Computer Graphics
Practical 2

INSA Fourth Year - 2023/2024
Maud Marchal, Glenn Kerbiriou

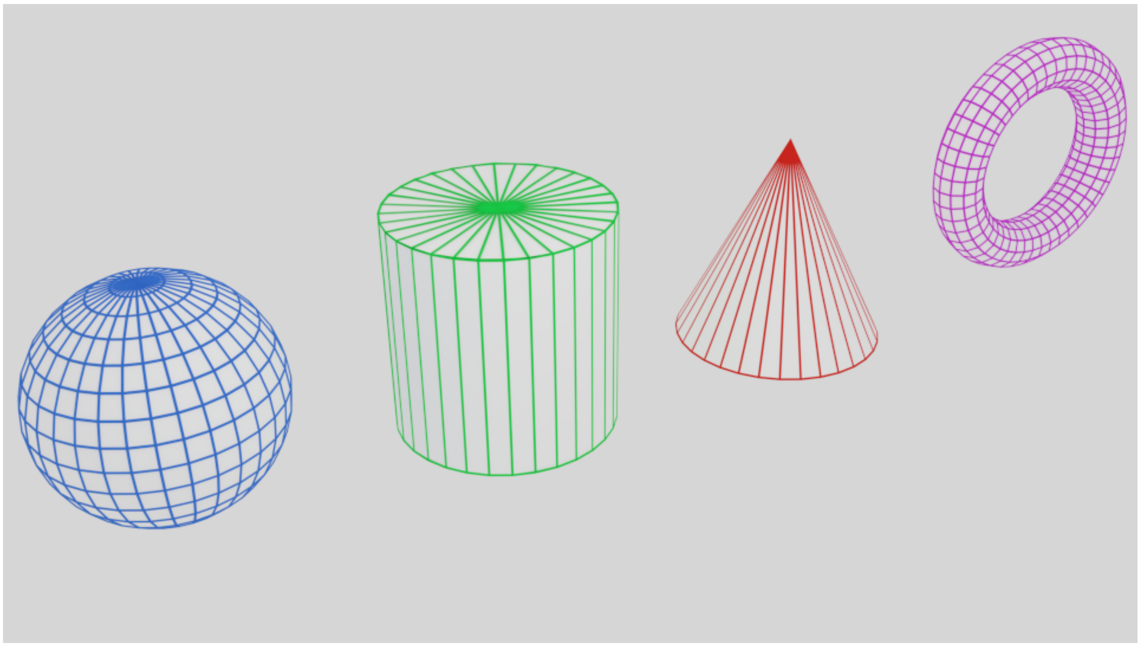
1 Objectives

Computer Graphics is a science mainly composed of three fields:

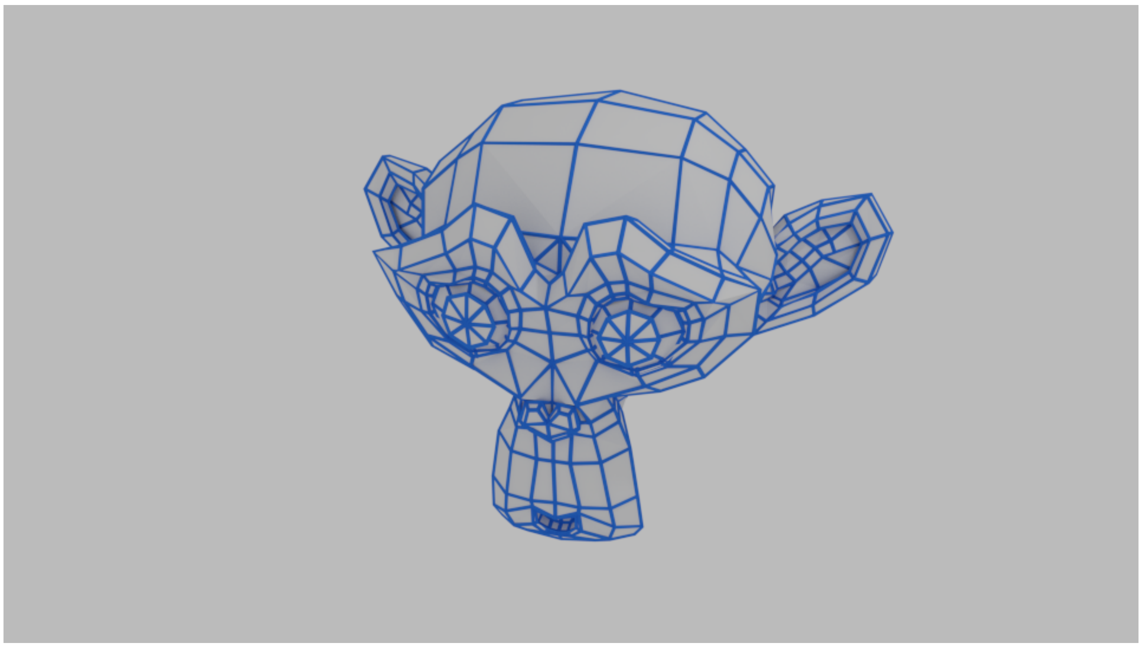
1. **Modeling**: How to create and represent 3D shapes efficiently.
2. **Rendering**: How to compute the appearance of a shape.
3. **Animation**: How to add a dynamic behavior to a shape in a virtual space.

