U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

**Computer Graphics**
L.EIC

# TP3 – Illumination and Materials

## Concepts and Practice

Teresa Matos
27 February 2023

# Illumination in scene

In order to visualize the geometries defined previously, we must:

- Define how illumination affects surfaces
- Define scene's illumination
- Define surfaces (geometries) attributes necessary for calculations

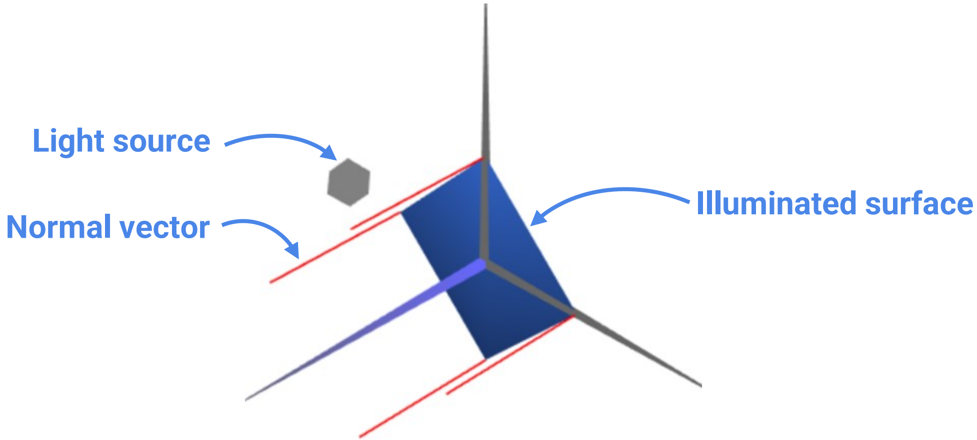


Fig. 1- The "diamond" example object and its normal vectors

# Local Illumination Model

With a **local illumination model**, we can calculate the **resulting color** of a **point** in a surface.

These calculations use attributes of **light sources** and surface **materials**.

The illumination is represented by several **components**:

- **Ambient** component (homogeneous)
- **Diffuse** component (dependent on light source's position)
- **Specular** component (dependent on light source and viewer's position)

# Local Illumination Model in WebCGF

With **WebCGF**, these calculations are performed with the default **shaders**

This uses an improved **local illumination model** with attenuation

The default shader uses **smooth shading** (Gouraud) to determine the color of all points in a surface. This method:

To be presented in next theoretical lesson

- Calculates the color for each **surface vertex**
- Calculates the color for the **remaining points** using bilinear interpolation

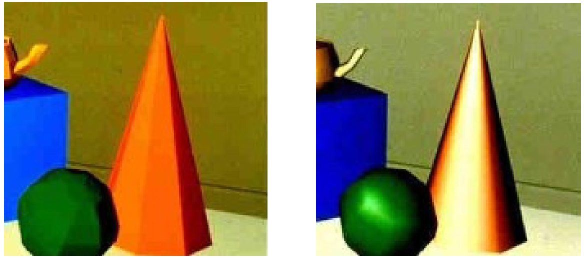

Fig. 2 - Flat and Smooth (Gouraud) shading

# Illumination - Light sources

In the context of this lesson, we will focus on **omnidirectional** light sources

Light sources are positioned in the **3D scene**

The **light intensity** is represented by ambient, diffuse and specular components

For some illumination models, **attenuation constants** may be defined and used

$$I = k_a I_a + f_{att}.[k_d(N.L_{ls}) + k_s.(V.R_{ls})^n].I_{ls}$$

**Fig. 3 – Local illumination model seen in theoretical class**

$$f_{att} = \min\left(1, \frac{1}{k_c + k_l d + k_q d^2}\right)$$

**Fig. 4 – Attenuation factor equation, with constant, linear and quadratic constants**

# Illumination - Light sources in WebCGF

The light sources are represented by the **CGFlight** class

The scene has a dedicated array of lights, pre-created with **eight lights**

In the provided code, the **initLights()** function sets the desired light sources

In the **display()** function, the lights are updated, to apply changes from GUI

```
CGFlight.setPosition(…)
CGFlight.setAmbient(…)
CGFlight.setDiffuse(…)
CGFlight.setSpecular(…)
```

```
CGFlight.setConstantAttenuation()
CGFlight.setLinearAttenuation()
CGFlight.setQuadraticAttenuation()
CGFlight.enable()/disable()
CGFlight.setVisible()
CGFlight.update()
```

