VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590014, KARNATAKA

A Mini Project Report

On

"Graphical implementation of Rubik's Cube using Open GL"

Submitted in Partial Fulfillment of the Requirement for

"CG Laboratory with Mini Project -VI Semester"

For the Award of Degree

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By:

AASIM INAMDAR (1SG17CS001)

B PRASHANTH (1SG17CS015)

Under the Guidance of:

Mrs. Anuradha Badage
Assistant Professor

Mrs. Chaithra N C
Assistant Professor

Department of Computer Science and Engineering

SAPTHAGIRI COLLEGE OF ENGINEERING

14/5,Chikkasandra, Hesarghatta Main Road

Bengaluru-560057

2019-2020

SAPTHAGIRI COLLEGE OF ENGINEERING

14/5, Chikkasandra, Hesaraghatta Main Road, Bengaluru – 560057.

Department of Computer Science and Engineering

Certificate

Certified that the Mini Project Work entitled **"Graphical implementation of Rubik's cube using Open GL"** carried out by **AASIM INAMDAR (1SG17CS001)** and **B PRASHANTH (1SG17CS015),** bonafide students of **Sapthagiri College of Engineering**, in partial fulfillment for the award of **Bachelor of Engineering** degree in **Computer Science and Engineering** of **Visvesvaraya Technological University**, **Belagavi** during the academic year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of CG Laboratory with Mini Project (17CSL68) prescribed for the said Degree.

Signature of the Guide	**Signature of the Guide**	**Signature of the HOD**
Prof. Anuradha Badage	**Prof. Chaithra N C**	**Dr. Kamalakshi Naganna**
Assistant Professor	**Assistant Professor**	**Professor& H.O.D**

EXTERNAL EXAMINATION

Name of the Examiners	**Signature with Date**
1._____	_____
2._____	_____

ACKNOWLEDGEMENT

Any achievement does not depend solely on the individual efforts but on the guidance, encouragement and co-operation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this mini project work. We would like to take this opportunity to thank them all.

We would like to express my profound thanks to **Sri. G Dayanand,** Chairman, Sapthagiri College of Engineering, Bangalore for his continuous support in providing amenities to carry out this Mini Project.

Special Thanks to **Sri G D Manoj,** Executive Director, Sapthagiri College of Engineering Bangalore, for his valuable suggestion.

Also we would like to express our immense gratitude to Dr. H Ramakrishna, Principal & Dr. M.H. Annaiah, Vice Principal, Sapthagiri College of Engineering Bangalore, for the help and inspiration during the tenure of the course.

We also extend our sincere thanks to **Dr. Kamalakshi Naganna**, Professor and HOD, Department of Computer Science and Engineering, Sapthagiri College of Engineering, for her constant support.

We would like to express our heartful gratitude to **Mrs. Anuradha Badage,** Associate professor and **Mrs. Chaithra N C,** Assistant professor, Department of Computer Science and Engineering, Sapthagiri College of Engineering, for their timely advice on the mini project and regular assistance throughout the work.

We also extend our sincere thanks to all the **Faculty members** and **supporting staff** Department of Computer Science and Engineering, Sapthagiri College of Engineering, for their constant support and encouragement.

Finally, we thank our parents and friends for their moral support.

AASIM INAMDAR (1SG17CS001)

B PRASHANTH (1SG17CS015)

ABSTRACT

This project demonstrates an operation of Rubik's cube virtually. The Rubik's cube is a physical 3-d combination puzzle. There are a lot of aspects where you would like to solve the Rubik's cube while you're on the go and it is not possible to take away the cube everywhere, thus we have made a computer graphic application to virtually solve the puzzle. The entire 3-D cube can be moved in all the x,y,z directions. The movements of the cube are controlled by using keyboard/ clicks. The cube can be viewed from all the side to solve it. The puzzle is popularly known to boost thinking capabilities and improve the logical reasoning and decision making cababilities.

Here keyboard functions as a primary input device. Events are generated when the particular key is pressed in the keyboards, causing the cube to rotate based on the event specified. The inbuilt GLUT function glutKeyboardFunc is the callback for events generated by pressing the key. When it occurs the ASCII code for the key that generated the event and the location of the mouse are returned. The display function glutDisplayFunc(display) helps redisplay the current output after each event.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Computer graphics are graphics created using computers and more generally, the representation and manipulation of image data by a computer. The term computer graphics has been used in a broad sense to describe "almost everything on computers that is not text or sound".

The development of computer graphics has made computers easier to interact with, better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized animation, movies and the video game industry.

The various applications of computer graphics are

- Graphs and charts
- Computer-Aided design
- Virtual-Reality environment
- Data Visualization
- Education and Training
- Computer Art
- Entertainment
- Image Processing
- Graphical User interfaces

Graphs and Charts:

An early application for computer graphics is the display of simple data graphs, usually plotted on a character printer. Data plotting is still one of the most common graphics applications, but today one can easily generate graphs showing highly complex data relationships for printed reports or for presentations using 35 mm slides, transparencies, or animated videos. Graphs and charts are commonly used to summarize financial, statistical, mathematical, scientific, engineering, and economic data for research reports, managerial summaries, consumer information bulletins, and other types of publications.

Computer Aided Design:

A major use of computer graphics is in design processes—particularly for engineering and architectural systems, although most products are now computer designed.

Generally referred to as CAD, computer-aided design, or CADD, computer-aided drafting and design, these methods are now routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles, home appliances, and a multitude of other products. The manufacturing process is also tied in to the computer description of designed objects so that the fabrication of a product can be automated, using methods that are referred to as CAM, computer-aided manufacturing.

Virtual Reality Environment:

It is a recent application of computer graphics which is used to create virtual-reality environments in which a user can interact with the objects in a three-dimensional scene. Specialized hardware devices provide three-dimensional viewing effects and allow the user to "pick up" objects in a scene. Animations in virtual-reality environments are often used to train heavy equipment operators or to analyze the effectiveness of various cabin configurations and control placements. This allows the designer to explore various positions of the bucket or backhoe that might obstruct the operator's view, which can then be taken into account in the overall tractor design.

Data Visualization:

Producing graphical representations for scientific, engineering, and medical data sets and processes is another fairly new application of computer graphics, which is generally referred to as scientific visualization. The term business visualization is used in connection with data sets related to commerce, industry, and other nonscientific areas. Numerical computer simulations, for example, frequently produce data files containing thousands and even millions of values. Similarly, satellite cameras and other recording sources are amassing large data files faster than they can be interpreted. Other visualization techniques include contour plots, renderings for constant-value surfaces or other spatial regions, and specially designed shapes that are used to represent different data types.

Education and Training:

Computer-generated models of physical, financial, political, social, economic, and other systems are often used as educational aids. Models of physical processes, physiological functions, population trends, or equipment, such as the color-coded diagram in for some training applications, special hardware systems are designed. Examples of such specialized systems are the simulators for practice sessions or training of ship captains, aircraft pilots, heavy-equipment operators, and air traffic-control personnel. Some simulators have no video screens; a flight simulator with only a control panel for instrument flying. But most simulators provide screens for visual displays of the external environment with multiple panels is mounted in front of the simulator.

Entertainment:

Television productions, motion pictures, and music videos routinely use computer-graphics methods. Sometimes graphics images are combined with live actors and scenes, and sometimes the films are completely generated using computer-rendering and animation techniques. Many TV series regularly employ computer-graphics methods to produce special effects, such as the scene in Figure from the television series Deep Space Nine. Some television programs also use animation techniques to combine computer-generated figures of people, animals, or cartoon characters with the live actors in a scene or to transform an actor's face into another shape. And many programs employ computer graphics to generate buildings, terrain features, or other backgrounds for a scene.

Computer Art:

Both fine art and commercial art make use of computer-graphics methods. Artists now have available a variety of computer methods and tools, including specialized hardware, commercial software packages (such as Lumena), symbolic mathematics programs (such as Mathematica), CAD packages, desktop publishing software, and animation systems that provide facilities for designing object shapes and specifying object motions. Example: use of a paintbrush program that allows an artist to "paint" pictures on the screen of a video monitor. A paintbrush system, with a Wacom cordless, pressure-sensitive stylus, was used to produce the electronic painting. The stylus translates changing hand pressure into variable line widths, brush sizes, and color gradations.

Image Processing:

The modification or interpretation of existing pictures, such as photographs and TV scans, is called image processing. In computer graphics, a computer is used to create a picture. Image-processing techniques, on the other hand, are used to improve picture quality, analyze images, or recognize visual patterns for robotics applications. However, image-processing methods are often used in computer graphics, and computer-graphics methods are frequently applied in image processing. Typically, a photograph or other picture is digitized into an image file before image-processing methods are employed. Then digital methods can be used to rearrange picture parts, to enhance color separations, or to improve the quality of shading OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation.

