VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Belagavi – 590 018

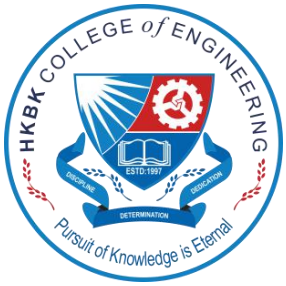
A
Mini Project Report
On

“Rubik’s Cube ”

Submitted in partial fulfillment of Bachelor of Engineering Degree
In
COMPUTER SCIENCE AND ENGINEERING
VI Semester, **18CSL67 – Computer Graphics & Visualization Laboratory with Mini Project**

Submitted by:
Aryan Bhardwaz 1HK18CS028
Under the guidance of
Prof. Mary Nathisiya
Assistant Professor, Department of Computer Science & Engineering

AUGUST 2021



Department of Computer Science and Engineering
HKBK COLLEGE *of* ENGINEERING
(Approved by AICTE & Affiliated to VTU)
Nagawara, Arabic College Post, Bangalore-45, Karnataka
Email: info@hkbk.edu.in URL: www.hkbk.edu.in

HKBK COLLEGE *of* ENGINEERING
Nagawara, Bangalore–560 045
Approved by AICTE & Affiliated to VTU

Department of Computer Science and Engineering

Certificate

Certified that the Mini Project Work entitled **"Rubik's Cube "**, carried out by **Aryan Bhardwaz (HK18CS028)** is bonafide students of **HKBK COLLEGE *of* ENGINEERING**, in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the **Visvesvaraya Technological University**, Belgaum, during the year **2020–21**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of **18CSL67–Computer Graphics & Visualization Laboratory with Mini Project** prescribed for the said Degree.

Prof. Mary Nathisiya	Dr. Loganathan R	Dr. M S Bhagyashekar
Guide	Professor & HOD	Principal

External Viva

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

ACKNOWLEDGEMENT

We would like to express our regards and acknowledgement to all who helped us in completing this project successfully.

First of all we would take this opportunity to express our heartfelt gratitude to the personalities, **Mr. C M Ibrahim**, Chairman, HKBKGI and **Mr. C M Faiz Mohammed**, Director, HKBKGI for providing facilities throughout the course.

We express our sincere gratitude to **Dr. M S Bhagyashekar**, Principal, HKBKCE for his support towards the attainment of knowledge.

We consider it as a great privilege to convey our sincere regards to **Dr. Loganathan. R.**, Professor and HOD, Department of CSE, HKBKCE, for his constant encouragement throughout the course of the project.

We would specially like to thank our guide **Prof. Mary Nathisiya**, Assistant Professor, Department of CSE, HKBKCE for her vigilant supervision and her constant encouragement. She spent her precious time in reviewing the project work and provided many insightful comments and constructive criticism.

Finally, we thank Almighty, all the staff members of CSE Department, our family members and friends for their constant support and encouragement in carrying out the project work.

Aryan Bhardwaz [1HK18CS028]

ABSTRACT

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 paper provides a basic understanding of the Rubik’s Cube and shows its mechanical art from the aspects of origin and development, characteristics, research status and especially its mechanical engineering design, as well as making a vision for the application in mechanism. First, the invention and origin of Rubik’s Cube are presented, and then the special characteristics of the cube itself are analyzed.

After that, the present researches of Rubik’s Cube are reviewed in various disciplines at home and abroad, including the researches of Rubik’s Cube scientific metaphors, reduction algorithms, characteristic applications, and mechanism issues. Finally, the applications and prospects of Rubik’s Cube in the field of mechanism are discussed.

TABLE OF CONTENTS

LIST OF FIGURES:

CHAPTER 1:
INTRODUCTION

INTRODUCTION

1.1 Introduction to Computer Graphics

Computer graphics started with the display of data on hardcopy plotters and Cathode Ray Tube (CRT) screens soon after the Introduction of computers themselves. 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, natural phenomenon and so on. 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 a keyboard, mouse, or touch- sensitive panel on the screen. Because of the close relationship between the input devices and the display, the handling of such devices is included in the study of computer graphics.

1.2 Uses of Computer Graphics :

Computer graphics is used in many different areas of industry, business, government, education, entertainment etc.

User Interfaces

Word-processing, spreadsheet and desktop-publishing programs are typical applications of such

user-interface techniques.

Interactive Plotting in Business, Science and Technology The common use of graphics is to create 2D and 3D graphs of mathematical, physical and economic functions, histograms, and bar and pie charts. Computer Aided Drafting and Design (CAD) In CAD, interactive graphics is used to design components and systems of mechanical, electrical

and electronic devices including structures such as buildings, automobile bodies, aero planes,ship hulls etc.Simulation and Animation for Scientific Visualization and

Entertainment Computer-produced animated movies are becoming increasing popular for scientific and engineering visualization. Cartoon characters will increasingly be modeled in the computer as 3D shape descriptions whose movements are controlled by computer commands.

2D Graphics

These editors are used to draw 2D pictures (line, rectangle, circle and ellipse) alter those with operations like cut, copy and paste. These may also support features like translation, rotation etc.

3D Graphics

These editors are used to draw 3D pictures (line, rectangle, circle and ellipse).These may also support features like translation, rotation etc.

About the project:

This project implements the movement of light source on the surfaces of the different objects.

1.3 ADVANTAGES

- Scientific visualization
- Information visualization
- Computer vision
- Image processing
- Computational geometry
- Computational topology
- Applied mathematics1.4 Header Files

Most of our applications will be designed to access OpenGL directly through functions in

three libraries. Functions in the main GL library have names that begin with the letters gl and are stored in a library usually referred 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. To interface with the window system and to get input from external devices into our programs, we need at least one library. For the X window system, this library is called GLX, for windows, it is wall etc.

GLUT will use GLX and X libraries.

#include<GL/glut.h>

OR

#include<GLUT/glut.h>

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. GLUT is designed for constructing small to medium sized OpenGL programs. While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications, GLUT is not a full-featured toolkit so large applications requiring sophisticated user interfaces are better off using native window system toolkits. GLUT is simple, easy, and small. The GLUT library has C, C++ (same as C), FORTRAN, and ADA programming bindings. The GLUT source code distribution is portable to nearly all OpenGL implementations and platforms. The current version is 3.7. Additional releases of the library are not anticipated. GLUT is not open source.

