Project 4 - Rubik's Cube

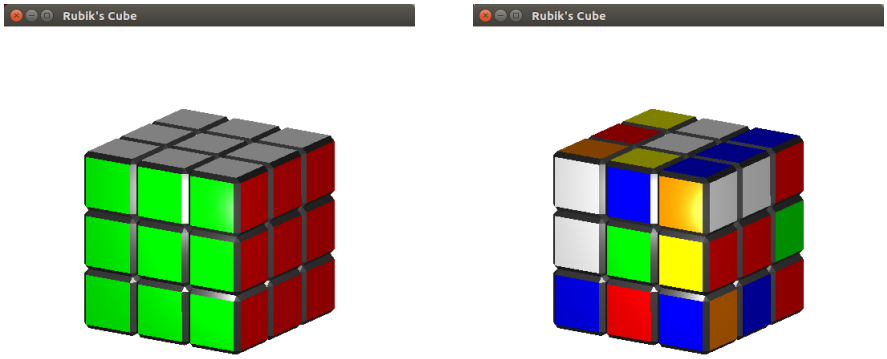
CS 1566 — Introduction to Computer Graphics

Check the Due Date on the CourseWeb

The purpose of this project is for you to use everything you have learned in class.

Introduction to Rubik's Cube

(Wiki) Rubik's Cube is a 3-D combination puzzle invented in by Hungarian sculptor and professor of architecture **Ern Rubik**. Originally called the Magic Cube, the puzzle was licensed by Rubik to be sold by Ideal Toy Corp. in 1980 via businessman Tibor Laczi and Seven Towns founder Tom Kremer and won the German Game of the Year special award for Best Puzzle that year. As of January 2009, 350 million cubes had been sold worldwide, making it the world's top-selling puzzle game. It is widely considered to be the world's best-selling toy.



Front

Right

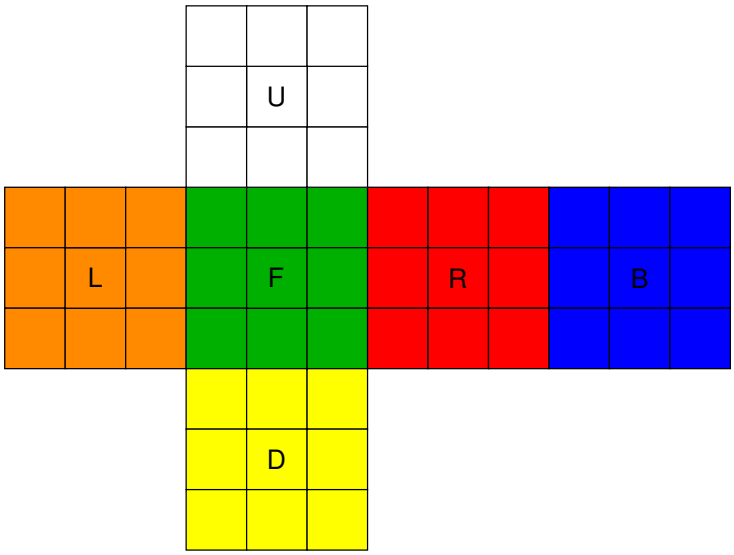
On the original classic Rubik's Cube, each of the six faces was covered by nine stickers, each of one of six solid colors: white, red, blue, orange, green, and yellow. The current version of the cube has been updated to colored plastic panels instead, which prevents peeling and fading. In currently sold models, white is opposite yellow, blue is opposite green, and orange is opposite red, and the red, white, and blue are arranged in that order in a clockwise arrangement. On early cubes, the position of the colors varied from cube to cube. An internal pivot mechanism enables each face to turn independently, thus mixing up the colors. For the puzzle to be solved, each face must be returned to have only one color. Similar puzzles have now been produced with various numbers of sides, dimensions, and stickers, not all of them by Rubik.

Although the Rubik's Cube reached its height of mainstream popularity in the 1980s, it is still widely known and used. Many speedcubers continue to practice it and similar puzzles; they also compete for the fastest times in various categories. Since 2003, the World Cube Association, the

Rubik's Cube's international governing body, has organized competitions worldwide and recognizes world records.

Building a Rubik's Cube (10 Points)

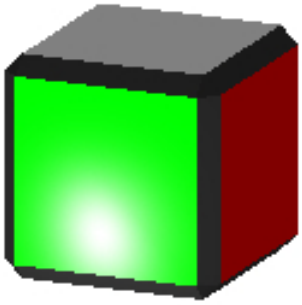
A Rubik's cube consists of 26 cubies. For simplicity, we are going to build it with 27 cubies. Each cubie has unique color(s) on its specific side(s). The following show the color scheme of a Rubik's cube:



The U(p), F(ront), L(eft), R(ight), B(ack), and D(own) indicate the orientation of each face when the green side is facing you and the white side is in the up direction.