OpenGL

OpenGL has become a widely accepted standard for developing graphics application. Most of our applications will be designed to access OpenGL directly through functions in three libraries. Functions in main GL library have names that begin with the letters gl and are stored in a library usually referred to as GL.

The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with the letters glu.

The third is called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system.

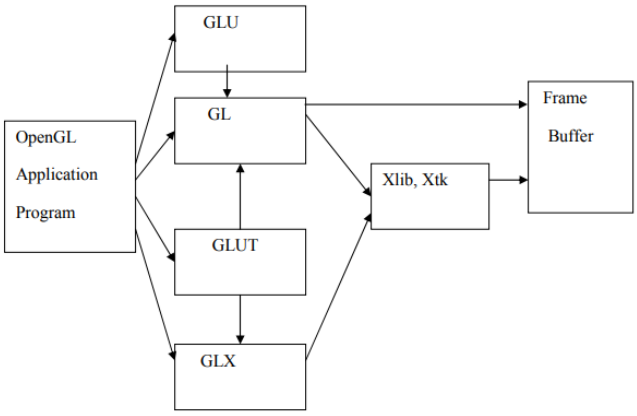

Fig 1.1: Library organization of OpenGL

1.1 Overview of the project

Rubik's Cube is a widely popular mechanical puzzle that has attracted attention around the world because of its unique characteristics. As a classic brain-training toy well known to the public, Rubik's Cube was used for scientific research and technology development by many scholars. This project uses simple logics and programming techniques to form the layout of the animation, thus making it easily understandable for all the programmers.

The animation also holds a wide scope for improvisation by allowing the programmer to make small changes and try out a lot of other implementations of the animation. Thus allowing the programmer to explore more perspectives to the simple animation, by creating a sense of keen interest towards logic development and design.

Since the animation is quite interesting, it proves to be quite addictive to the users. It can also be seen as an aspect of focus improving along with improvements in the user's hand-eye co-ordination.

Application

Rubik's Cube also has some applications in the field of psychotherapy. Rubik's Cube has gradually been taken seriously in the field of education because it contains rich scientific knowledge. Some primary and secondary schools carry out cube-based mathematics courses to improve students' learning ability. There are also some relevant studies showing that the use of "play cubes" as a teaching guide in primary school mathematics activities is conducive to stimulating students' interest in learning and guiding students to think abstractly and develop spatial ideas. Rubik's Cube was viewed as a tool in applying rather sophisticated mathematics to generate some solution algorithms.

In addition, there are some ideas on the application of the cube in aerospace, such as a combination of the cube structure and a satellite, using the cube satellite to drive the cube unit location and orientation reconstruction in order to complete a variety of tasks. Zhao proposed an application design of deformable magic-square-type deep-space exploration aircraft, applying the magic structural features of modularity and rotatability to a spacecraft, so that each box of the cube has a clear division of lab, and then through cooperation to complete the deep space exploration task. Inspired by the ubiquitous Rubik's Cube toy, an equipment-free method was proposed for fabricating paper analytical devices.

1.2 Aim of The Project

The aim is to develop a Rubik's cube virtually, using computer graphics that helps us to create the available puzzle in an interactive visual manner. Having a virtual device or blueprint of a model helps in visualizing the model before it can be built. The main aim of the project is to demonstrate how a Rubik's cube can be solved in a virtual space using computer graphics. Rubik's cube developed can manipulated in ways possible to solve the puzzle in mathematical and scientific manner. We can rotate it in different views - top, bottom, left, right, etc. using certain defined keywords. The Rubik's cube can be viewed from all the angles; it can be operated and rotated to solve the puzzle.

CHAPTER 2

REQUIREMENT SPECIFICATION

A software requirement definition is an abstract description of the services which the system should provide, and the constraints under which the system must operate. It should only specify the external behavior of the system.

2.1 Functional requirements

In software engineering, a **functional requirement** defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases.

The various methods used in this project are as follows:-

- **Display**
  The module draws the output on the screen and the functions in it.
- **Menu**
  This module specifies the action corresponding to menu entry.
- **Keyboard**
  The module specifies the action corresponding to the key board.
- **Mouse**
  This module specifies the action corresponding to the mouse.
- **Idle**
  This module is used to display the object more times using some delay t.

2.2 Non-functional requirements:

These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process and standards. Non-functional requirements often apply to the system as a whole.

Non-Functional Requirements are as follows:-

2.2.1 Dependability:

The dependability of a computer system is a property of the system that equates to its trustworthiness. Trustworthiness essentially means the degree of user confidence that the system will operate as they expect and that the system will not 'fail' in normal use.

2.2.2 Availability:

The ability of the system to deliver services when requested. There is no error in the program while executing the program.

2.2.3 Reliability:

The ability of the system to deliver services as specified. The program is compatible with all types of operating system without any failure

2.2.4 Safety:

The ability of the system to operate without catastrophic failure. This program is user friendly and it will never effects the system.

2.2.5 Security:

The ability of the system to protect itself against accidental or deliberate intrusion.

2.3 Details of the software

Here, the coding of our project is done in Microsoft Visual C++ which is a commercial integrated development environment (IDE) with OpenGL (Open Graphics Library) which is a standard specification to produce 2D and 3D computer graphics. We use, the OpenGL Utility Toolkit called GLUT which is a library of utilities for OpenGL programs.

2.3.1 Microsoft Visual C++

Microsoft Visual C++ is a commercial integrated development environment (IDE) product engineered by Microsoft for the C, C++ and C++/CLI programming languages. It has tools for developing and debugging C++ code, especially code written for the Microsoft Windows API, OpenGL API, the DirectX API and the Microsoft .NET Framework.

2.3.2 OpenGL and GLUT

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics, describing a set of functions and the precise behaviors that they must perform. From this specification, hardware vendors create implementations - libraries of functions created to match the functions stated in the OpenGL specification, making use of hardware acceleration where possible. Hardware vendors have to meet specific tests to be able to qualify their implementation as an OpenGL implementation.

GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API)for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works across all PC and workstation OS platforms.

2.4 Software requirements

- Operating system : Windows 7 and further
- Tools : Microsoft Visual Studio 2010 and newer
- OpenGL 2.0
- Graphics Library : glut.h

2.5 Hardware requirements

- Processor : Pentium Pro
- Memory : 128MB RAM
- 40GB Hard Disk Drive

CHAPTER 3

DESIGN

Data flow design is as shown below - covering the flow of the data in the system. It describes the relation between user input and the system behavior.

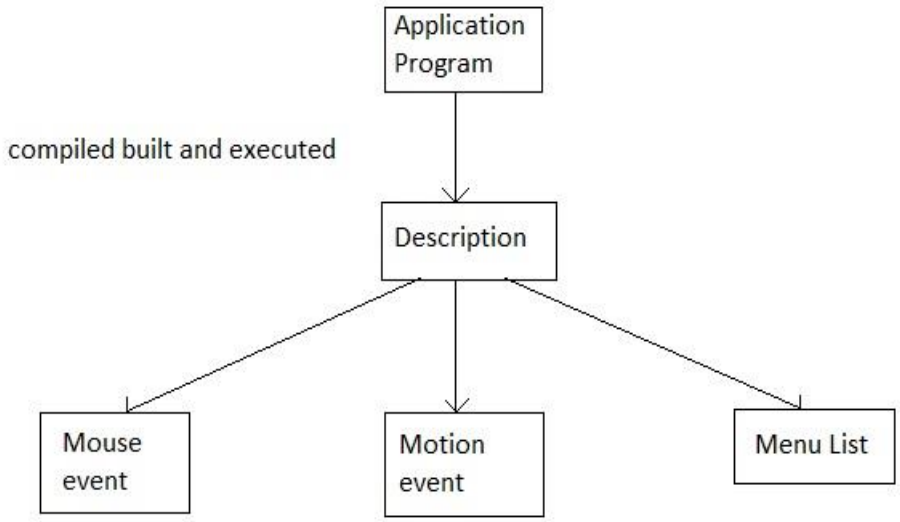

Figure No. 3.1 Flow chart for Representation of Rubik's Cube algorithms.