1.2 STATEMENT OF PROBLEM

Rubik’s Cube is a 3D combination puzzle invented in 1974 by Hungarian sculptor and professor of architecture Ernő Rubik [1] and was originally called the Magic Cube [2, 3]. This invention caused the widespread interest in the world owing to its unique characteristics, which exerted a

profound impact on mankind. Rubik’s Cube is listed as one of the 100 most influential inventions during the 20th century [4]. Additionally, it is widely considered to be the world’s best-selling toy [5]. It won a German Game of the Year special award [6] and won similar awards for best toy in the UK, France, and the US. [7]. Although the Rubik’s Cube reached its height of main- stream popularity in the 1980s, it is still widely known and used. It not only attracts Rubik’s Cube enthusiasts conducting research into Rubik’s Cube reduction algorithms [8–10] but also draws the attention of scientists and technical workers from various walks of life for its sophisticated design and ideas [11]. On the one hand, the Rubik’s Cube structure has several features such as rotation, permutation and combinations, and cycle and symmetry, which were treated as physical models or tools to study specific scientific issues or were studied by using scientific theory or methods in some areas. All in all, the principles of Rubik’s Cube are contained in numerous scientific systems that involve permutations and combinations, symmetries, and cyclicality. On the other hand,scholars began to explore the inner movement principles of the Rubik’s Cube structure. The application prospects of Rubik’s Cube have been discussed according to its rotation characteristics.

1.3 OBJECTIVE OF THE PROBLEM

The project improvised on this basic idea to convert a normal 2 Dimensional game into a graphical 3 Dimensional game.The original game is a 2 Dimensional game that involves swapping of any 2 tiles and arranging the tiles in the correct sequence. We modified this idea to include a blank position.

This now increases level of difficulty as only the tiles adjacent to the blank position can now be translated, unlike the Puzzle Slider game.

CHAPTER 2:
SYSTEM REQUIREMENTS SPECIFICATIONS

SYSTEM REQUIREMENTS 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 behaviour of the system.

Functional Requirements

Functional Requirements define the internal working of the software. The following conditions must be taken care of: The ability to perform correct operations when corresponding keys are pressed.

Non-functional Requirements

Non-functional requirements are requirements which specify criteria that can be used to judge the operation of the system, rather than specific behaviours. This should be contrasted with functional requirements that specify specific behaviour or functions. Typical non-functional requirements are reliability and scalability. Non-functional requirements are "constraints", "quality attributes" and "quality of service requirements".

Types of non-functional requirements

- √ **Volume:** Volume of a system (the total space occupied) varies depending on how the component assemblies are arranged and connected.
- √ **Reliability:** System reliability depends on component reliability but unexpected interactions can cause new types of failure and therefore affect the reliability of the system.
- √ **Security:** The security of the system (it's ability to resist attacks) is a complex property that cannot be easily measured. Attacks maybe devised that were not anticipated by the system designers and may use default built-in safeguards.
- √ **Repairability:** This property reflects how easy it is to fix a problem with the system once it has been discovered. It depends on being able to diagnose the problem, access the components that are faulty and modify or replace these components.
- √ **Usability:** This property reflects how easy it is to use the system. It depends on the

technical system components, its operators and its operating environment.

Software Requirements:

Operating System : Kali Linux
Front End : Terminal

Coding Language : C

Hardware Requirements

System : Intel® Core™ i3 – 6006U CPU @ 2.00GHz

Hard Disk : 30 GB or above

Monitor : 15 VGA color

RAM : 256 MB or above

Introduction to Environment

OpenGL is a software interface to graphics hardware. This interface consists of about 120 distinct commands, which you use to specify the objects and operations needed to produce interactive three - dimensional applications.

OpenGL is designed to work efficiently even if the computer that displays the graphics you create isn’t the computer that runs your graphics program. This might be the case if you work in a networked computer environment where many computers are connected to one another by wires capable of carrying digital data. In this situation, the computer on programs can work across a network even if the client and server are different kinds of computers. If an OpenGL program isn’t running which your program runs and issues OpenGL drawing commands is called the client, and the computer that receives those commands and performs the drawing is called the server. The format for

transmitting OpenGL commands (called the protocol) from the client to the server is always the same, so OpenGL across a network, then there’s only one computer, and it is both the client and the server.

CHAPTER 3: DESIGN

CHAPTER 3: DESIGN

Requirement analysis encompasses all the tasks that go into the instigation, scoping and definition of a new or altered system. Requirement analysis is an important part of the design process. Here we identify the needs or requirements. Once the requirements have been identified the solution for the requirements can be designed.

We design a project with specific goals, tasks and outcomes. The more specific and the more closely it is aligned with traditional instructional objectives, the better

3.1 High Level Design

High Level Design (HLD) is the overall system design - covering the system architecture and database design. It describes the relation between various modules and functions of the system. Data flow, flow charts and data structures are covered under HLD.

The different modules used in this project are as follows:

- **glutInit(int *argcp, char **argv);**

  The argument 'argcp' is a pointer to the program’s unmodified 'argv' variable from main. Upon return, the value pointed to by 'argcp' will be updated, because glutInit extracts any command line options intended for the GLUT library 'argv'. Like argcp, the data for argv will be updated because glutInit extracts any command line options understood by the GLUT library.

- **glutInitDisplayMode(unsigned int mode);**

  It sets the initial display mode. The argument 'mode' is the display mode, normally the bitwise OR-ing of GLUT display mode bit masks.

  See values below:

  GLUT_SINGLE: Bit mask to select a single buffered window. This is the default if neither GLUT_DOUBLE or GLUT_SINGLE are specified.

  GLUT_RGB: Bit mask to select a RGBA mode window.

- **glutMainLoop(void);**

  The module glutMainLoop enters the GLUT event processing loop. This routine should be called atmost once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

- **glutCreateWindow(char *name);**

  This creates a top-level window. The argument 'name' is provided to the system as the window's name with the intent that the window system will label the window with the name.

- **glutDisplayFunc(void (*func)(void));**

  Sets the display callback for the current window. The 'func' is the new display callback function.

- **glLoadIdentity(void);**

  It replaces the current matrix with the identity matrix. It is semantically equivalent to calling glLoadMatrix with the 4X4 identity matrix.

- **glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha);**

  Specifies the red, green, blue, and alpha values used when the color buffers are cleared. The intial values are all 0. Values specified by glClearColor are clamped to the range [0-1].

- **glutReshapeFunc(void (*func)(int width, int height));**

  This sets the reshape callback for the current window. The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created or whenever an overlay for the window is established.