For this project, use your code from Project 1 which will allow you to rotate your Rubik's cube to verify that you build it correctly. Note that only a specific side(s) of a cubie has a color. If a side is hidden, the color of that side should be black. For this part:

- 10 points, if you build your Rubik's cube using simple cube (6 sides 36 vertices). It will look very boxy but it will work.
- 20 points, if you build your Rubik's cube using a little more complicated cube. This can be done by having black edges that slopes down from one side to another as shown below:

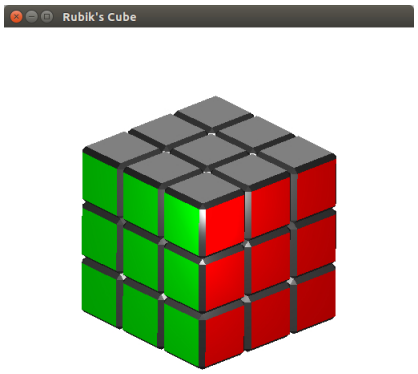


Turn Each Face (30 Points)

For this part, you must implement your application such that a user can turn each face independently using keyboard inputs. There are six faces on a Rubik's cube, Front, Right, Up, Left, Back, and Down. In general, any faces can be the front face. However, the rest of faces depend on how you hold your Rubik's cube. For simplicity, we will use the one shown in the color scheme above as follows:

- Front - green
- Right - red
- Up - white
- Left - orange
- Back - blue
- Down - yellow

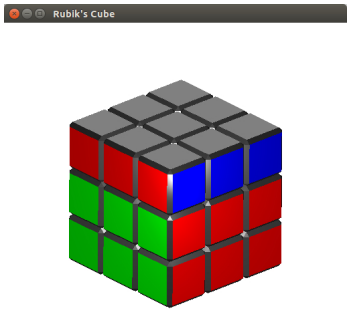
A user will simply use keys `f` (front), `r` (right), `u` (up), `l` (left), `b` (back), and `d` (down) to turn each face of the Rubik's cube. The direction of the turn depends on the left-hand rule where your thumb in the same direction of the face and your four fingers represents the direction of the turn. Starting from a solved Rubik's cube



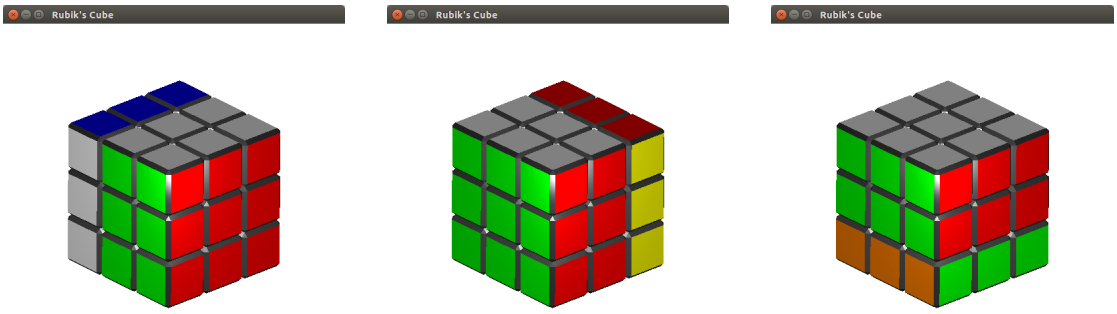
result from each face turn from the solved Rubik's cube are shown below:

Front

Right



Up



Left Back Down

Note that one of the most import part of this Rubik's turning is to keep track of transformation matrix of each cubie. The animation is required for this part. Make sure to animate each turn.

Shuffle the Rubik's Cube (20 points)

Use the key `s` to shuffle the Rubik's cube. You can implement this anyway you want. Make sure to make it turns about 20 to 30 times. **Hint**: use the turn functionality from previous part will help you easily achieve this part. Just simply generate a sequence of random turns. Do not forget about animation so that your rubik's cube will slowly shuffles itself.

Solve the Rubik's Cube (30 points)

The function to solve a Rubik's cube is given. There are two files that must be included as a part of your project as follows:

- `solve_rc.h`: This is the header file that contains signatures of functions that you can use in your project.
- `solve_rc.c`: This is the implementation file of `solve_rc.h`. Be sure to compile it into the object file, if you use `makefile`. For those who use IDE, include this file into your project.

The content of the file `solve_rc.h` is shown below:

```
void r_string_reset();
char *r_string_get();
void r_string_front();
void r_string_right();
void r_string_up();
void r_string_left();
void r_string_back();
void r_string_down();
char *solve_rc();
```

Functions `r_string_front()`, `r_string_right()`, `r_string_up()`, `r_string_left()`, `r_string_back()`, and `r_string_down()` must be called when you make a turn. For example, if a user press `f` to turn the front, the function `r_string_front()` must be called **when the turn is done**. Similarly for the shuffle, make sure to call these functions based on how you shuffle the Rubik's cube. These

function will keep track of the status of the Rubik's cube. This is very important because the program will solve the Rubik's cube based on this status.

The function `solve_rc()` returns a sequence of characters and numbers representing a solution. The following is an example of the string returned by the `solve_rc()` function:

```
U1B1D2R1F3B2R1D3L3B2D3F2L3D1L2B2D1B2D1R2U2B2U3F2U3R2D2B2D2B2U2L2U2
```

Every character will be followed by a one digit number. The character represents the face to be turned and the number represents the number of times. For example `U1` means turn up one time, `D2` means turn down 2 times, and so on. Just simply translate the returned string and use it to control sequence of turns. If you keep track to the status of your Rubik's cube correctly (calling those `r_string_xx()` function, them the solution should always work.

The `r_string_result()` function is used to reset the rubik's status to solved. You do not really need to use this function. If you call `r_string_xx()` functions correctly during solving animation, its status should be back to solved.

Light (10 Points)

For this project, you must incorporate lighting effect into this project. However, we are going to make it simple. Instead of using material the same way as in project 3, you are going to simply use color. In other words, You will have an array of vertices, an array of colors, and an array of normals. Then, inside your vertex shader, simply use normal, light position, and viewer position to adjust the intensity of the color of each vertex before you send it out to the fragment shader. No need to use model view and projection in this project.

Submission

The due date of this project is stated on the CourseWeb. Late submissions will not be accepted. Zip all files related to this project into the file named `project4.zip` and submit it via CourseWeb. After the due date, you must demonstrate your project to either TA or me within a week after the due date.