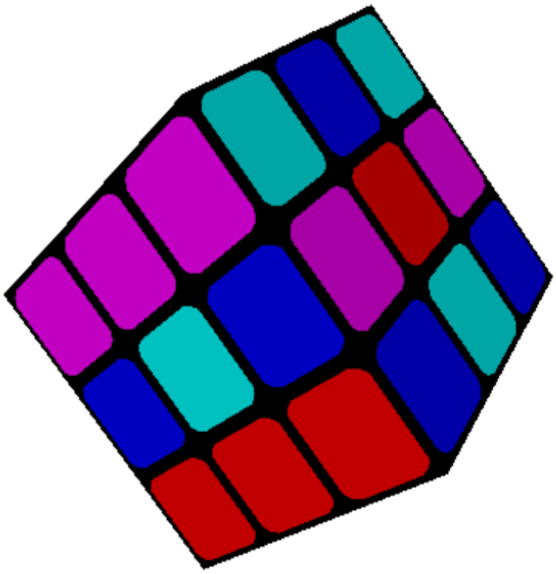
Assignment 2 – Rubic's Cube



In this assignment you will render a dynamic Rubic's cube.

Part 1 - building the cube

1. Go to course site to useful links and download forClass3D.zip
2. Try to compile and run it on your computer (The project was built on VS2015). Try to clean and rebuild it.
3. Use the cube mesh to render a cube. See what happen when you translate and rotate the cube.
4. For building the Rubic's cube you will need to duplicate the cube 26 or 27 times and use translate to move each cube to its initial position. Rotate the cube to see if you did it properly.
5. Change key_callback function in order to rotate the Rubic's cube around x and y axis of the scene using the arrows keys.
6. Change the Fragment Shader so each side of the cube will get its own unique color.

Part 2 – data structures

1. Build data structure of at least 2 matrices (one for translation and one for rotation). This data structure will save the transformations you did for each object.
2. Build data structure for single cube and give each cube an index. You will need to know which of the cube you need to rotate each time when you rotate a wall in the Rubic's cube.
3. Build data structure to all scene which will hold the camera properties and cubes data.
4. Try to implement one wall rotation in the Rubic' cube.

Part 3 – rotations in a row (the difficult part)

1. Add to callback_key function:

a. 'R' press state for right wall rotation (90 degrees clockwise).
b. 'L' press state for right wall rotation (90 degrees clockwise).
c. 'U' press state for up wall rotation (90 degrees clockwise).
d. 'D' press state for down wall rotation (90 degrees clockwise).
e. 'B' press state for back wall rotation (90 degrees clockwise).
f. 'F' press state for front wall rotation (90 degrees clockwise).
g. ' ' press state for flipping rotation direction (from clockwise to counter clockwise or vise versa).
h. 'Z' press state: dividing rotation angle by 2;
i. 'A' press state: multiply rotation angle by 2 (until maximum of 180);

2. Plan where each cube supposes to be after every rotation and how you can follow each cube (hint: use index array).
3. Try to implement few rotations in a row according to your plan.
4. 10 points bonus for implementing 2x2, 4x4, 5x5 Rubic's cube.
5. 10 points bonus for implementation of random mixer and efficient solver. You must be able to display the mixer and the solver action on the screen and write mixer and solver steps sequence to files mixer.txt and solver.txt.

Part 4 – Submission

1. If you did one of the bonuses or both add readme file which explain what you did and how to activate it.
2. Zip your project and rename it to <id1>_<id2>.zip
3. Submit to the submission system before deadline.