- **glViewport(GLint x, GLint y, GLsizei width, GLsizei height);**

  The parameters ‘x’, ‘y’ specify the lower left corner of the viewport rectangle, in pixels. The parameters ‘width’, ‘height’ specify the width and height of the viewport.

glViewport specifies the affine transformation of x and y from normalized device coordinates to window coordinates.

- **glFlush(void);**

  The glFlush function forces execution of OpenGL functions to be processed.

- **glClear(void);**

  The glClear function is used to set initial values for other buffers that are available in OpenGL.

- **glutInitWindowSize(int width, int height);** and **glutInitWindowPosition(int x, int y);**

  Windows created by glutCreateWindow will be requested to be created with the current initial window position and size.

- **glutKeyboardFunc(void (*func)(unsigned char key,int x, int y));**

  glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character.

3.2 Flowchart For Rubik’s cube:

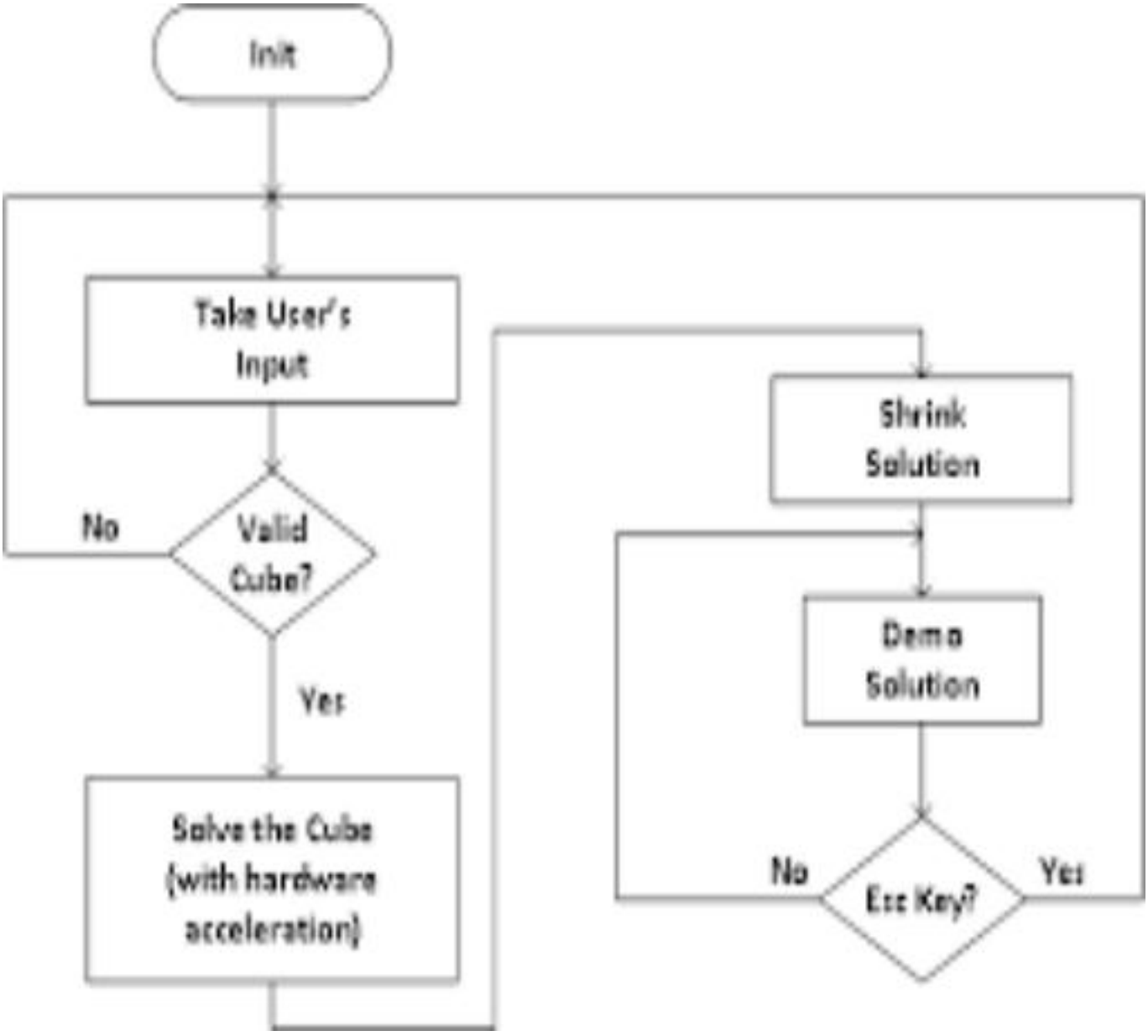


Fig 3.2: Flowchart for Rubik’s cube

CHAPTER 4:
IMPLEMENTATION

IMPLEMENTATION:

Implementation is an act or instance of implementing something that is, the process of making something active or effective. Implementation must follow any preliminary thinking in order for something to actually happen. In the context of information technology, implementation encompasses the process involved in getting new software or hardware to operate properly in its environment. This includes installation, configuration, running, testing and making necessary changes.

Libraries Used:

OpenGL function names begin with the letters gl and are stored in a library usually referred to as GL. The first is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects, such as spheres, and other tasks that users prefer not to write repeatedly. We use a readily available library called the OpenGL Utility Toolkit (GLUT) that provides the minimum functionality that should be expected in any modern windowing system.

Explanation of Functions:

1.file_gen is used to check whether high score file already exists. If it does not, it will create. If it does, it will read data from existing file.
2.render_header to display the player's name along with a running timer. The timer gets its values from the clock class, which maintains the time elapsed since the game started in hour, minutes and seconds.
3.hiscore_check to determine the position of the player in the highscores list. If he completes the game in a time lesser than the top 10 players, it implements an insertion sort to insert the player's name and time into the existing high score list.
4.write_file is used to write a text file of the current player class. The player class contains details of the player name and the time he took to complete the game. All these details, along with the pre-existing records of the highscores are arranged in ascending order of time and are written into a text file.
5.sub is the display function for the sub window created to display the highscores. This window is activated when the user chooses the option "highscores" from the menu obtained on right click of the mouse.

6. Mouse events: When mouse event occurs, the ASCII code for the corresponding coordinates that generate the event and the location of mouse are returned.

Mouse callback function is

```
glutMouseFunc (mouse);
Void mouse (int btn, int state, int x, int y)
{
if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
begin = x;
}
```

7. Menu Entry:GLUT provides one additional feature, pop_up menus, which we can use with the mouse to create sophisticated interactive application

```
glutCreateMenu ();
glutAddMenuEntry ();
glutAttachMenu (GLUT_RIGHT_BUTTON);
```

User Defined Functions:

User defined functions is a programmed routine that has its parameters set by the user of the system. User defined functions often are seen as programming shortcuts as they define functions that perform specific tasks within a larger system, such as a database or spreadsheet program.