# Illumination - Materials

Materials represent appearance-related attributes of the surfaces

The **reflection coefficients** are defined by ambient, diffuse and specular colored components

The **shininess** factor, which affects the specular component, is also defined

Even without light sources, a surface may be visible by defining the **emission** component

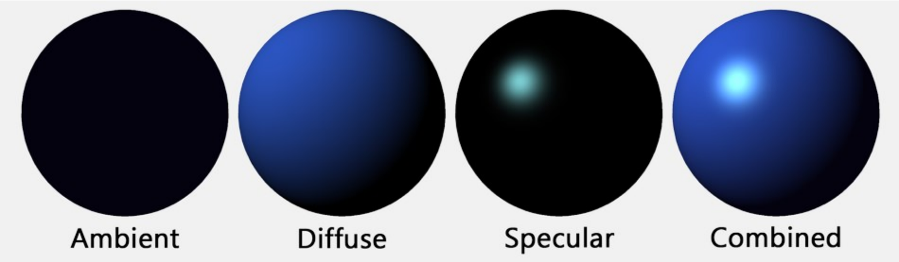


Fig. 5 – Combination of the three illumination components

# Illumination - Materials in WebCGF

Materials are defined using the **CGFappearance** class

Similar to geometric transformations, they are **applied to the scene**

All objects drawn afterwards are **affected by it**

The scene has a **default material** applied in the scene

```
CGFappearance.setAmbient(…)
CGFappearance.setDiffuse(…)
CGFappearance.setSpecular(…)
CGFappearance.setShininess(…)
CGFappearance.setEmission(…)
CGFappearance.apply(…)
```

# Illumination - Surface normal in WebCGF

The **surface normal vectors** are used in the local illumination calculations

These vectors must be defined for **each defined vertex** of the object

The **CGFobject** class has an array for the normal vectors

This array may be filled in the **initBuffers()** function, otherwise it will be filled with vectors = (1,1,1)

These vectors are visible in the scene using **enableNormalViz()** function

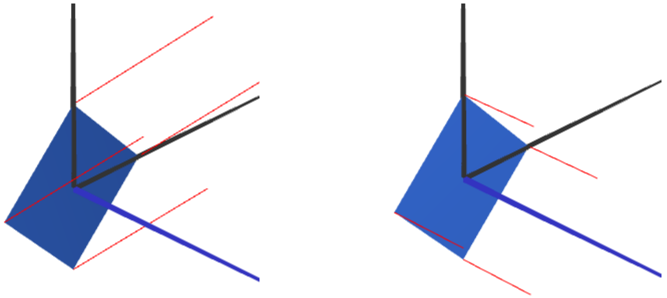

Fig. 6 – Default and corrected normal vectors

# Illumination – Defining normal vectors

The normal vectors are defined in the same order as the vertices

Their value depends on what face the corresponding vertex belongs to

This means that 2+ vertices may have the same coords but different normal vectors (e.g., in a cube, each vertex is used on 3 faces, so 3 different normals)

Vertices[0] is associated to normals[0] and so on

**Group by faces:**

```
vertices = [
0 -1, 0, 0,
1  0,-1, 0,
2  0, 1, 0,
3  1, 0, 0
]
```

These 4 vertices are part of…

**Indices:**

```
indices = [
  0, 1, 2,
  1, 3, 2
]
```

….2 triangles defined by these indices, which are facing…

**Normal vectors**

```
normals = [
0 0, 0, 1,
1 0, 0, 1,
2 0, 0, 1,
3 0, 0, 1
]
```

… the Z+ direction

$\vec{N} = \{0, 0, 1\}$

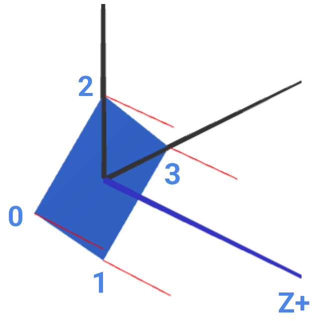


Fig. 7 – Diamond with corrected normal vectors

# Figure references

- Figure 1 and 6 are screenshots of WebCGF scene
- Figure 2, 3, and 4 are from slides of theoretical class 3 and 4 (coming this week)
- Figure 5 obtained from:

https://clara.io/learn/user-guide/lighting_shading/materials/material_types/webgl_materials