This practical introduces two of the most basic modeling techniques:

- **Parametric modeling**: The surface of an object is described by a set of equations that can be used to build its geometry. This technique is mostly used to describe simple objects that can be then assembled in more complex ones.



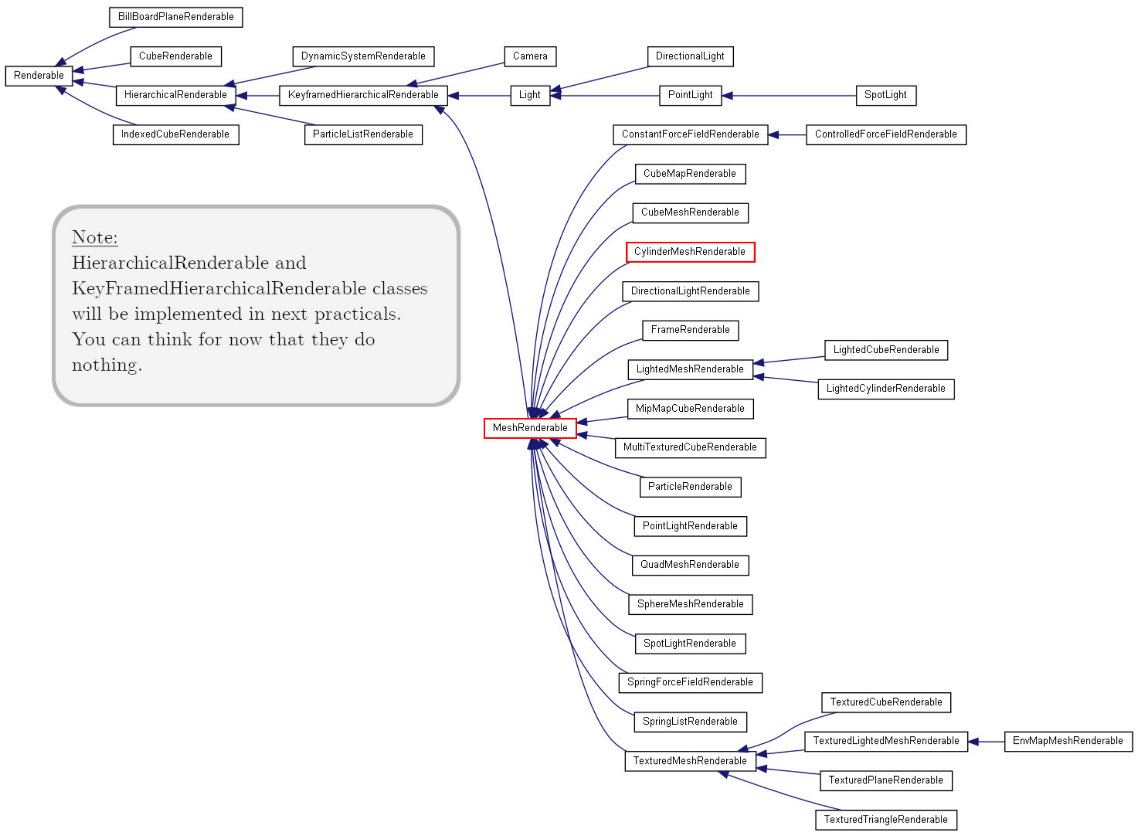
- **Descriptive modeling**: Step by step, vertices and faces are built to describe the surface of an object. Without any sculpting tools, this is certainly the most tedious way of sculpting. Therefore many methods were proposed to ease this process.



