OpenGL - Lab Session 1

First steps in OpenGL and modeling

During this lab session, we will make our first steps in OpenGL and practice on modeling objects from geometric primitives.

1 Introduction

Download the code provided on the course webpage :

http://www-evasion.imag.fr/Membres/Estelle.Duveau/ICG.html

Open `TPOpenGL.vcproj` and compile and run with `Debug` → `Start Debugging` or hotkey `F5`.

A sphere on a plane should be displayed in shades of blue and red. We will now detail the program (`main.cpp`), study the influence of OpenGL parameters and add new models.

2 First steps

We will first see how the program works.

2.1 GLUT

OpenGL can not create nor manage a viewer. Additional libraries such as `GLUT` are required. In the `main(...)` function of the program, the window is created and initialized and the callback functions are defined with `GLUT`. As you can see, if a keyboard event occurs, the function called will be `specialKey(...)` if it's on a special key (e.g. `F1`, `PageUp`, ...) or `commonKey(...)` otherwise. The function in which the display is done is `renderScene(...)`.

2.2 OpenGL pipeline

The OpenGL pipeline can be modeled as :

1. Create OpenGL context
2. Loop
   (a) Manage mouse/keyboard events
   (b) Display :
      i. Clear screen
      ii. Viewpoint
      iii. Place object, modify primitive parameters and draw primitives

As can be noticed in `main(...)`, the OpenGL context is created with `initScene(...)`. Study this function to understand what it does. We will come back to it more specifically in section 3.

As mentionned previously, the keyboard events management is done in `commonKey(...)` and `specialKey(...)`. There is no mouse events management. We will study it in section 2.4.

As mentionned previously, the display is done in `renderScene(...)`. Let's decipher it.

2.3 Display

For readability, the modification of the primitive parameters and the drawing of the primitives (e.g. `GL_QUAD_STRIP`, `GL_QUADS`, ...) used to draw the objects (e.g. sphere, plane, ...) have been grouped in specific functions (e.g. `drawSphere(...)`, `drawPlane(...)`)

Thus, in `renderScene`, the screen is cleared and for each object, the object is placed and the specific drawing function is called. However, from the OpenGL pipeline outlined previously, the 'viewpoint' step is missing. That is because it is driven by the keyboard and therefore managed during the 'Manage mouse/keyboard events' step.

2.4 Keyboard callback

Indeed, in `specialKey(...)`, some global parameters are modified and the function `setCamera(...)` is called if the keys *left, right, up, down, page up, page down* are pressed. The viewpoint is thus defined in `setCamera(...)`. Test the influence of each of the forementionned keys on the viewpoint.

3 OpenGL parameters

OpenGL is a state machine. As such, it has a predefined number of parameters that have a value at a given time. This value is retained until explicitely changed.

3.1 Modes, booleans and scalars

Depending on its type, the syntax to modify this value changes. Let's take examples in `initScene(...)`

- **Modes :** change the way the color shading is done (flat or smooth) ;
- **Booleans :** disable/enable the depth test, the color for materials, the first light, ... ;
- **Scalars :** change the background color.

3.2 Primitive parameters

Let's focus on some parameters driving the display of primitives (still in `initScene(...)`) :

- **PolygonMode :** This parameter controls the rendering style. Change it to display edges (`GL_LINE`), vertices (`GL_POINT`) or faces (`GL_FILL`).
  The first argument gives the faces on which the rendering style is applied. Render the front faces with points and the back faces filled.
  Now render the front faces with points and, on the next line, with faces. What is displayed? Why?
- **PointSize :** When on point-rendering, change the point size.
- **LineWidth :** When on line-rendering, change the line width.

4 Modeling

For the moment, we only have spheres and planes as high-level primitives. Create new ones in the provided functions `drawX(...)` where X is :

- Cube
- Cylinder
- Cone

Don't forget to specify the color and normal of the vertices.