Step 1: STEP 1: Define vertices for 27 cubes which are used to compose one whole cube known as "Rubiks cube".

```
#include<string.h>
#include<GL/glut.h>
#include<GL/gl.h>
#include<GL/glu.h>
#include<stdio.h>

void *font = GLUT_BITMAP_HELVETICA_18;
```

```
char defaultMessage[] = "Rotation Speed:";
char *message = defaultMessage;

void output(int x, int y, char *string)
{
	int len, i;
	glRasterPos2f(x, y);
	len = (int) strlen(string);
	for (i = 0; i < len; i++)
	{
		glutBitmapCharacter(font, string[i]);
	}
}
```

Step 2: Define colors for each cube of the Rubiks cube todistinguish one cube from the other and as well as color forthe speed meter, which is used to control the speed ofrotation.

```
 GLfloat color[][3]={{1.0,1.0,1.0},   //white
                     {1.0,0.5,0.0},   //orange
                     {0.0,0.0,1.0},   //blue
                     {0.0,1.0,0.0},   //green
                     {1.0,1.0,0.0},   //yellow
                     {1.0,0.0,0.0}, //red
            {0.5,0.5,0.5}, //grey used to represent faces of cube without colour
            {.6,.5,.6}//speed meter colour
};
```

Step 3: Output( ) function: This function is used to display the message using the Commands glutBitmapCharacter( ) and glRasterpos*( ).

```
void display()
{
```

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
 speedmeter();
 glColor3fv(color[0]);
output(1,8,message);
glColor3f(0.5,0.9,0.3);
output(-17,9,"By:");
output(-17,8,"Aryan Bhardwaz");
output(-17,7,"1HK28CS028");
output(-17,6,"HKBKCE");
glColor3fv(color[0]);
output(1,-6,"Right click for controls...");
glPushMatrix();
glRotatef(25.0+p,1.0,0.0,0.0);
glRotatef(-30.0+q,0.0,1.0,0.0);
glRotatef(0.0+r,0.0,0.0,1.0);
}
```

Step 4: Draw a polygon via list of vertices with the line of 3 pixels wide.

```
void polygon(int a,int b,int c,int d,int e)
{
glColor3f(0,0,0);
glLineWidth(3.0);
glBegin(GL_LINE_LOOP);
glVertex3fv(vertices[b]);
glVertex3fv(vertices[c]);
glVertex3fv(vertices[d]);
glVertex3fv(vertices[e]);
glEnd();
glColor3fv(color[a]);
glBegin(GL_POLYGON);
glVertex3fv(vertices[b]);
glVertex3fv(vertices[c]);
```

```
        glVertex3fv(vertices[d]);
        glVertex3fv(vertices[e]);
        glEnd();
}
```

Step 5: Add entries to the menu and link the menu to the right

```
int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize(500, 500);
        glutCreateWindow("RUBIK'S CUBE");
        glutReshapeFunc(myreshape);
        glutIdleFunc(spincube);
        glutMouseFunc(mouse);
        glutMotionFunc(motion);
        glutCreateMenu(mymenu);

        glutAddMenuEntry("Orientation:",0);
        glutAddMenuEntry("Top                 :a",1);
        glutAddMenuEntry("Top Inverted      :q",2);
        glutAddMenuEntry("Right               :s",3);
        glutAddMenuEntry("Right Inverted   :w",4);
        glutAddMenuEntry("Front               :d",5);
        glutAddMenuEntry("Front Inverted   :e",6);
        glutAddMenuEntry("Left                :f",7);
        glutAddMenuEntry("Left Inverted    :r",8);
        glutAddMenuEntry("Back                :g",9);
```

```
        glutAddMenuEntry("Back Inverted     :t",10);
        glutAddMenuEntry("Bottom              :h",11);
        glutAddMenuEntry("Bottom Inverted :y",12);
        glutAddMenuEntry("Reverse move     :o",0);
        glutAddMenuEntry("",0);
        glutAddMenuEntry("Rotation Controls:",0);
        glutAddMenuEntry("X-Axis      :4 & 6",0);
        glutAddMenuEntry("Y-Axis      :2 & 8",0);
        glutAddMenuEntry("Z-Axis      :1 & 9",0);
        glutAddMenuEntry("Origin      :5",0);
        glutAddMenuEntry("",0);
        glutAddMenuEntry("Rotation Speed:",0);
        glutAddMenuEntry("Increase     :m",0);
        glutAddMenuEntry("Decrease    :n",0);
        glutAddMenuEntry("",0);
        glutAddMenuEntry("Exit",13);

