A Mini Project Report

on

RUBIK'S CUBE SIMULATION

Submitted by

SHETTY PRAJNESH SHIVANATH
4SN15CS082

VIKYATH K. NAIGA
4SN15CS097

COMPUTER SCIENCE & ENGINEERING

(Visvesvaraya Technological University)

Under the Guidance of

Mr. ARAVIND NAIK

Assistant Professor

Department of Computer Science & Engineering

SRINIVAS INSTITUTE OF TECHNOLOGY

MANGALURU-574143, KARNATAKA

2017 – 2018

SRINIVAS INSTITUTE OF TECHNOLOGY
MANGALURU -574143 – KARNATAKA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled ***"RUBIK'S CUBE SIMULATION",*** *is an authentic record of the work carried out by*

SHETTY PRAJNESH SHIVANATH — **4SN15CS082**

VIKYATH K. NAIGA — **4SN15CS097**

as prescribed by Visvesvaraya Technological University, Belagavi, for VI Semester B.E. in Computer Science & Engineering during the year 2017-2018.

Mr.Aravind Naik	Prof. Shivakumar G.S	Dr. Shrinivasa Mayya D
Project Guide	Head of the Department	Principal

Name of the Examiners **Signature with Date**

1.

2.

ABSTRACT

The aim of this project is to present a simulation of the Rubik's cube in 3D. This project is done in OpenGL using C code. It graphically represents all the basic mechanisms that take place with the Rubik's cube. The graphical representation of the cube consists of six faces, each with 9 colored facets, for a total of 54 facets. The simulation asks for the user to enter the color values of the cube, after which it demonstrates the minimum steps required to solve the cube. Rotation of the cube is done with the help of the keyboard. Overall, the objective is to simulate the cube and help in finding an easier solution to the Rubik's cube.

TABLE OF CONTENTS

LIST OF FIGURES

Acknowledgement

The satisfaction that accompanies the successful completion of any work would be incomplete without thanking the persons who made it perfect with their constant guidance and encouragement.

We take this opportunity to express our profound gratitude and deep regards to our Project guide and mentor**, Mr. Aravind Naik,** Assistant Professor, Department of Computer Science and Engineering**,** for his support and guidance. His vision and suggestions throughout the project period has been fundamental in the completion of the project.

We are highly grateful and would like to express our wholehearted thankfulness to our Project Co-ordinator **Mr. Sudarshan K.,** Associate Professor, Department of Computer Science and Engineering, for his encouragement and timely advice provided to us during the project work.

We extend our warm gratitude to **Prof. Shivakumar G. S,** Head of the Department, Department, Computer Science and Engineering, for his constant support and advice that helped us to complete this project successfully.

We are extremely grateful to our beloved **Principal, Dr. Shrinivasa Mayya D** for his encouragement to come up with new ideas and advice to express them in a systematic manner.

We also like to thank all Teaching & Non-teaching staff of Department of Computer Science and Engineering, for their kind co-operation during the course of our work.

Finally we are extremely thankful to our family and friends who helped us in our work and made the project a successful one.

Shetty Prajnesh Shivanath

Vikyath K.Naiga

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

Computer graphics is a sub-field of computer science and is concerned with digitally synthesizing and manipulating visual content. Although the term refers to three-dimensional computer graphics, it also encompasses two-dimensional graphics and image processing. Computer graphics is often differentiated from the field of visualization, although the two have same similarities. Graphics are visual presentation on some surface like wall, canvas, computer screen. Graphics often combine text, illustration and color.

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube (CRT) screens soon after the introduction of computers. It has grown to include the creation, storage and manipulation of models and images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures. Computer graphics today is largely interactive. The user controls the contents, structure and appearance of objects and of their displayed images by using input devices, such as keyboard, mouse, or touch-sensitive panel on the screen.

Graphical interfaces have replaced textual interfaces as the standard means for user-computer interaction. Graphics has also become a key technology for communicating ideas, data and trends in most areas of commerce, science, engineering and education. Much of the task of creating effective graphic communication lies in modeling the objects whose images we want to produce.

Computer graphics is no more a rarity. Even people who do not use computers in their daily work encounter computer graphics in television commercials and as cinematic special effects. Computer graphics is an integral part of all user interfaces and is indispensable for visualizing objects.