CHAPTER 4

IMPLEMENTATION

To implement the Current system we have used different functions of our project which are as follows:

4.1 USER DEFINED FUNCTIONS

- **main() :** The execution first starts with the main() function. It's an entry point for GLUT OpenGL application.
- **Output()** : It is used to position pixel and bitmap write operations.
- **Polygon()** : To draw cube using this function.
- **Speedmeter()** : Display the duration / speed of solving.
- **Transpose()** : To make an inverted spin of the cube.
- **Spincube()** : Displays the spin of the cube when an instruction is passed - right, left, top, etc.
- **Myreshape()** : Function is called when window size is changed.
- **Motion()** : To move the face of the cube.
- **topc()** : To make top inverted movement.
- **bottomc()** : To make bottom inverted movement.
- **frontc()** : To make front inverted movement.
- **backc()** : To make back inverted movement.
- **rightc()** : To make right inverted movement.
- **leftc()** : To make left inverted movement.
- **Colorcube1()** : To give colour and shape to centre piece.
- **Colorcube2()** : To give colour and shape to bottom centre piece.
- **Colorcube3()** : To give colour and shape to left centre piece.
- **Colorcube4()** : To give colour and shape to right centre piece.
- **Colorcube5()** : To give colour and shape to top centre piece.
- **Colorcube6()** : To give colour and shape to front centre piece.
- **Colorcube7()** : To give colour and shape to back centre piece.
- **Colorcube8()** : To give colour and shape to top left centre piece.
- **Colorcube9()** : To give colour and shape to top right centre piece.
- **Colorcube10()** : To give colour and shape to top front centre piece.
- **Colorcube11()** : To give colour and shape to top back centre piece.

- **Colorcube12()** : To give colour and shape to bottom left centre piece.
- **Colorcube13()** : To give colour and shape to bottom right centre piece.
- **Colorcube14()** : To give colour and shape to bottom front centre piece.
- **Colorcube15()** : To give colour and shape to bottom back centre piece.
- **Colorcube16()** : To give colour and shape to top left back piece.
- **Colorcube17()** : To give colour and shape to top left front piece.
- **Colorcube18()** : To give colour and shape to top right back piece.
- **Colorcube19()** : To give colour and shape to top right front piece.
- **Colorcube20()** : To give colour and shape to centre left back piece.
- **Colorcube21()** : To give colour and shape to centre left front piece.
- **Colorcube22()** : To give colour and shape to centre right back piece.
- **Colorcube23()** : To give colour and shape to centre right front piece.
- **Colorcube24()** : To give colour and shape to bottom left back piece.
- **Colorcube25()** : To give colour and shape to bottom left front piece.
- **Colorcube26()** : To give colour and shape to bottom right back piece.
- **Colorcube27()** : To give colour and shape to bottom right front piece.

4.2 BUILT IN FUNCTIONS

- **voidglClear (GLbitfield*mask*);**
  - ***mask*** – Bitwise OR of masks that indicate the buffers to be cleared. The four masks are GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT, GL_ACCUM_BUFFER_BIT and GL_STENCIL_BUFFER_BIT.
  - It clears buffers to preset values.

- **voidglClearColor (GLclampf*red,* GLclampf*green,* GLclampf*blue,* GLclampf*alpha*);**
  - *red, green, blue, alpha* – specify the red, green, blue and alpha values used when the color buffers are cleared. The initial values are all 0.
  - It specifies clear values for the color buffers.

- **void glColor3f (GLfloat*red,* GLfloat*green,* GLfloat*blue*);**
  - *red, green, blue* – specify new red, green, and blue values for the current color.
  - It sets the current color.

- **glutCreateWindow (char *_name_);**
  - *name* – ASCII character string for use as window name.
  - It creates a top-level window.

- **voidglutDisplayFunc (void (*_func_) (void));**
  - *func* – the new display callback function.
  - It sets the display callback for the current window.

- **voidglutInitWindowSize (int_width_, int_height_);**
  - width– width in pixels.
  - height – height in pixels.
  - It is used to set the initial window size.

- **voidglutMainLoop (void);**
  - It enters the GLUT event processing group. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

- **voidglutPostRedisplay (void);**
  - It marks the current window as needing to be redisplayed.

- **void display (void);**
  - It contains the function definition for display callback.

- **voidglutIdleFunc (void (*_func_) (void));**
  - It sets the global idle callback.

- **glPushMatrix();**
  - void glPushMatrix(void) glPushMatrix pushes the current matrix stack down by one level, duplicating the current matrix.

- **glPopMatrix();**
  - void glPopMatrix(void) glPopMatrix pops the top matrix off the stack, destroying the contents of the popped matrix. Initially, each of the stacks contains one matrix, an identity matrix.

CHAPTER 5

TESTING

Testing has been conducted as tabulated below.

No	Functions with parameters under test	Expected result	Actual result	Comments
1	All the output statement must be in there position mentioned	The object and particle drop should appear at the window	The object and particle appeared.	PASS
2	Moving function	The Object has to be Moved from one location to desired location	The object Moved.	PASS
4	Key Function	When 'a' is pressed, the top portion of the cube rotates. When 'q' is pressed, the top portion of the cube rotates inverted. When 's' is pressed, the right portion of the cube rotates. When 'w' is pressed, the right portion of the cube rotates inverted. When 'd' is pressed, the Front portion of the cube rotates. When 'e' is pressed, the Front portion of the cube rotates Inverted. When 'f' is pressed, the Left portion of the cube rotates. When 'r' is pressed, the Left portion of the cube rotates	The keys has to work properly, has per menu.	PASS

		Inverted. When 'g' is pressed, the Back portion of the cube rotates. When 't' is pressed, the back portion of the cube rotates Inverted. When 'h' is pressed, the bottom portion of the cube rotates. When 'y' is pressed, the bottom portion of the cube rotates Inverted.		

