# Lab 8

- More than one light source illuminating the scene
- Storing triangle mesh models in text files using a simplified version of the OBJ file format
- Exact representation of polyhedral models vs. Approximate representation of curved models
- Gouraud's method for estimating surface normal vectors
- Analysis, for a given model, of the effect of associating different sets of normal vectors to the mesh vertices
- Material Properties: setting properties for different materials; effects on the final model representations

## 1.1 Several point light sources illuminating the scene

Analyze the incomplete example **WebGL_example_24.html**.

The aim of this example is to use more than one point-light source to illuminate a 3D model, using the Phong illumination model.

Identify the main changes regarding the previous example:

- A class has been defined that allows creating several light sources with different features. Identify the attributes and methods associated with each object instance.

- It is possible to apply different rotation transformations to each light source, which can be animated.

- The instantiated light sources are stored in a global *array*.

- The global structure of the **computeIllumination()** has been changed to allow for more than one light source and to apply rotation transformations to each light source. Verify how a transformation matrix is computed and applied to each light source.

**Questions:**

- What is the location of each light source illuminating the scene? What are their features?
- What kind of movement do they have? Which one is moving faster?
- What is the color of the model? Which material properties were assigned to it?

**Tasks:**

- Visualize the model of the cube and the model of the tetrahedron.
- Change the reflection coefficients and analyze the effects of those changes.

**Task:**

Add more point light sources to the scene:

- A frontal light source, having green light.
- A light source situated below the model and having white light. Make it rotate around the XX' axis.

**Suggestions:**

Add functionalities allowing to:

- Change the reflection coefficients associated with the material defining the model.
- Select and change the color of a light source.
- Select and change the location of a light source, as well as its distance to the scene (directional light source vs. point light source).

## 1.2 Several point light sources illuminating the scene — GPU — *Per Vertex Shading*

Consider now a new version of the previous example, where the illumination computations are carried out on the GPU: **WebGL_example_24_GPU_per_vertex.html**.

**Tasks:**

- Add the same light sources to the scene, as in the previous example.
- For both examples, visually compare the illumination results obtained for models with low level-of-detail.

**Suggestion:**

- Create a scene with four spheres and analyze the illumination effects that are obtained.

## 1.3 Polyhedral surfaces vs. Curved surfaces — The OBJ file format

Analyze the incomplete example **WebGL_example_25.html**.

The aim of the example is to allow reading models stored in text files, using a simplified version of the **OBJ file format**.

An OBJ file usually contains the **coordinates of the vertices** defining a model's surface and the **unit normal vectors** associated with those vertices.

Examine the function that allows reading the files defined in the simplified version of the OBJ file format.

**Task:**

Load the models defined in the files:

- **cubo.OBJ** and **cuboGouraud.OBJ**
- **tetraedro.OBJ** and **tetraedroGouraud.OBJ**

Compare the obtained effects.

Analyze the data stored in each one of those files.

**Task:**

Load the models defined in the files:

- **prismaTriangular.OBJ** and **cilindroAproxPrismaTriang.OBJ**
- **prismaHexagonal.OBJ** and **cilindroAproxPrismaHexagonal.OBJ**

Rotate those models around the YY' axis. Compare the obtained effects.

Rotate them also around the other coordinate axes.

Analyze the data stored in each one of those files.

**Question:**

Can you identify the error in the model defining the triangular prism?

### 1.4 Effects of different material features

**Task:**

Add to the previous example the possibility of assigning different features to the material defining each model.

Use the table in the file **Caracteristicas_Materiais.pdf**.

### 1.5 Effects of different material features — GPU — *Per Vertex Shading*

Consider now a new version of the previous example, where the illumination computations are carried out on the GPU: **WebGL_example_25_GPU_per_vertex.html**.

**Task:**

- For both examples, visually compare the illumination results obtained for models with low level-of-detail.

### 1.6 Effects of different material features — GPU — *Per Vertex Shading*

Consider now the additional example: **WebGL_example_NEW.html**.

It is now possible to instantiate basic 3D models (cube, tetrahedron, sphere) and assign them different geometrical features (position, size, etc.) and material properties.

To accomplish that:

- The file **sceneModels.js** was created and functions defined to **construct** the desired models. A global array contains the instantiated models.

- Some existing functions were adapted: **drawScene()**, **drawModel()** and **initBuffers()**.

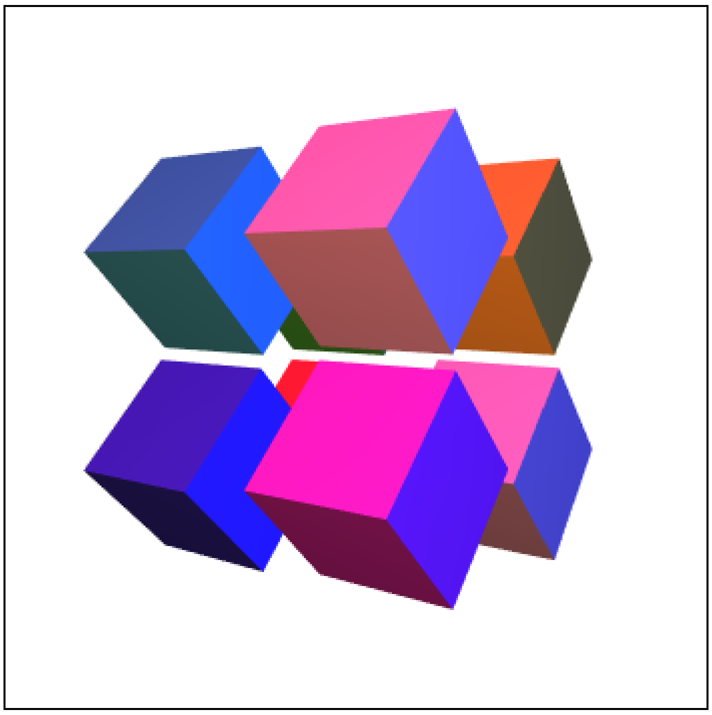
**Task:**

- Analyze the **sceneModels.js** file and verify how the various models are created.

- Note how the parallelepiped model is instantiated from the basic cube model, by changing the scaling factors

**Suggestions:**

- Create a new example: the scene is made up of **four cubes** with different features and positioned as shown. The cubes should rotate around a vertical axis.



- Create another example: the scene is now made up of many models whose shape and features are randomly defined. The figure shows a possible scene.