1.2 OpenGL

OpenGL is a low level graphics library specification. It makes available to the programmer a small set of geometric primitives - points, lines, polygons, images, and bitmaps. OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered (drawn). Since OpenGL drawing commands are limited to those that generate simple geometric primitives (points, lines, and polygons), the OpenGL Utility Toolkit (GLUT) has been created to aid in the development of more complicated three-dimensional objects such as a sphere, a torus and even a teapot.

GLUT is designed to fill the need for a window system independent programming interface for OpenGL programs. The interface is designed to be simple yet still meet the needs of useful OpenGL programs. Removing window system operations from OpenGL is a sound decision because it allows the OpenGL graphics system to be retargeted to various systems including powerful but expensive graphics workstations as well as mass-production graphics systems like video games, set-top boxes for interactive television, and PCs. GLUT simplifies the implementation of programs using OpenGL rendering. The GLUT application programming interface (API) requires very few routines to display a graphics scene rendered using OpenGL. The GLUT routines also take relatively few parameters.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 Functional Requirements

This project attempts a 3D animation on Rubik's Cube Simulation using OpenGL. The software has been written using C language and data structures like simple structure type are used.

The project requires the access of OpenGL utility toolkit through the use of the header file "glut.h". This header file, in addition to the usual header files is needed for the working of the project. For running the program, any basic PC running Ubuntu or windows XP (or higher) with compatible version of Microsoft Visual Studio is sufficient.

2.2 Non-Functional Requirements

The software should produce the informative error messages, if any errors are found in the input program. It should use memory as less as possible, dynamic memory allocation is preferable to accomplish this task.

2.3 Hardware Requirements

The minimum/recommended hardware configuration required for developing the proposed software is given below:

- Any processor above 1.2GHz.
- 4GB RAM.
- Approximately 170 MB free space in the hard disk.
- Hard disk access time must be less than 19 milliseconds.
- Standard keyboard and mouse.
- VGA and high resolution monitor.

2.4 Software Requirements

- Open GL library flies (glut.h, glu.h and gl.h).
- WINDOWS XP/7 or any higher versions or Ubuntu Operating system.
- Microsoft Visual Studio for Windows.

2.4.1 Purpose of C Language for the project

C is a minimalist programming language. Among its design goals were that it could be compiled in a straight forward manner using a relatively simple compiler, provided low-level access to memory, generated only a few machine language instructions for each of its core language elements and did not require extensive run-time support. As a result, C code is suitable for many system programming applications that had traditionally been implemented in assembly language.

Despite its low-level capabilities, the language was designed to encourage machine independent programming. A standard compliant and portably written C program can be compiled for a very wide variety of computer platforms and operating systems with minimal change to its source code.

2.4.2 Graphics in C

C itself doesn't support any graphics library. In order for C to be completely portable from platform to platform it has focused on providing platform independent functions, such as file access, text string manipulation, mathematical functions, etc. So in order to get graphics in C, one needs to do one of four things:

- Write a C/C++ program and embed OpenGL primitives for graphics
- Make calls to the Windows API
- Make calls to the OpenGL subsystem
- Use a graphics library

2.4.3 OpenGL subsystem

OpenGL provides a powerful but primitive set of rendering commands, and all higher-level drawing must be done in terms of these commands. Also, OpenGL programs have to use the underlying mechanisms of the windowing system. A number of libraries exist to simplify the programming tasks, including the following:

- The OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections, performing polygon tessellation, and rendering surfaces. The library is provided as part of every OpenGL implementation.
- The OpenGL Utility Toolkit (GLUT) is a window system-independent toolkit. GLUT routines use the prefix glut.

CHAPTER 3

SYSTEM DESIGN

The design of the algorithm is reasonably simple. User gives the input in the input page screen and the cube is constructed according to the color inputs.

3.1 Basic information

Rubik's cube has thousands of solutions with each sequence more complex and complicated than the ones before. There are various algorithms to demonstrate the solution with the least possible moves. More the jumbled color cubes, more is the complexity to solve the Rubik's cube. This project focuses on the solution of the Rubik's cube with more efiiciency. It allows the user to enter the colors of all the 54 facets of the cube and then constructs the cube based on the user's input. The cube is demonstrated as a 3D object with its front, left and upper face visible to the user. The project displays the instruction for rotation of the cube.

3.2 Flowchart

A flowchart is a type of diagram that represents an algorithm, workflow or process. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields. Like other types of diagrams, they help visualize what is going on and thereby help understand a process, and perhaps also find less-obvious features within the process.