CHAPTER 6

SNAPSHOTS

6.1 Shuffled Cube

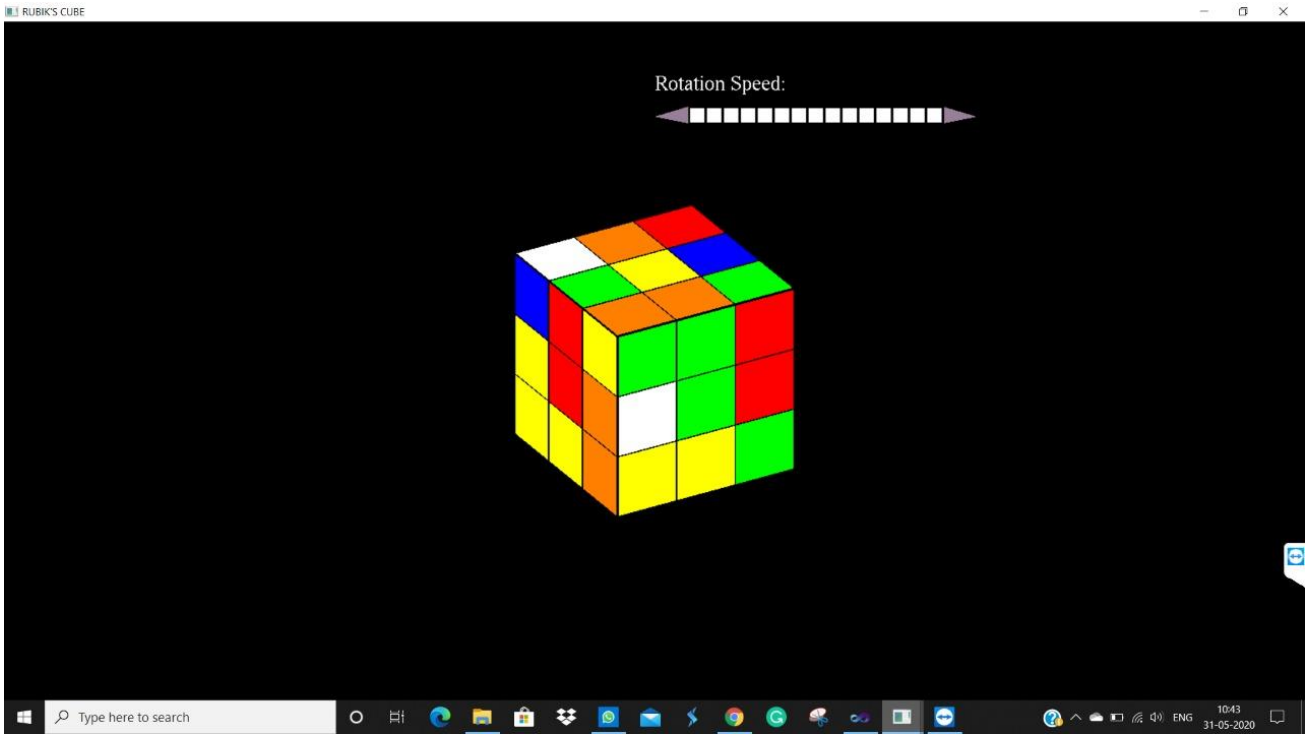

Figure 6.1: Shuffled Cube

6.2 Front Rotation

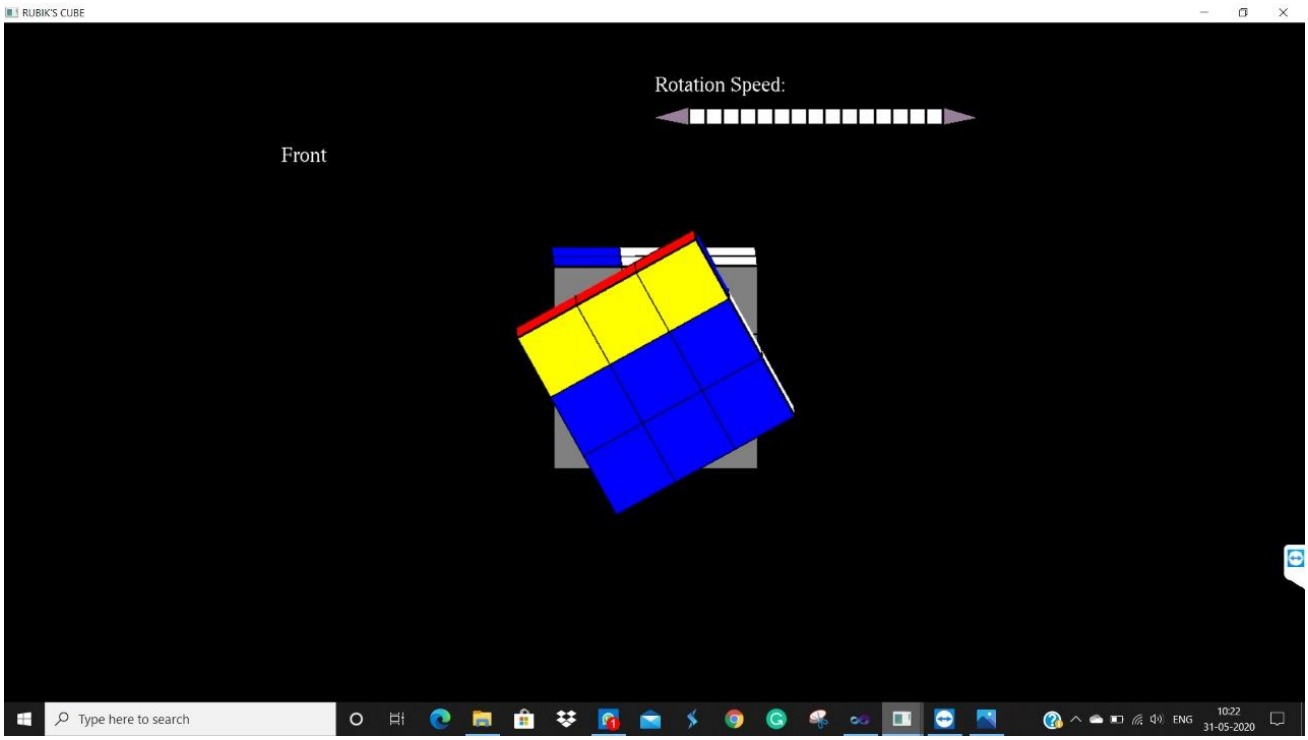

Figure 6.2: Front Rotation of the Cube

6.3 Front Inverted Rotation

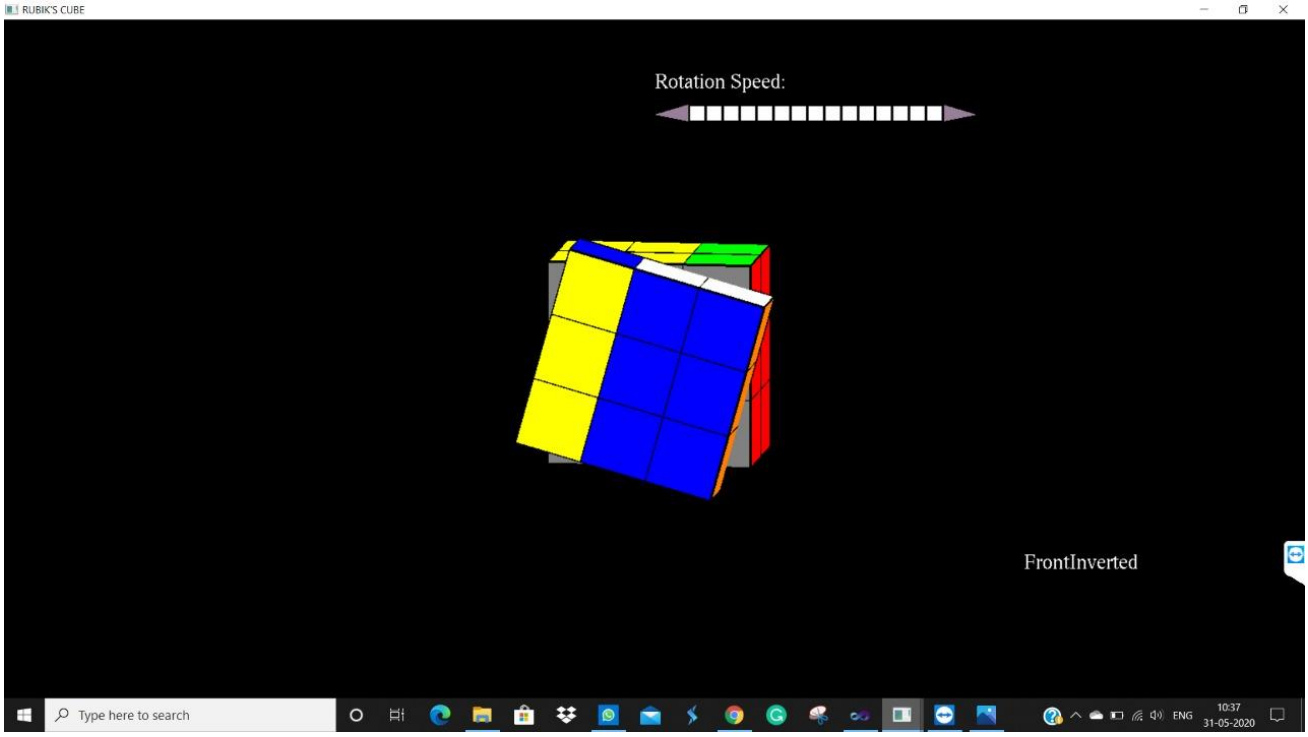

Figure 6.3: Front Inverted Rotation of the Cube

6.4 Back Rotation

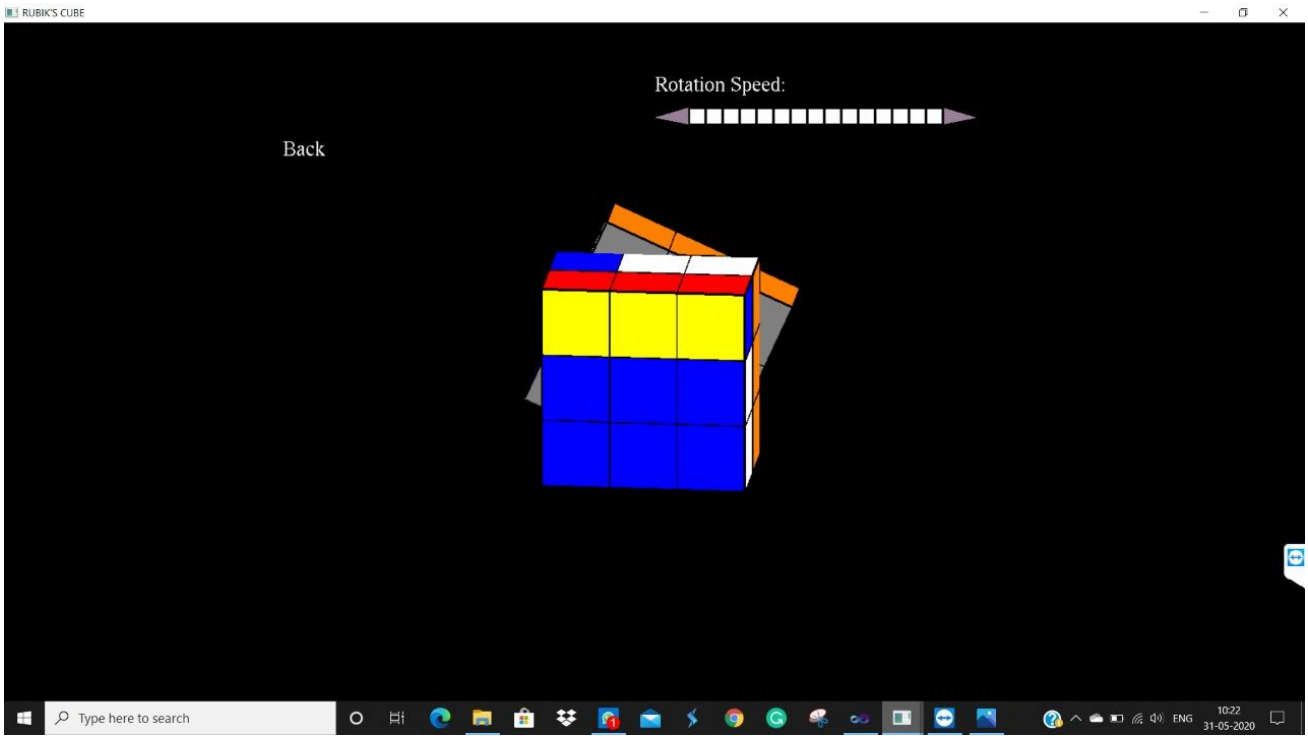

Figure 6.4: Back Rotation of the Cube

6.5 Back Inverted Rotation

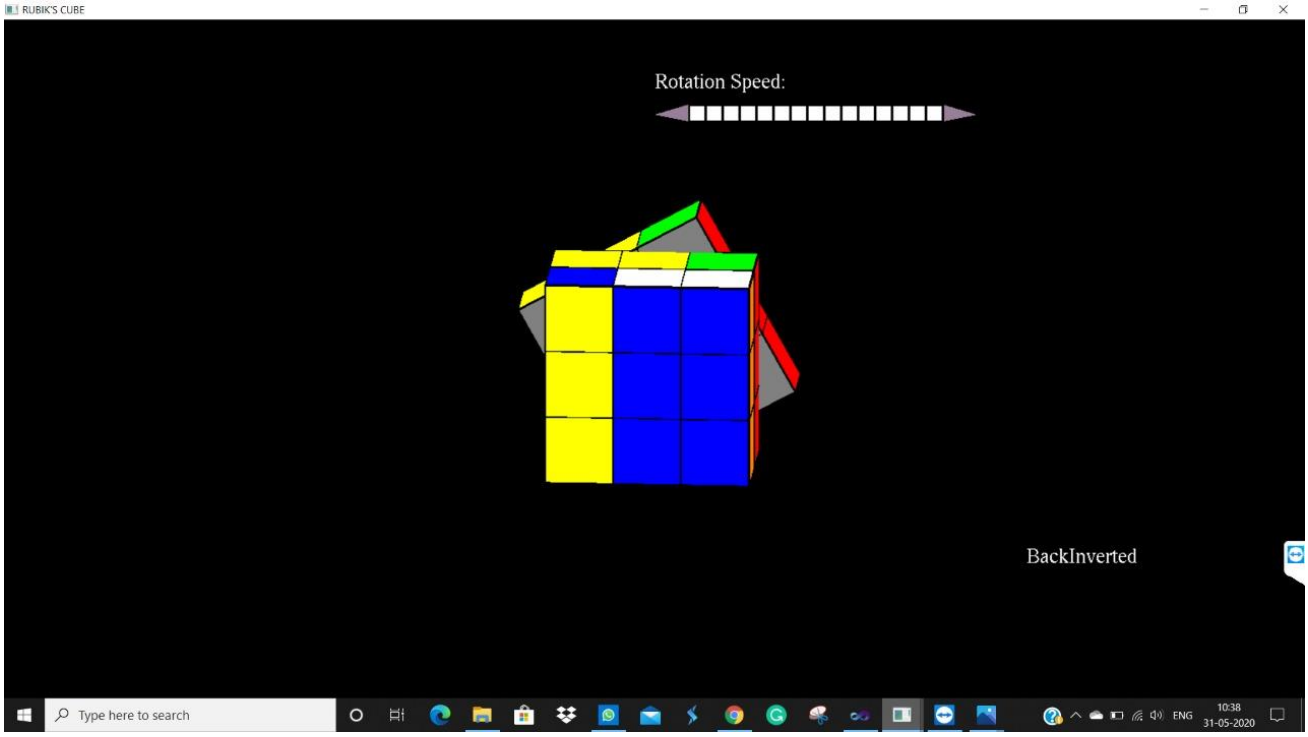

Figure 6.5: Back Inverted Rotation of the Cube

6.6 Top Rotation

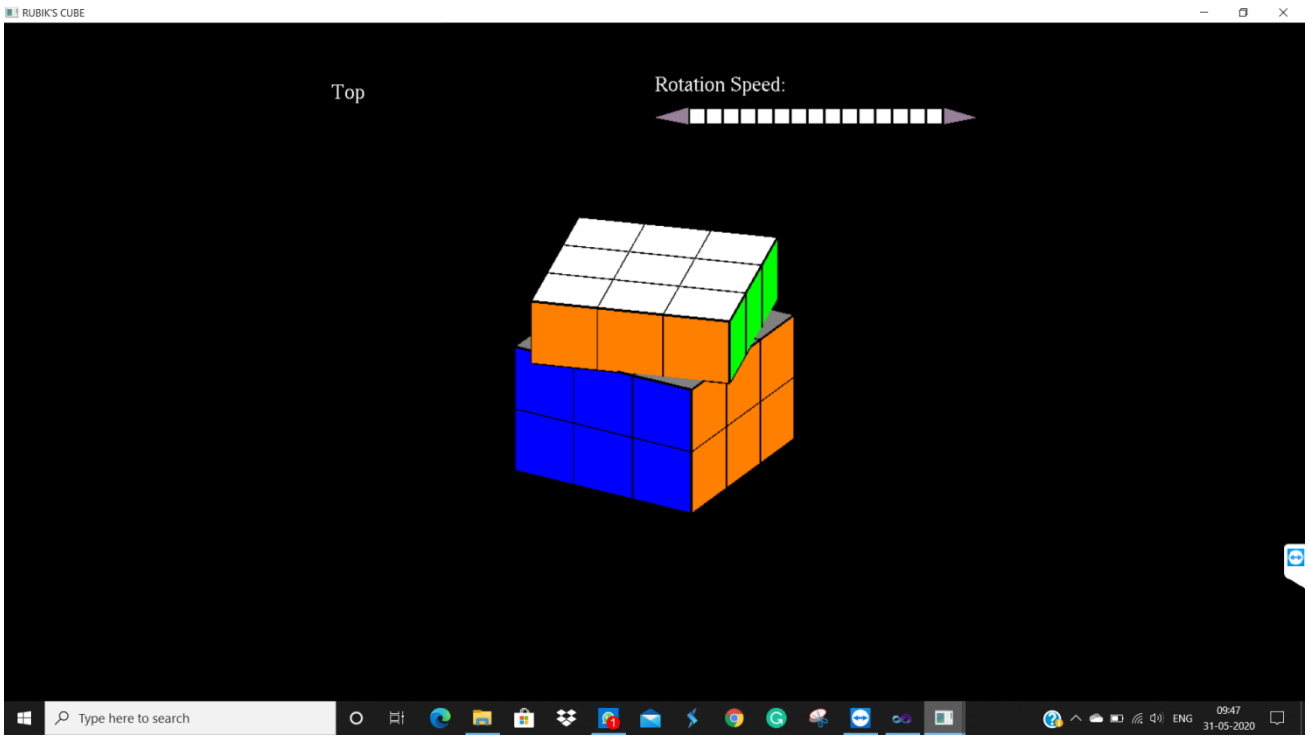

Figure 6.6: Top Rotation of the Cube

