normais (inversa da transposta da ModelView).
- `u_n_lights` (int): Número atual de luzes ativas na cena.
- `u_shading_mode` (int): define o modo de sombreamento (0 = Phong, 1 = Gouraud).
- `u_lights[MAX_LIGHTS]`: Array contendo as informações das luzes (posição, cores, direção, *cutoff*, abertura).
- `u_material` (struct): Estrutura com as propriedades do material (Ka, Kd, Ks, brilho).

#### Variáveis de Saída (out - Varyings):

- `v_normal` (vec3): Vetor normal interpolado (enviado para o Fragment Shader se Phong).
- `v_viewer` (vec3): Vetor que aponta para o observador (enviado para o Fragment Shader se Phong).
- `v_color` (vec3): A cor final calculada no vértice (apenas utilizado se o modo for Gouraud).

## 2.2. Fragment Shader (shader.frag)

### Variáveis de Entrada (in - Varyings):

- `v_normal` (`vec3`): Vetor normal interpolado vindo do Vertex Shader.
- `v_viewer` (`vec3`): Vetor de visão interpolado.
- `v_color` (`vec3`): Cor pré-calculada (usada apenas no modo Gouraud).

### Variáveis Uniformes (uniform):

- Mantém as mesmas estruturas de `u_lights`, `u_material`, `u_n_lights` e `u_shading_mode` descritas no Vertex Shader para efetuar os cálculos pixel-a-pixel no modo Phong.

### Variáveis de Saída (out):

- `color` (`vec4`): A cor final do fragmento (pixel) a ser renderizada no ecrã.

## 3. Estruturas de Dados

A cena não foi carregada via JSON, mas construída programaticamente.

- **Luzes:** As luzes são geridas através de um array (`lights`). Cada objeto armazena o seu estado (`active`, `type`, `onCamera`), posição geométrica, cor (`intensities`) e parâmetros de `spotlight` (`spotInfo`, `axis`).
- **Câmara:** Representada por um objeto contendo os vetores `eye`, `at` e `up`, além dos parâmetros de projeção (`fovy`, `near`, `far`, `aspect`).

## 4. Funcionalidades Implementadas

O projeto cumpre todos os requisitos base especificados no enunciado e implementa várias funcionalidades classificadas como "Desafios" ou extras.

### 4.1. Requisitos Base

- **Modelo de Iluminação de Phong:** Implementado corretamente com componentes Ambiente, Difusa e Especular.
- **Cena:** Plataforma e 4 objetos (Cubo, Toro, Cilindro e Coelho) posicionados corretamente.
- **Materiais:** O material do Coelho é editável em tempo real, enquanto os outros objetos possuem materiais fixos distintos.
- **Alternância de Shading:** Implementado suporte para alternar entre **Phong** (cálculo no Fragment Shader) e **Gouraud** (cálculo no Vertex Shader) via interface.
- **Spotlights:** Implementação completa com direção central, ângulo de abertura (`aperture`) e fator de atenuação (`cutoff`).

## 4.2. Funcionalidades Extra e Desafios (Challenges)

### 1. Gestão Dinâmica de Luzes (Adicionar/Remover):

- Em vez de um número fixo, o utilizador pode adicionar até 3 luzes e remover luzes dinamicamente.

### 2. Sistema de Coordenadas de Luz (Desafio World e Camera):

- Foi implementado um sistema (Lock to Camera) que permite definir se uma luz está fixa no **Mundo** ou fixa na **Câmara**.
- No Javascript, é feita a transformação vetorial: se a luz estiver no Mundo, a sua posição é multiplicada pela matriz mView antes de ser enviada ao shader. Se estiver na Câmara, é enviada diretamente. Isto permite criar efeitos de "lanterna" (luz segue o observador) ou "poste de luz" (luz estática) na mesma cena.

### 3. Tipos de Luz via Interface (Dropdown):

- Implementação de um seletor para alternar o tipo de luz entre **Point**, **Directional** e **Spotlight**. A lógica ajusta automaticamente o componente w da posição e gera a visibilidade dos controlos da interface (ex: esconde o eixo e abertura se não for Spotlight).

### 4. Navegação "Fly Mode" (Desafio):

- Implementação de movimentação da câmara utilizando o teclado:
  - **W / S:** Mover para frente/trás.
  - **A / D:** Mover para esquerda/direita (Strafe).
  - **Q / E:** Mover para cima/baixo (implementação extra).
- Controlo de orientação ("Look Around") utilizando o rato.

### 5. Botão Reset Camera:

- Funcionalidade para repor instantaneamente a câmara para a posição e orientação iniciais, facilitando a navegação caso o utilizador se perca na cena.

### 6. Botão On/Off por Luz:

- Controlo individual para ligar ou desligar cada fonte de luz sem ter de a remover da cena.