The following flowchart represents the various processes that takes place in the Rubik's Cube Simulation.

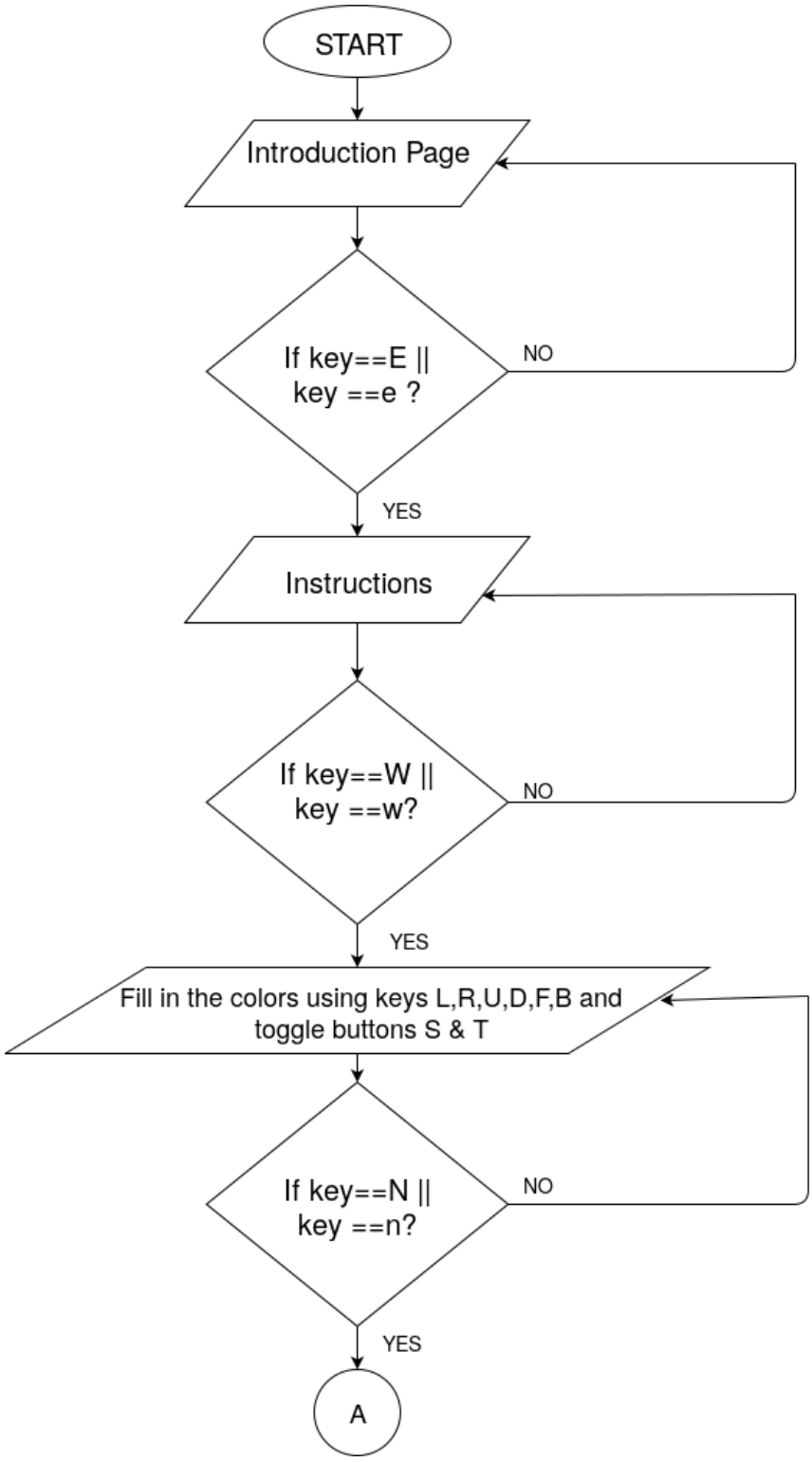

Fig 3.2 : Flowchart of Rubik's Cube Simulation

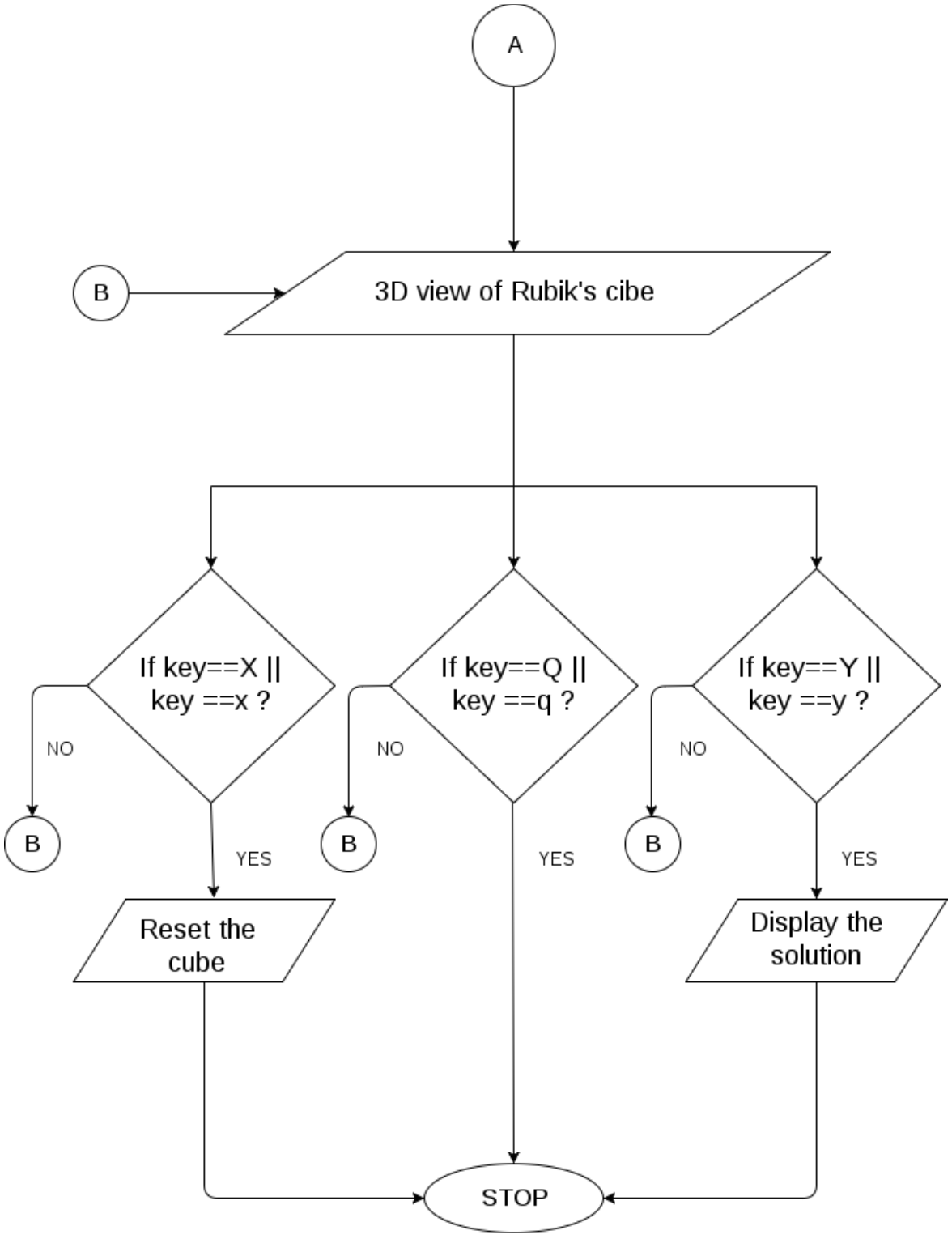

Fig 3.2(b) : Flowchart of Rubik's Cube Simulation (continued)

CHAPTER 4

IMPLEMENTATION

4.1 Functions used in Rubik's Cube Simulation

The Algorithm is constructed in a set of functions. Several user defined functions and built_in functions are used for implementing this project.

The following are the functions used in this project and their purpose: **glutInitDisplayMode (GLUT_DOUBLE|GLUT_RGB):** specifies whether to use an *RGB* or color-index color model. You can also specify whether you want a single- or double-buffered window. (If you're working in color-index mode, you'll want to load certain colors into the color map; use glutSetColor () to do this.) Finally, you can use this routine to indicate that you want the window to have an associated depth, stencil, and/or accumulation buffer.