6.7 Top Inverted Rotation

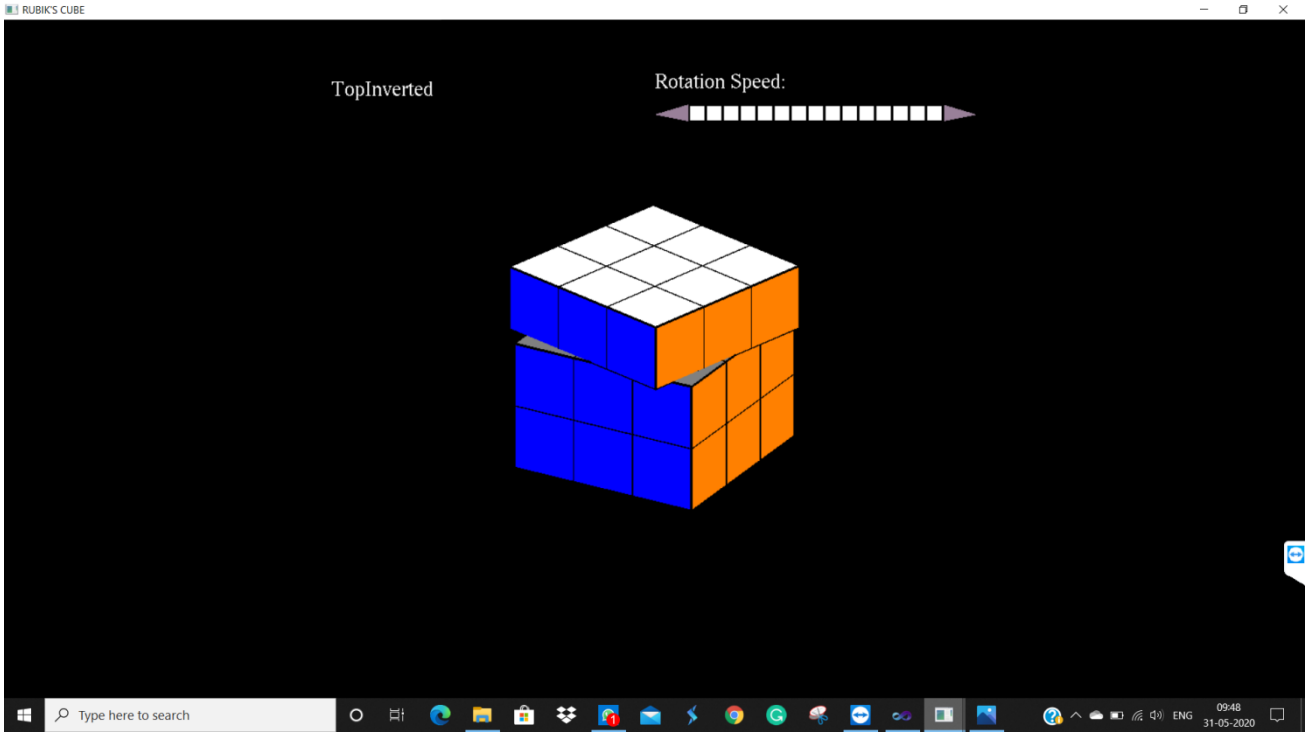

Figure 6.7: Top Inverted Rotation of the Cube

6.8 Bottom Rotation

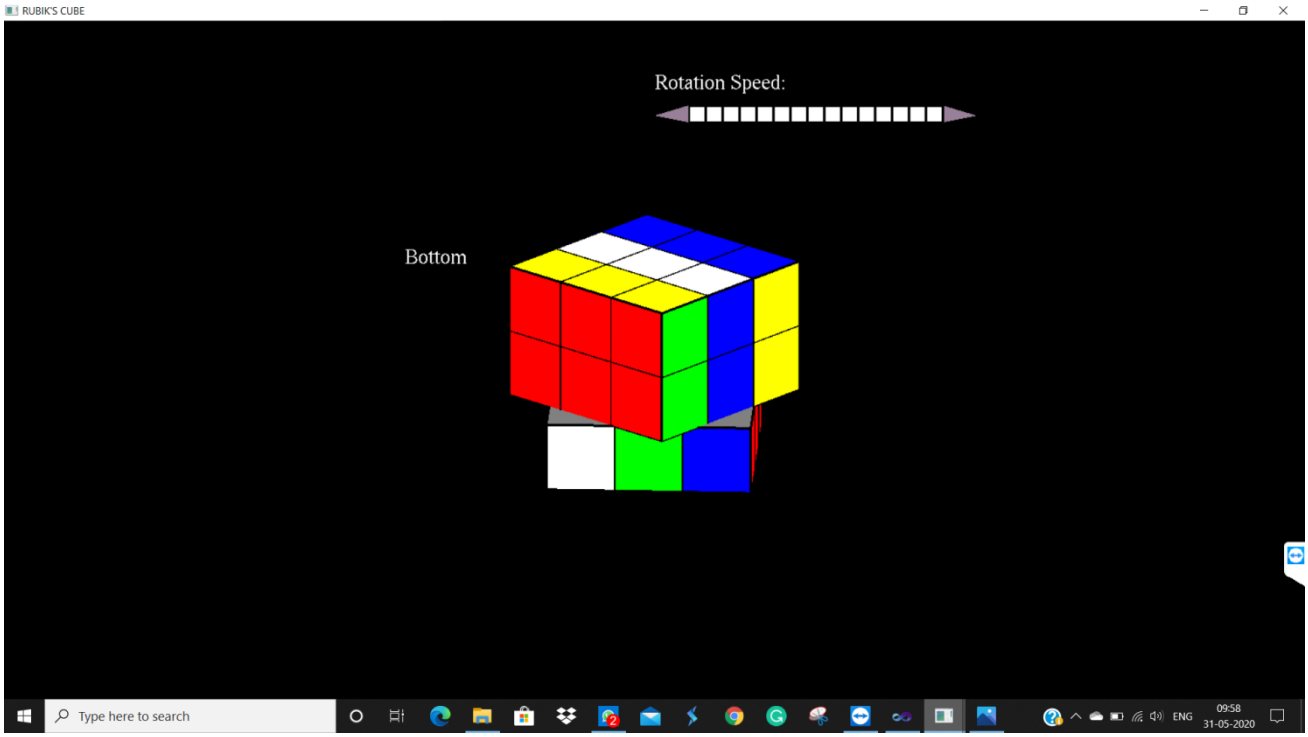

Figure 6.8: Bottom Rotation of the Cube

6.9 Bottom Inverted Rotation

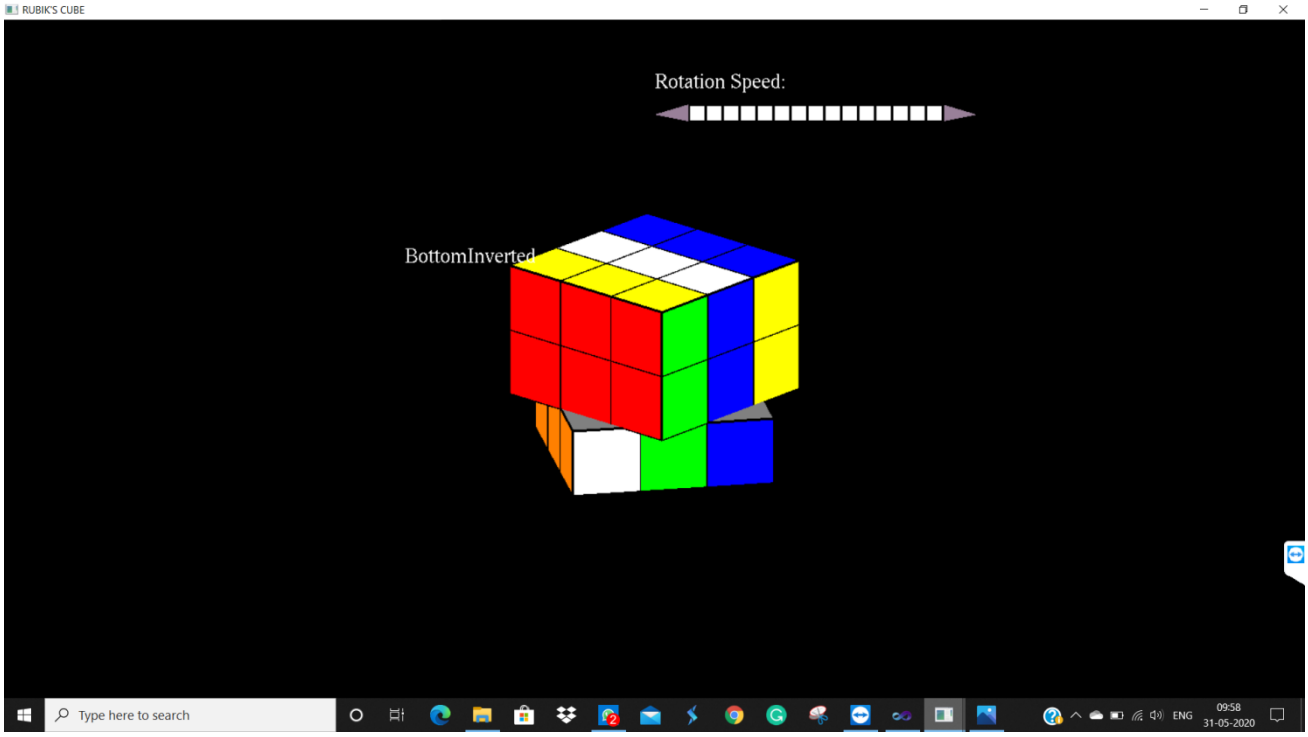

Figure 6.9: Bottom Inverted Rotation of the Cube

6.10 Left Rotation

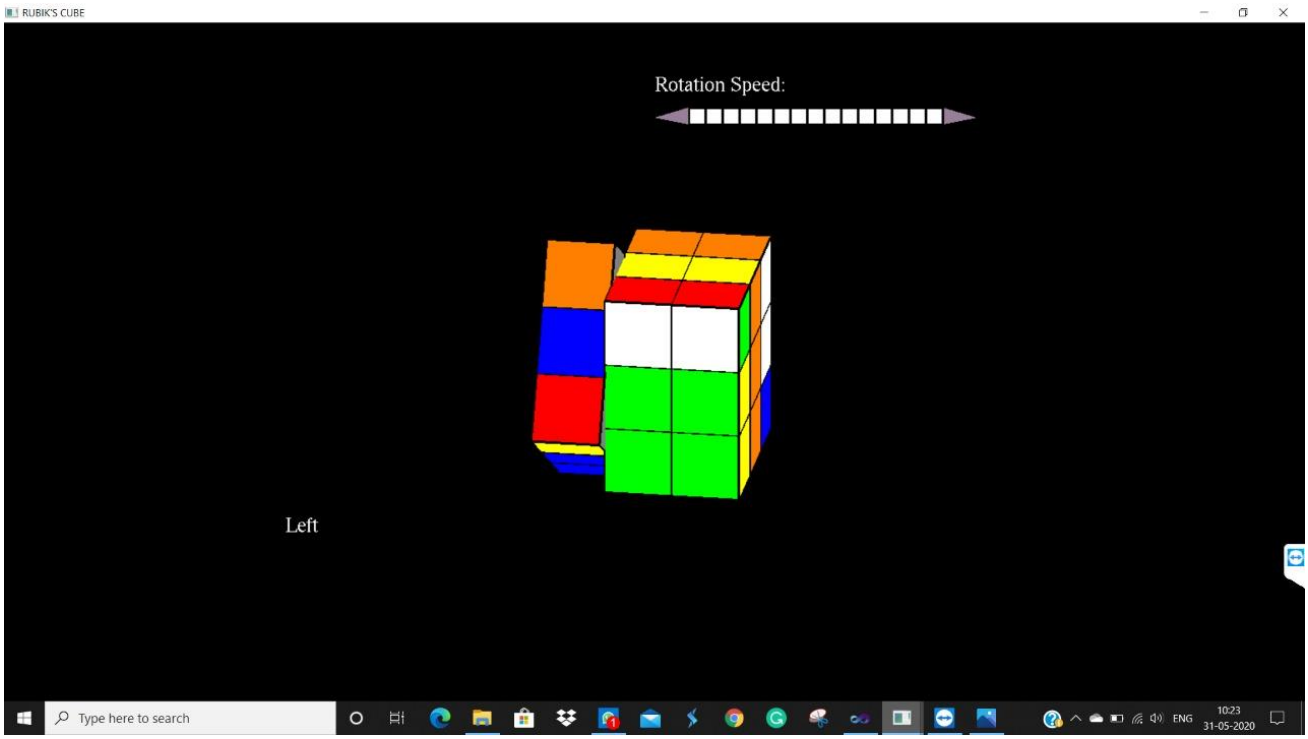

Figure 6.10: Left Rotation of the Cube

6.11 Left Inverted Rotation

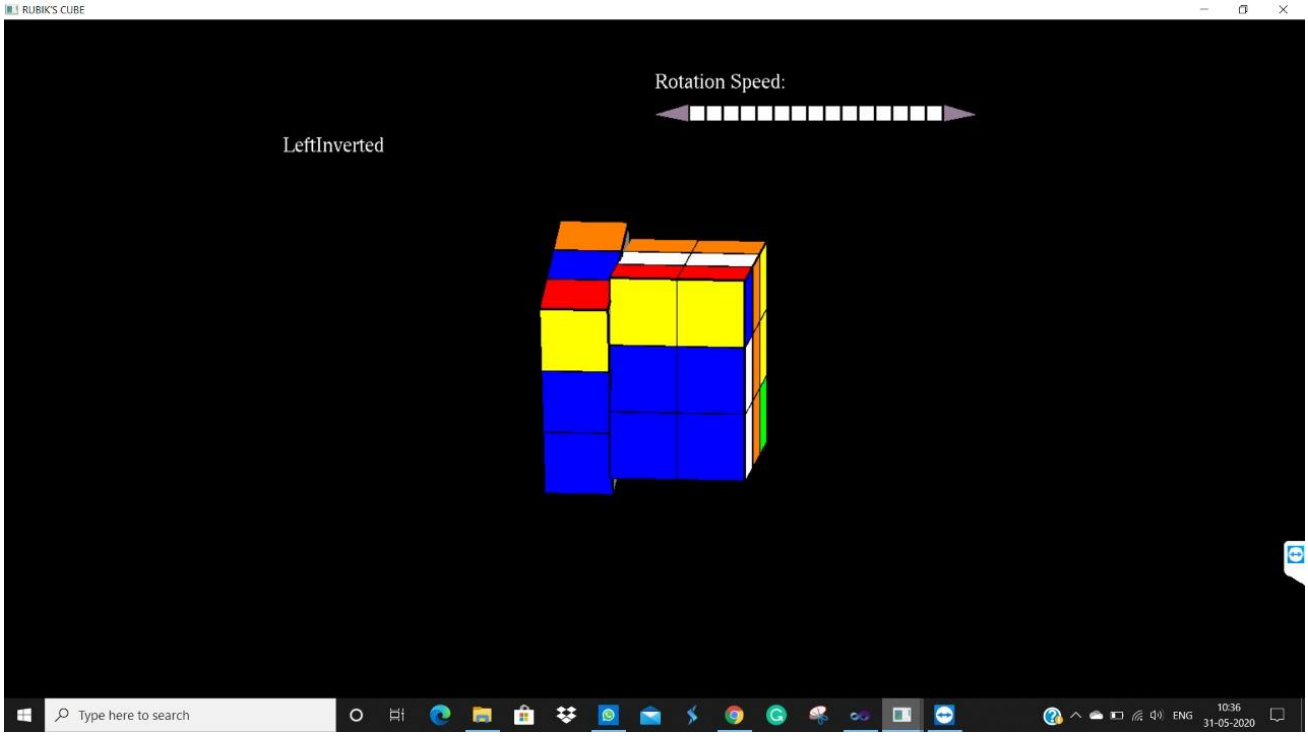

Figure 6.11: Left Inverted Rotation of the Cube

6.12 Right Rotation

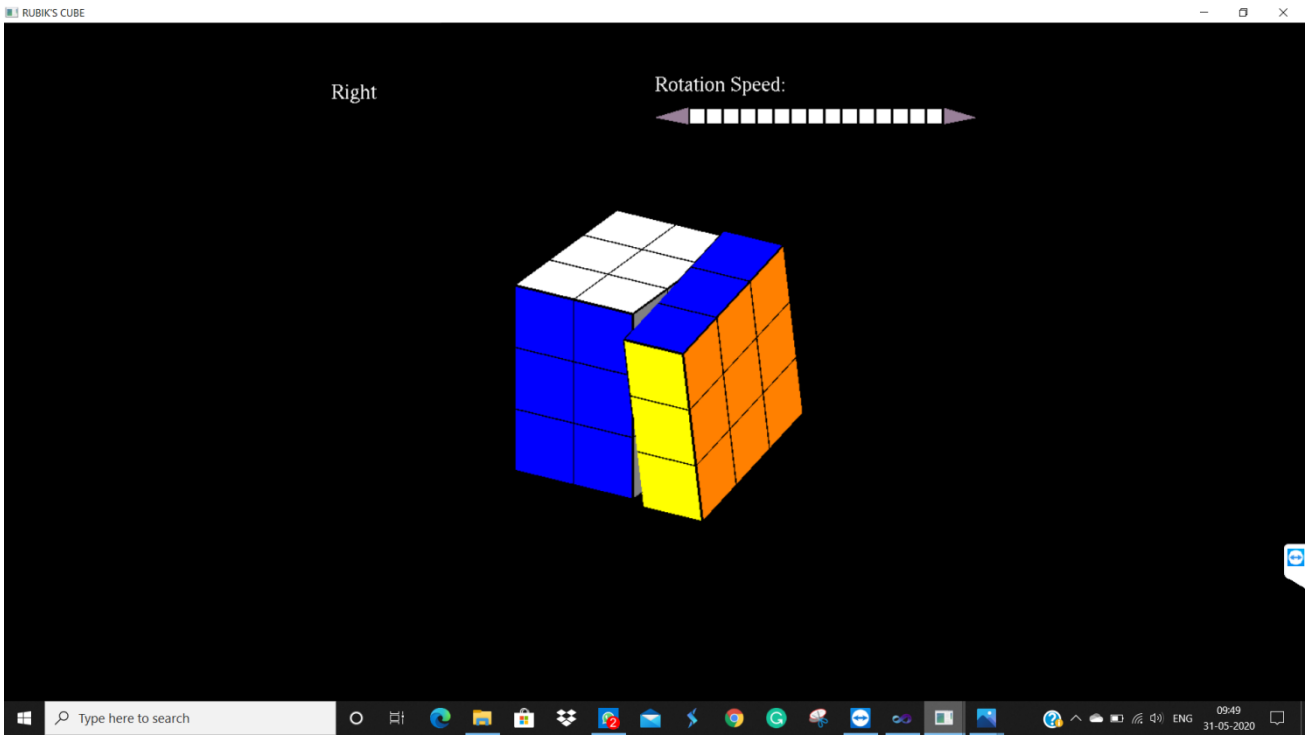

Figure 6.12: Right Rotation of the Cube