        glutAttachMenu(GLUT_RIGHT_BUTTON);
        glutKeyboardFunc(keyboard);
        glutDisplayFunc(display);
        glEnable(GL_DEPTH_TEST);
        glutMainLoop();
        //return 0;
}
```

Step 6:Stop.

CHAPTER 5:
SNAPSHOTS

SNAPSHOTS

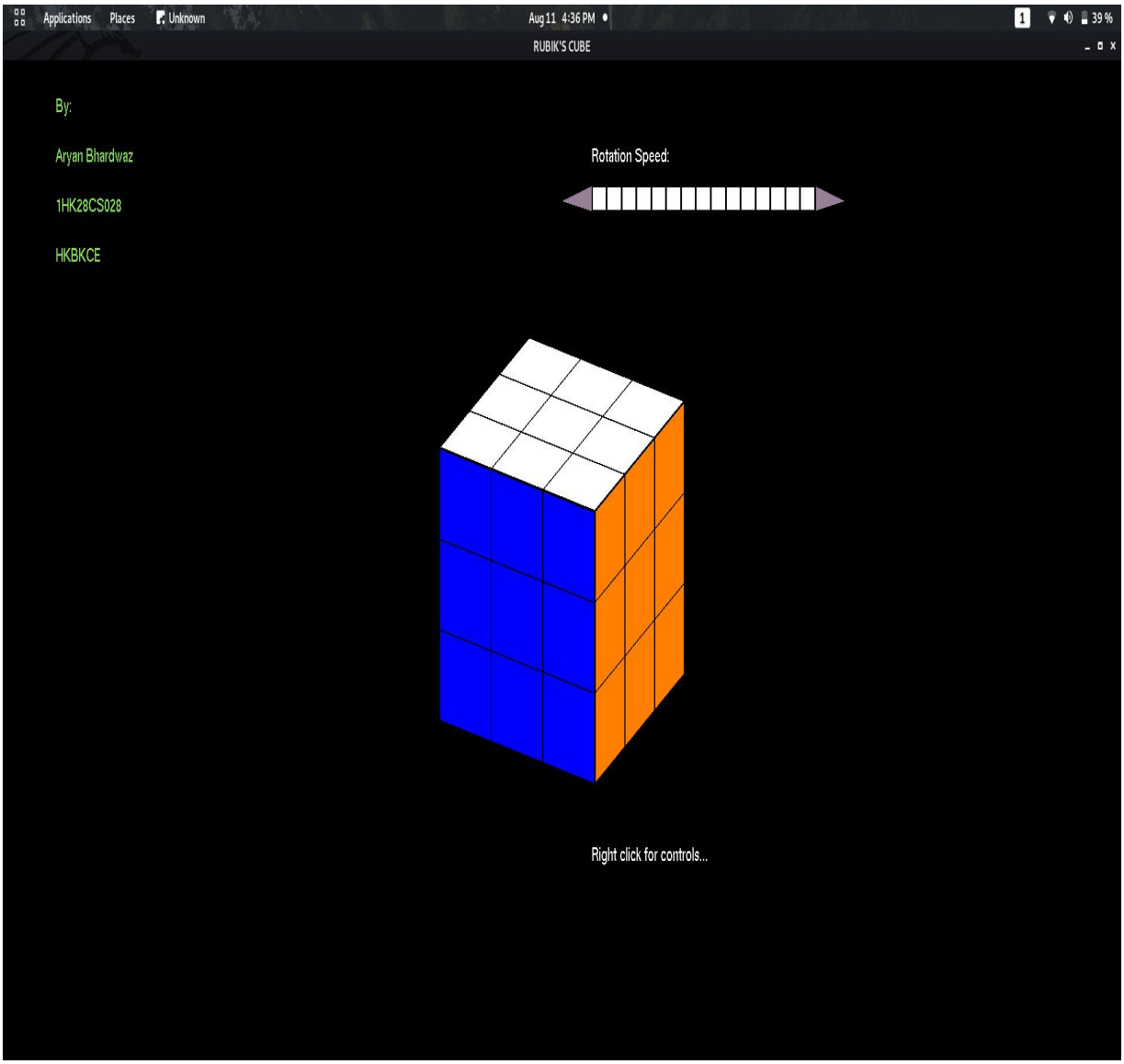


Fig 5.1: Home

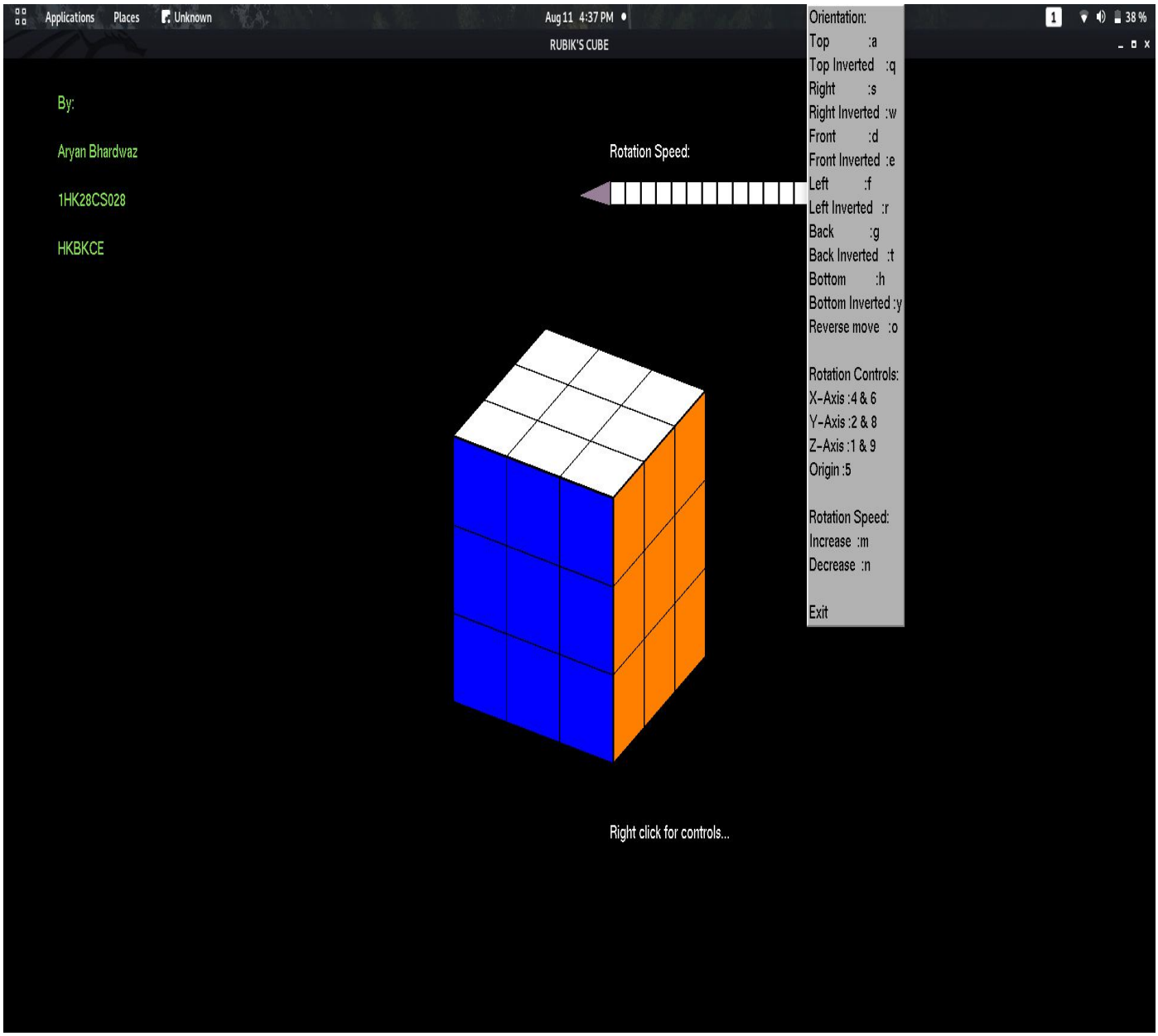


