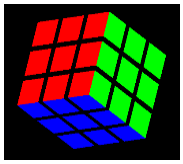
COMP 410/510, Computer Graphics, Spring 2017
Programming Assignment # 2

Rubik's Cube



Due Date: Monday, April 2

This project will give you some experience with interaction and with building, transforming and manipulating 3D models.

Rubik's cube consists of 27 sub-cubes arranged to form a larger 3×3×3 cube. Each of the sub-cubes has sides colored as red, green, blue, white, yellow and magenta. The same-colored faces of the sub-cubes are initially aligned so that the larger cube appears as if it had six solid differently colored faces.

We can move the Rubik's cube by rotating any plane of 9 sub-cubes by 90 degrees. The usual way of playing with Rubik's cube is that one person distorts the original configuration by making a series of 90 degree rotations and the other tries to return the cube to its original state.

The first part of this assignment is to construct a Rubik's cube and to display it. Since we will manipulate the cube in the second part, you should give some thought to how to represent the cube in order to emphasize that it includes 27 instances of a simple cube.

The second part of the project involves manipulating the Rubik's cube. You are required to put it initially in some random configuration by having your program perform a random series of rotations of various faces. Once the cube is initialized, it will be ready for the user to manipulate.

The user should be able to look at the cube from different angles to decide on a move. Thus, the user should be able to rotate the large cube as a whole any time she wishes. You should give some thought to what kind of user interface you want for the user to specify how she wants to look at the cube and what move she would like to perform. You are expected to implement a picking mechanism to decide on which face to rotate. The easiest way to implement picking is to make use of another buffer other than the front buffer for rendering each object with some unique color. See the relevant lecture slides on Input & Interaction (do **not** use the deprecated glRenderMode() functionality). See also Section 3.13 in your textbook to see different alternatives of user interface to rotate objects with mouse (or keyboard).

Your program is not expected to know how to solve Rubik's cube for any initial configuration. Since we are just beginning to talk about viewing, you may simply use orthographic viewing, thus the viewing procedures in the simple sample programs should suffice.

Your grade will depend on how nicely you model the Rubik's cube, both statically and dynamically, the quality of your user interface, and the quality of your display. You should consider things like whether or not you want to display edges of the sub-cubes in addition to the face colors and whether you want to show a move as simply the cube in its final position or as a continuous rotation of smaller angles. However, simply showing the cube without the animation

can be done very trivially and will not necessarily demonstrate the understanding of transformations, which is the key goal of this project.

Implementation hints:

- The most difficult part of this assignment is to be able to keep track of the position of each sub-cube as the user applies a series of rotations of the faces. If you specify an index i (i = 0,1,…,5) for each face, and an index j (j = 0,1,…,26) for each sub-cube, then the indices $\{j\}$ of the 9 cubes belonging to each face i change each time a rotation is applied, depending on which face is rotated.
- Once you determine which face, thus which sub-cubes to rotate, with picking, the next task is to be able perform this rotation properly. For that, you may need to store 28 different modelview matrices, one matrix to store the global view angle, and 27 matrices for the sub-cubes (that is, one matrix for each sub-cube). Of course there may be different ways of achieving the same goal.
- For smooth rotations, you may need glut timer function to adjust the speed of the rotation.
- You are also required to provide a help interface that instructs the user how to use input devices for interaction. The simplest way is to define a keyboard callback function that prints a description to the command line window.

Grading policy:

You are normally capable and expected to accomplish all the tasks necessary to manipulate the Rubik's cube properly. Anyway, please submit your work even if it is partial, but make sure that your code can be compiled and executed and does at least some part of the task coherently. The tentative grading policy is as follows:

- **%35** for initial modeling and display of the Rubik's cube with an interface to view the cube as a whole from different angles.
- **%35** for picking one face and rotating it properly at least once.
- **%35** for smooth animations and capability of applying a series of face rotations (including the random initialization)

Remember that an interface providing help on how to use the program is mandatory. Creative design and implementation will be rewarded. Cheating will not be tolerated. Try to do best you can. Do not leave the assignment to the last minute, start working on it as soon as possible and see how fast you advance.

Submit only the source files, i.e., the project files, ready to be compiled and run. Send the files via e-mail to your TA (and to me as a carbon copy). Ask for an acknowledgement of receipt when you have sent your files.

Bon courage!

Note: You must NOT use any of the deprecated OpenGL functions except GLUT, which were all removed as of version 3.1. Your program **must** include a vertex shader and a fragment shader.