The Display Callback: glutDisplayFunc (void (*func)(void)) is the first and most important event callback in the code. Whenever GLUT determines the contents of the window need to be redisplayed, the callback function registered by glutDisplayFunc () is executed. Therefore it registers display callback function

Running the program: The very last thing to be done is call glutMainLoop (void). All windows that have been created are now shown, and rendering to those windows is now effective. Event processing begins, and the registered display callback is triggered. Once this loop is entered, it is never exited!

The various other functions used are described in more detail below:

1. **glutBitmapCharacter (font,string[size]);** it renders the character in the named bitmap font and advances the current raster position.
2. **glutInit (&argc, char **argv);** this command initializes GLUT. The argument from main are passed in and be used by the application.
3. **glutInitWindowSize (1400, 900);** this command specifies the initial height width of the window in pixels.
4. **glutCreateWindow ("Name");** creates a window on the display. The string title can be used to label the window.

5. **glutMainLoop ();** causes the program to enter an event processing loop.
6. **cube();** creates a cube according to the specified dimensions.
7. **setColor();** sets the colors of the cube as per the input.
8. **test();** to find the solution of the cube
9. **leftShift();** to move the left side of the cube in clockwise direction.
10. **rightShift();** to move the right side of the cube in clockwise direction.
11. **upShift();** to move the upper side of the cube in clockwise direction.
12. **downShift();** to move the lower side of the cube in clockwise direction.
13. **frontShift();** to move the front side of the cube in clockwise direction.
14. **backShift();** to move the back side of the cube in clockwise direction.

4.2 Algorithm

STEP 1:Start

STEP 2:Display the welcome page

if 'E' or 'e' is pressed then

Start

STEP 3:The instructions are displayed

if 'W' or 'w' is pressed then

goto Input page

STEP 4: Input page is displayed

Use L, R, U, D, F & B to fill the faces and T & S to toggle the color and the position number respectively.

if 'N' is pressed then

goto Construct Cube page

STEP 5:The cube is constructed and the solution of the cube is display as a set of characters

Use l, r, u, d, f, b, L, R, U, D, F & B to rotate the specific parts of the cube.

if 'X' or 'x' is pressed then

Reset the cube back to its original position.

if 'Y' or 'y' is pressed then

Show the user the solution of the Rubik's Cube.

if 'Q' or 'q' is pressed then

goto Step 6

STEP 6:Stop

CHAPTER 5

USER MANUAL

Keys and Actions for the user are given below

Keyboard Keys	Action
l	Move the left side of the cube in clockwise direction.
r	Move the right side of the cube in clockwise direction.
u	Move the upper side of the cube in clockwise direction.
d	Move the lower side of the cube in clockwise direction.
f	Move the front side of the cube in clockwise direction.
b	Move the back side of the cube in clockwise direction.
L	Move the left side of the cube in anticlockwise direction.
R	Move the right side of the cube in anticlockwise direction.
U	Move the upper side of the cube in anticlockwise direction.
D	Move the lower side of the cube in anticlockwise direction.
F	Move the front side of the cube in anticlockwise direction.
B	Move the back side of the cube in anticlockwise direction.
E/e	Move to the instruction page.
W/w	Move to the input page.
S/s	Toggle the color position.
T/t	Toggle the color.
N/n	Display the 3D Rubik's cube with the given input color values.
X/x	Reset the cube back to its original position.
Y/y	Display the solution of the Rubik's cube.
Q/q	Exit the project