Fig 5.2: control key

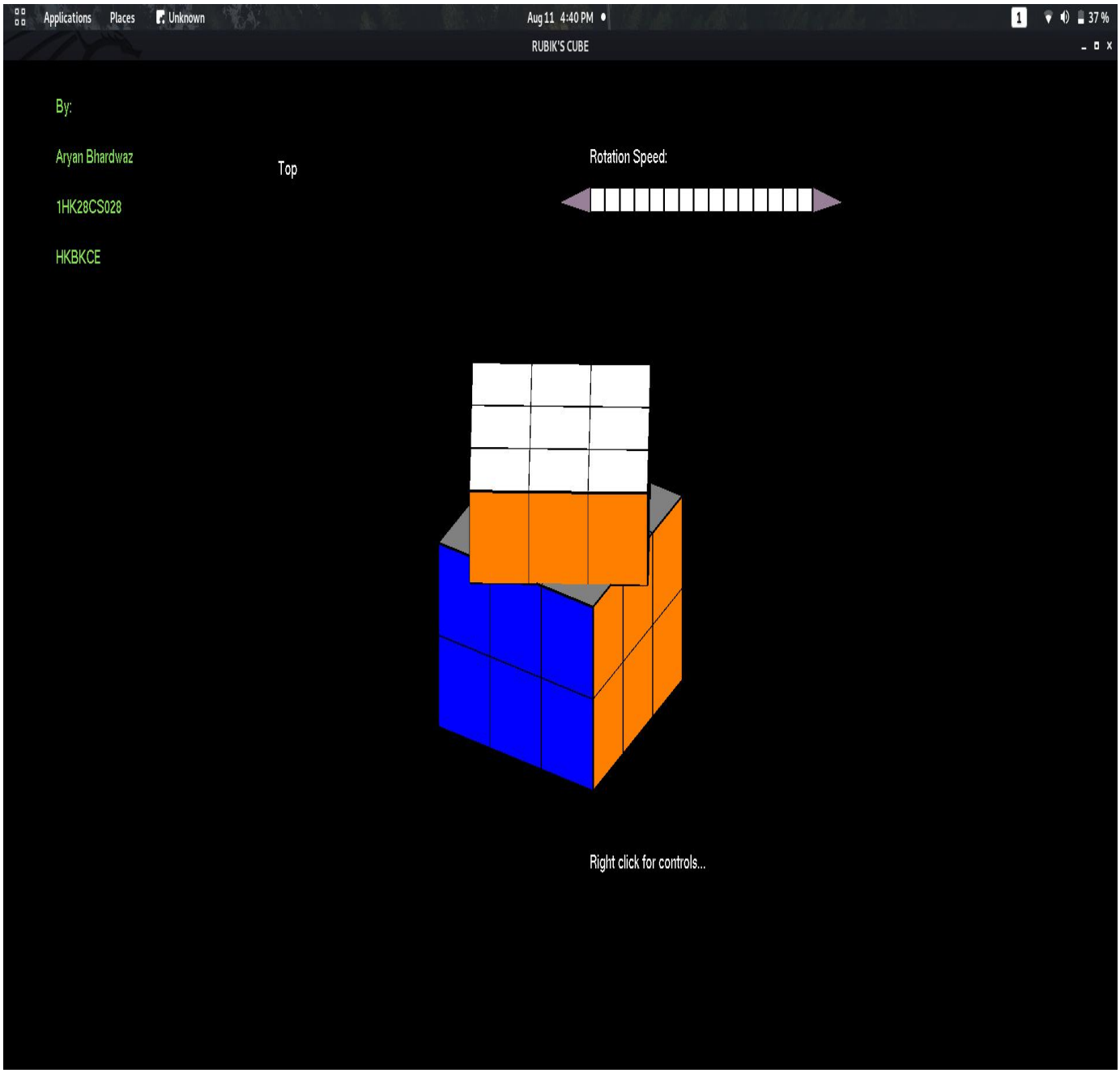


Fig 5.3: Top Rotations:

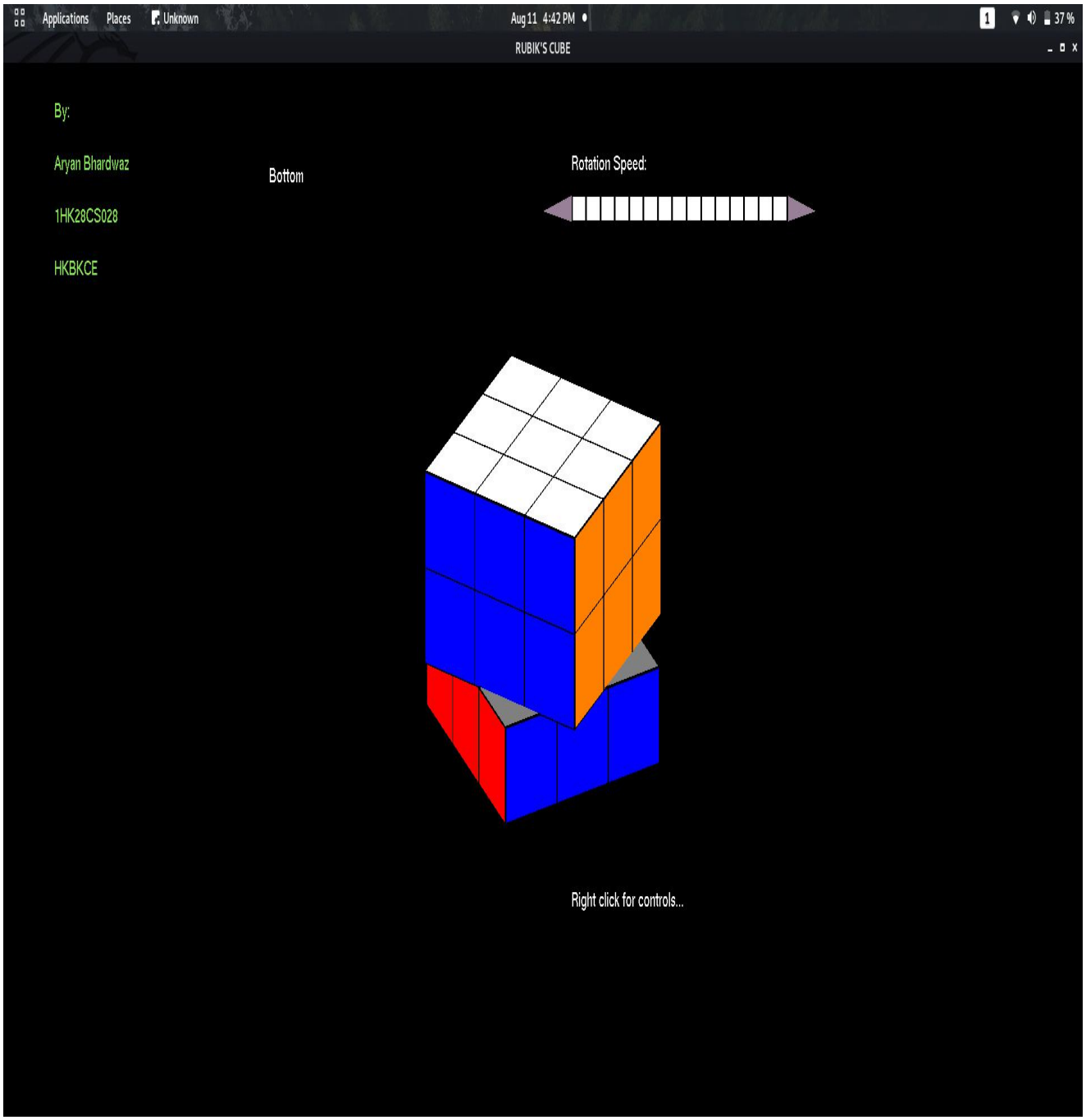


Fig 5.4: Bottom Rotations:

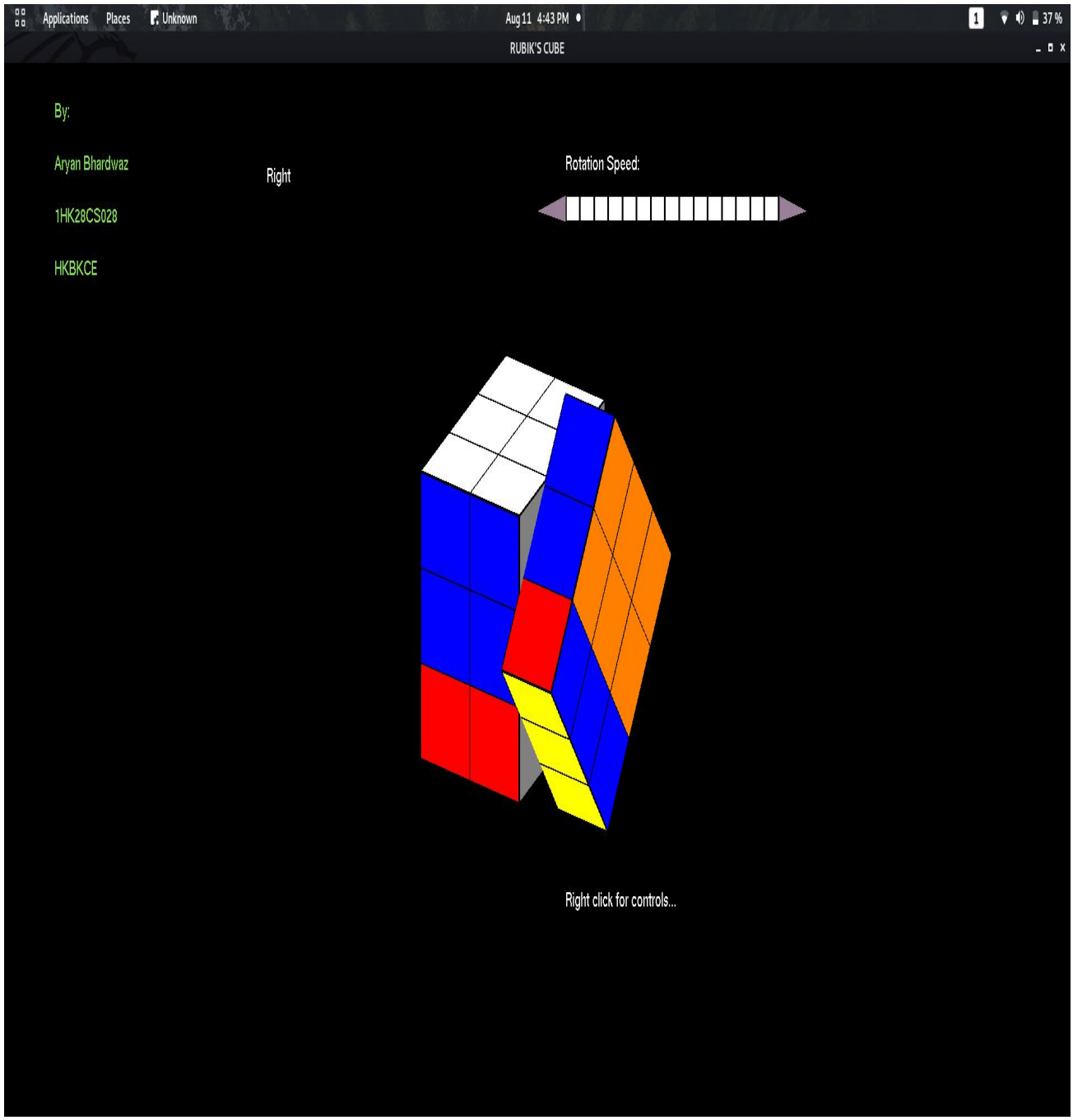