6.13 Right Inverted Rotation

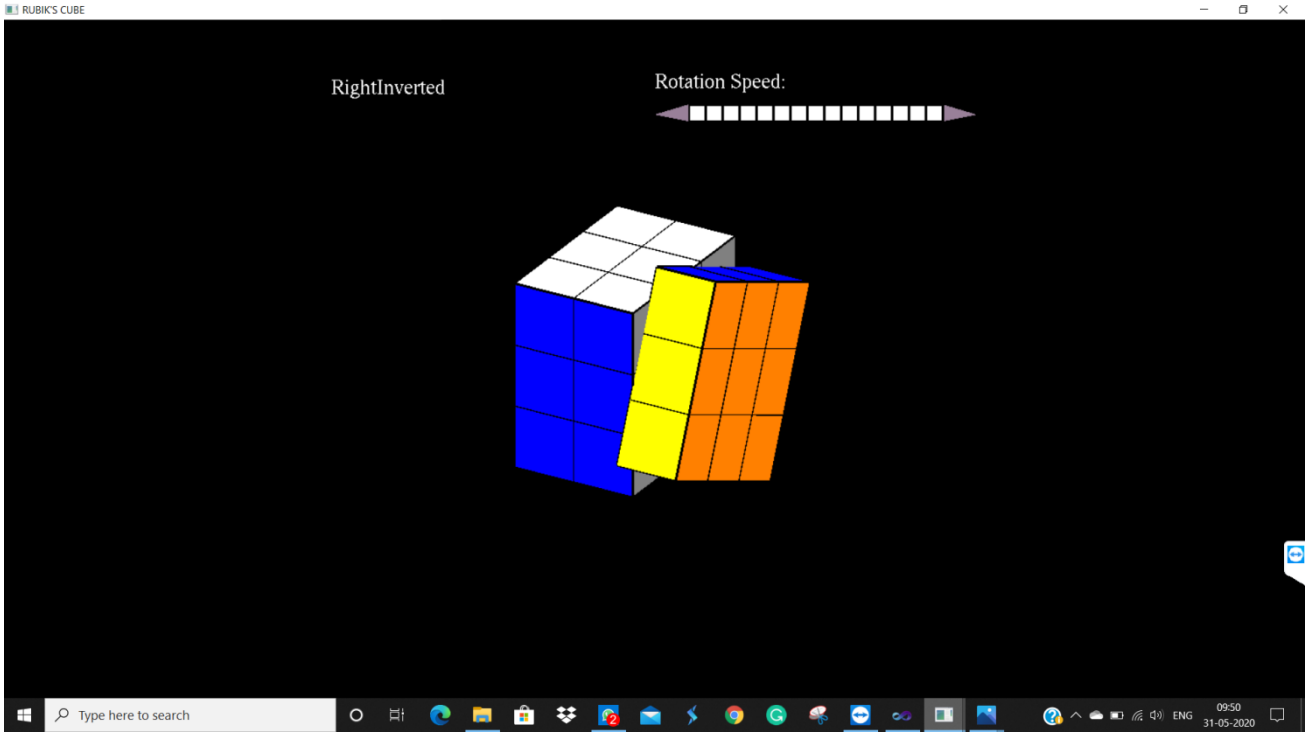

Figure 6.13: Right Inverted Rotation of the Cube

6.14 Solved Cube

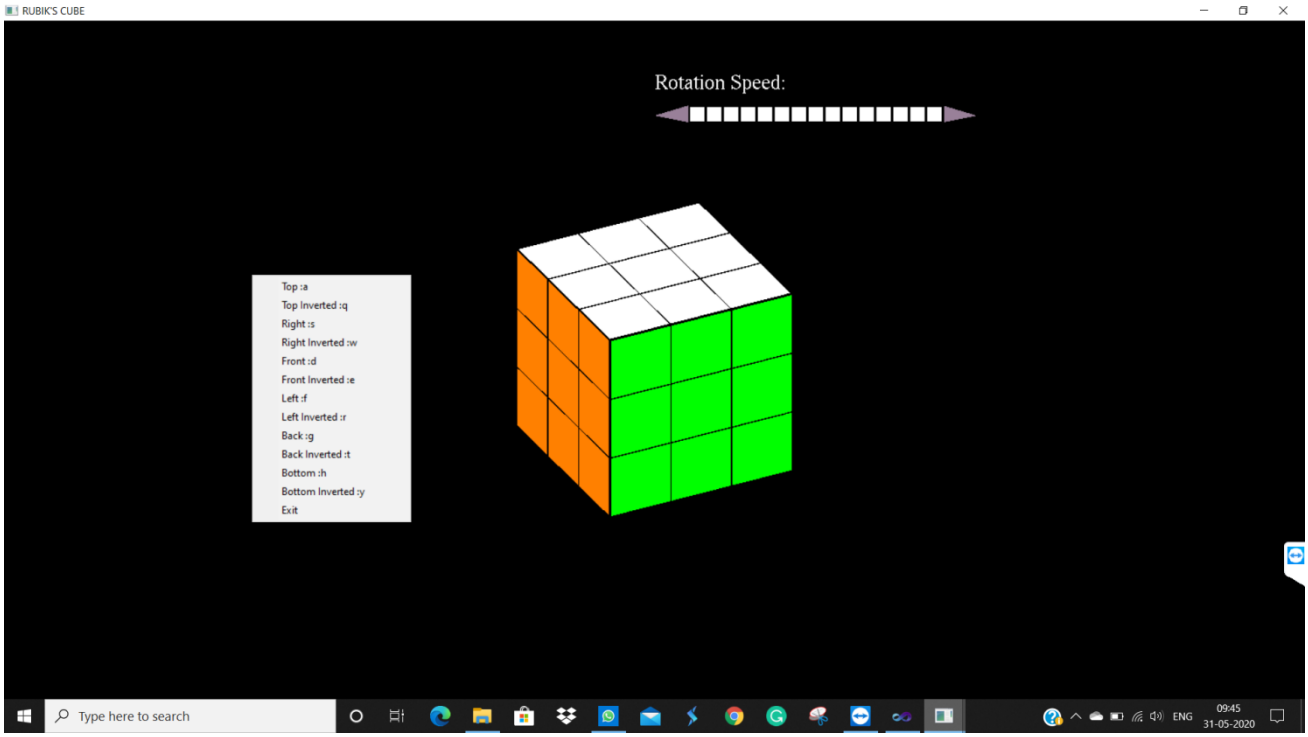

Figure 6.14: Solved Cube

CHAPTER 7

CONCLUSION

The development of computer graphics has made computers easier to interact with and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized the animation and video game industry.

We started with modest aim with no prior experience in any programming projects as this, but ended up in learning many things, fine tuning the programming skills and getting into the real world of software development with an exposure to corporate environment. During the development of any software of significant utility, we are faced with the trade-off between speed of execution and amount of memory consumed. This is simple interactive application. It is extremely user friendly and has the features, which makes simple graphics project. It has an open source code and no security features has been included. The user is free to alter the code for feature enhancement. Checking and verification of all possible types of the functions are taken care. Care was taken to avoid bugs. Bugs may be reported to creator as the need may be .So, we conclude on note that we are looking forward to develop more such projects with an appetite to learn more in computer graphics.

Future Enhancement

Further, we would like to enhance the appearance in the following ways:

- 3D View
- Enhanced appearance
- Background animation
- Colours and Shading
- 3D functions
- Multi-player interface

BIBILOGRAPHY

Reference Books

- Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd / 4th Edition, Pearson Education,2011
- Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL,5th edition. Pearson Education, 2008
- James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with OpenGL: pearson education

Websites

- www.opengl.org
- www.google.co.in(OpenGLprojects)
- www.wikipedia.com