Rubik's Cube (version 1.2)

This program is a graphical simulation of Rubik's Cube. Manipulating and solving the cube is just as the real thing.

Installation

The program will run with QB64 GL1.2 and higher. Use the URL to download the zip file and extract the folder “Rubik's Cube” into your QB64 folder. (When extracting the folder, be careful that the extraction method doesn't create a further level of folder with the same name). From the IDE, load the program “Rubik's Cube v1_2.bas” and make sure that you have the Run Option “Save EXE in source folder” checked.

https://www.dropbox.com/s/weos935r9w66fki/Rubik's Cube.zip?dl=0

Using the Program

The program displays a Rubik's Cube. Two views are shown, one from the front and a second (smaller) one from the rear. The rear view is as if you moved around the cube to view it from behind (and is not how you would view the rear-facing sides in a mirror).

When the program has started, press F4 to scramble the cube.

Then use the function keys F1-F3, F5-F7, CtlF1-CtlF3, CtlF5-CtlF7, AltF1-AltF3 and AltF5-AltF7 to rotate and manipulate the cube. On-screen instructions are given as to how these function keys manipulate the cube. How the manipulations take place may seem peculiar to begin with, but it is a question of getting used to which keys operate which manipulation.

If you are unable to solve the cube, press F4 and this will perform that task.

Program Details

In order to simulate a Rubik's Cube, I had two aspects to deal with. Simulating a 3D cube graphically, and writing the logic to know where the colours are and when the cube is solved – when every face has only one colour.

I created the program originally before I had become acquainted with the amazing QB64 _MAPTRIANGLE(3D) which creates a 3D space into which surfaces are placed. This method deals with perspective and handles nearer object occlusion of more distant objects. This would have been exactly what I required. However, in this program I created my own perspective mathematics, and rules such that only forward-facing surfaces are displayed. Also, at every screen update the displayable objects are ordered from furthest to nearest.

The logic part has to determine which colours are on which faces of all the 26 small cubes which make up the complete cube. To deal simultaneously with the graphics and logic is exceedingly complicated and now I come to re-examine this code two years later, I do not exactly

understand what is going on: but it still works! Additionally, I have never actually had a Rubik's Cube in my hands and do not know the solution methods. So, this programs “solves” the cube merely by undoing all the moves and is therefore a great cheat.