CHAPTER 6

SCREEN SHOTS

Fig 6.1: Welcome Screen

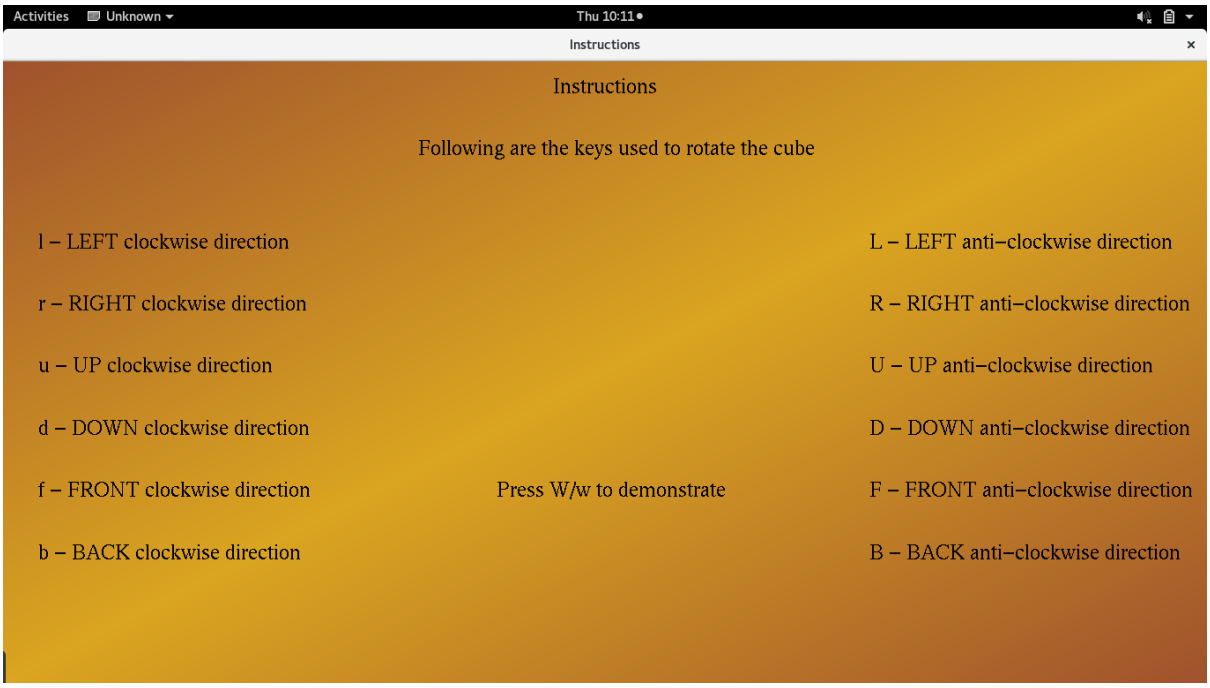

Fig 6.2: Instructions Screen

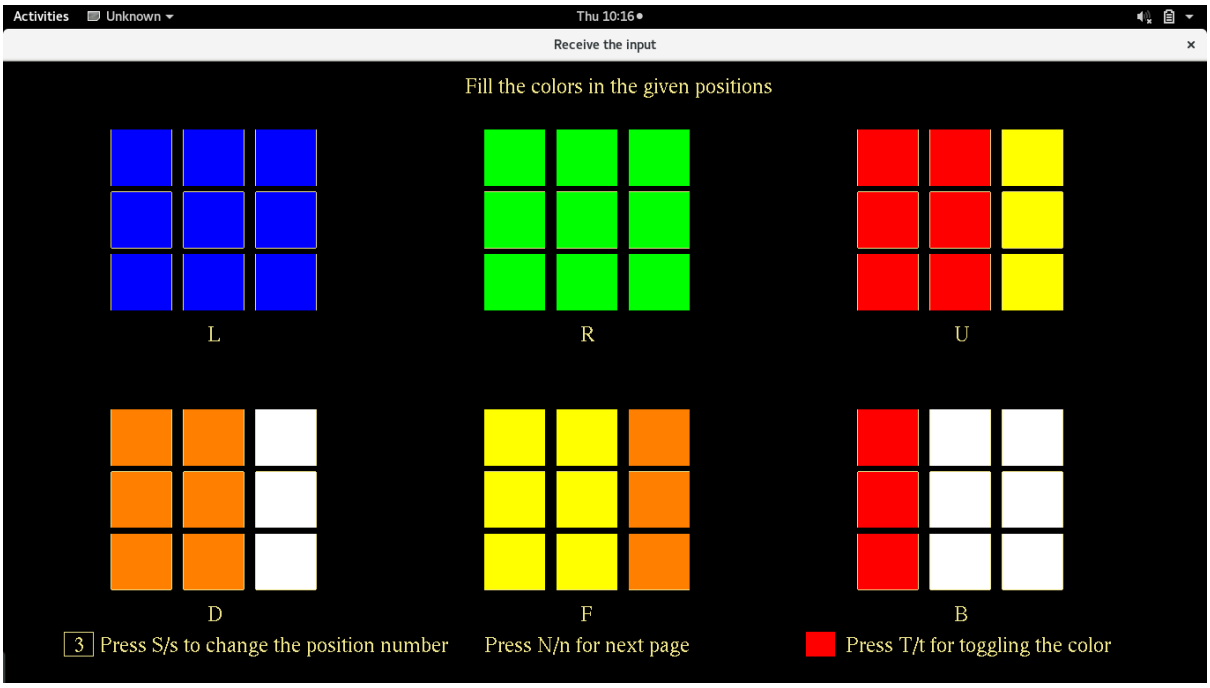

Fig 6.3: Input Screen

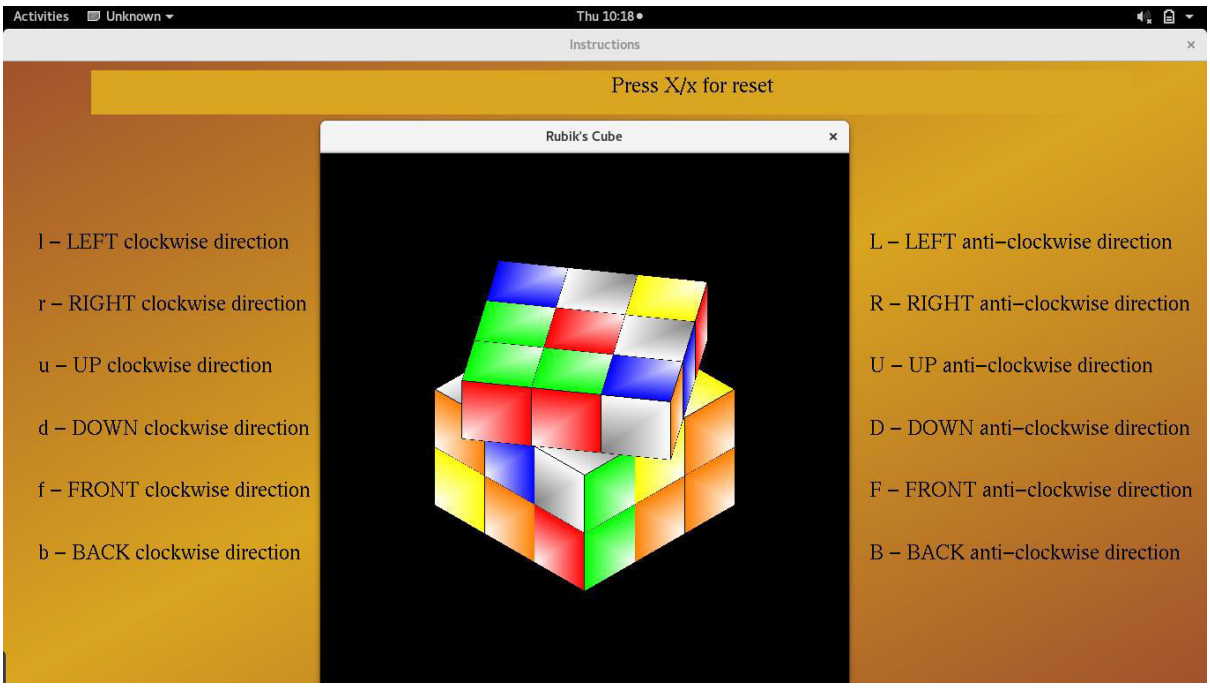

Fig 6.4: Rubik's Cube Screen

CONCLUSION AND SCOPE FOR FUTURE WORK

Designing and implementing Rubik's cube in graphics is a great experience in itself. People will find it more friendly then the real-world cube as well as receive the solution in the project itself.

Rubik's Cube Simulation is an open source project where one can add additional functions to improvise the code. Also the project can have a more efficient algorithm to solve the cube more quickly and with least possible moves. Even more features and good graphics can be added to this application to make it even more interesting and provide good graphical user interface. Developing this application in compatible with the touch screen will provide even more comfortable and easy use.

BIBLIOGRAPHY

[1] Edward Angel, Interactive Computer Graphics A Top-Down Approach with OpenGL, 5th Edition, Addison-Wesley, 2008.

[2] James D Foley, Andries Van Dam, Steven K Feiner, John F Hughes, Computer Graphics, Addison-wesley 1997.

[3] Donald Hearn and Pauline Baker: Computer Graphics- OpenGL Version, 2nd Edition, Pearson Education, 2003.

[4] Thistlethewaite's 52 move algorithm Paper : To solve the Rubik's cube in layers.