2 MeshRenderable class

As you can imagine, the logic that you have implemented in `CubeRenderable` and `IndexedCubeRenderable` can be written once then reused by other Renderables. To do this, we provide the `MeshRenderable` class that gathers positions, normals, colors, indices and texture coordinates buffers.

Carefully read `MeshRenderable::do_draw()`. You will notice that this class implements both indexed or non-indexed geometry and other drawing modes than `GL_TRIANGLES` such as `GL_LINES` (see FrameRenderable.cpp). There is a `protected` constructor which is used by the `CylinderMeshRenderable` class that you will complete in this practical. More generally, you are very likely to derive your own Renderables from `MeshRenderable` unless you need some special rendering behavior (*e.g.* see `ParticleListRenderable::do_draw()`).



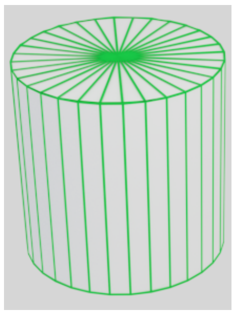
Note:
HierarchicalRenderable and KeyFramedHierarchicalRenderable classes will be implemented in next practicals. You can think for now that they do nothing.

3 Exercice 1: cylinder

The lateral surface of a unit cylinder can be described by the following parametric equations:

$$\begin{cases} x = & \cos\theta \\ y = & \sin\theta \\ z = & h \end{cases}$$

for $\theta \in [0, 2\pi[$ and $h \in [0, 1]$



In the `CylinderMeshRenderable` class, you will create a "canonical" cylinder centered along the z axis, with a radius of 1 and bases on h = 0 and h = 1.

- Identify what number of triangles a slice represents.
- Complete the geometry using the equation of a cylinder.
- Set different colors to the bases and the lateral surface of your cylinder.

- Add your new renderable to the viewer.

Warning: A skeleton of class CylinderMeshRenderable is provided; comments indicate where you need to complete the code.

4 Exercice 2: normals

This exercise is about normal computations. It serves two purposes:

- Getting more familiar with normal computing.
- Preparing the next practical about illumination and rendering.

Compute the normal at each vertex in two different ways:

- **Normal per face**: Each vertex stores the normal of the face.
- **Normal per vertex**: Each vertex stores the normal at its position.

Warning 1: Normals should be **normalized**. It must be ensured by construction or using the `glm::normalize()` function.

Warning 2: So far, normals are not used by your shaders. As a test, you can modify your shaders to take as an input the normals `vNormal` and display them instead of the colors. You may also remap them from $[-1, 1]^3$ to $[0, 1]^3$.

5 Exercice 3: mesh

- Look at the mesh files `cat.obj` and `pillar.obj` with a text editor to get an idea of how a mesh can be stored into a file.
- Use the function in the file `Io.hpp` to import an obj mesh in the class `MeshRenderable`.
- Add two `MeshRenderable`s in the viewer that load the cat and the pillar respectively.
- Use the following functions to place the cat on the pillar.
  - `HierarchialRenderable::setGlobalTransform()`
    (same usage as `Renderable::setModelMatrix()` in Practical 1)
  - `getTranslationMatrix()`
  - `getScaleMatrix()`
  - `getRotationMatrix()`

6 Exercice 4: projet

Start working on the modeling of your boat and North Pole environment.