Fig 5.5: Right Rotations:

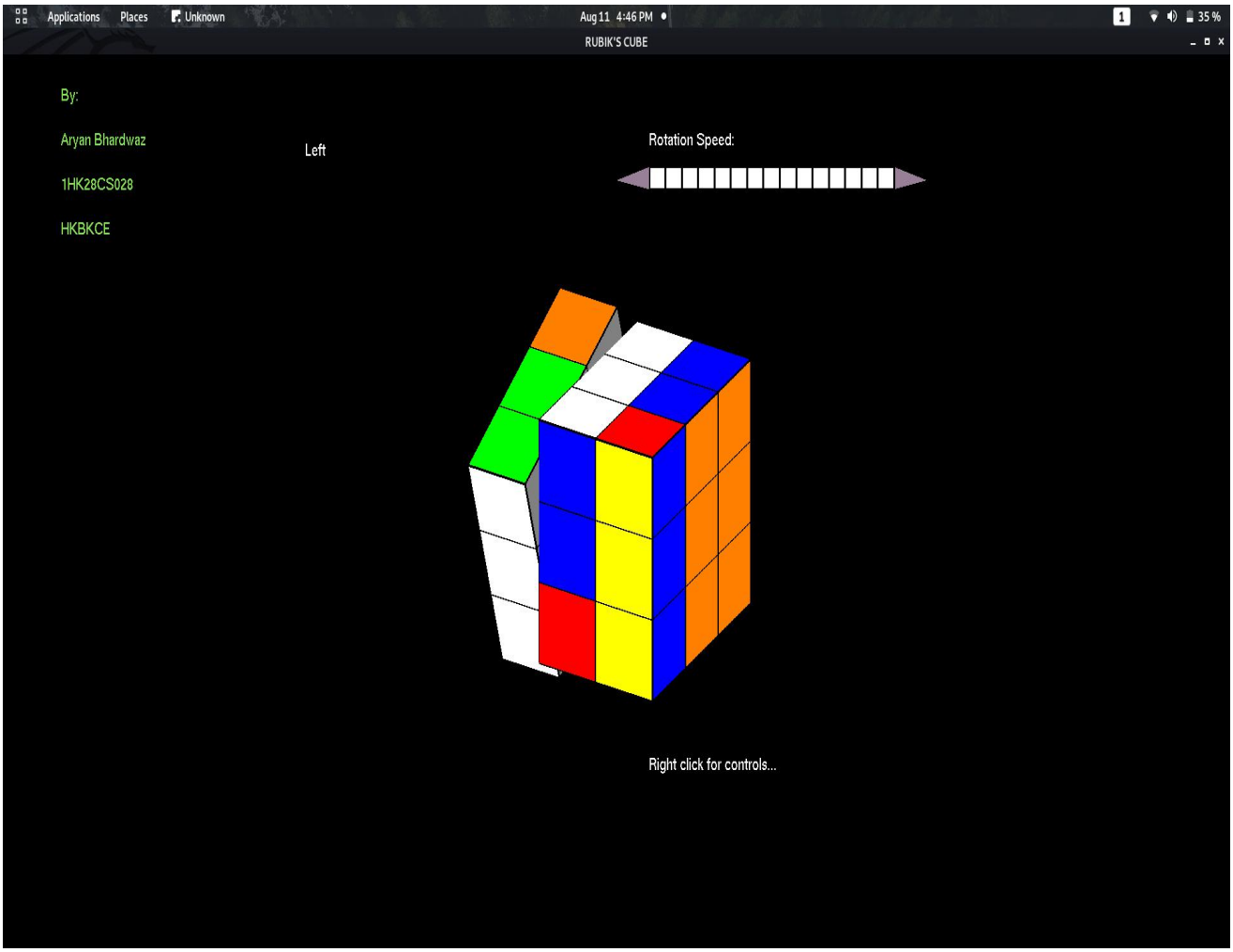


Fig 5.6: Left Rotations:

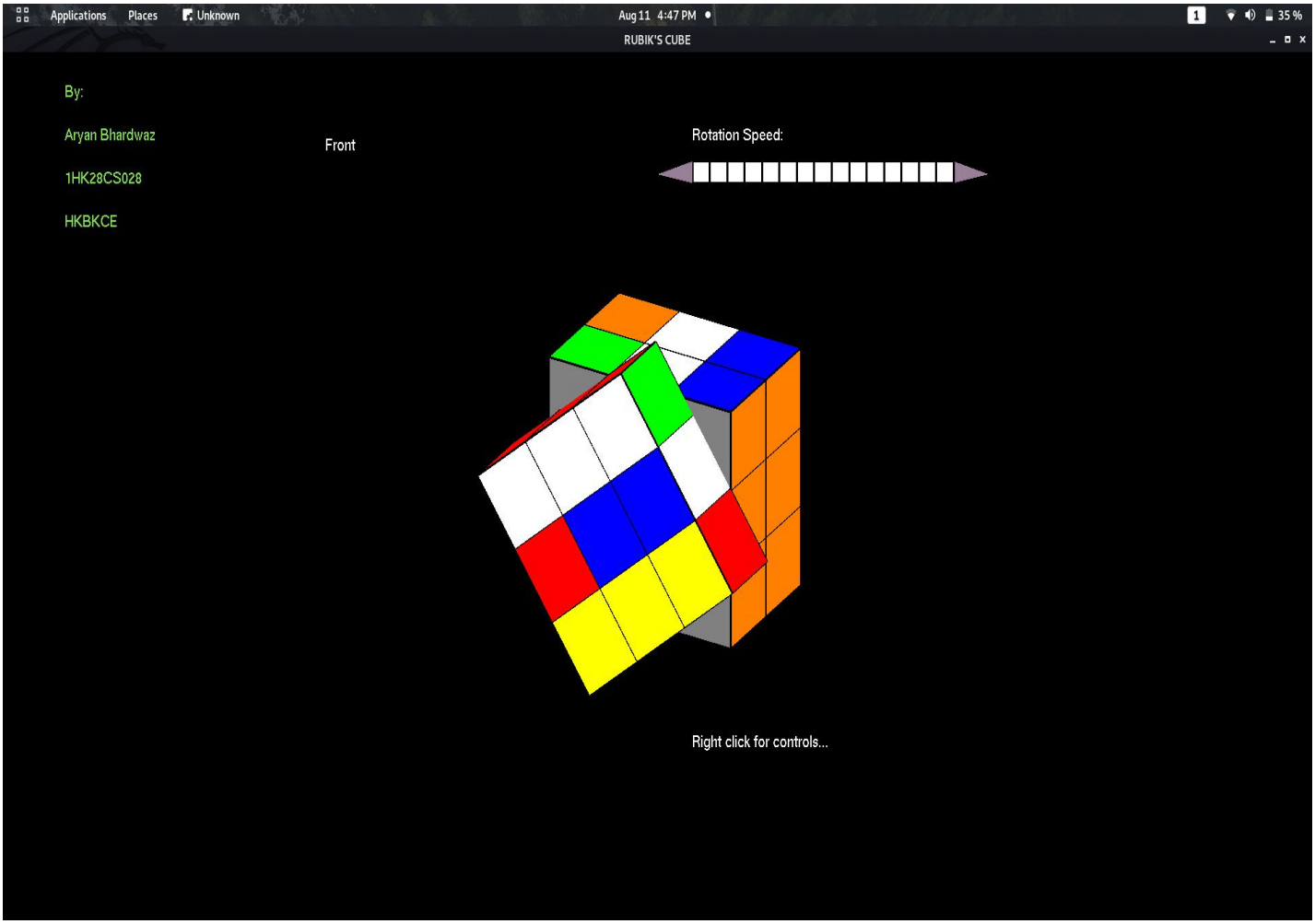


Fig 5.7: Front Rotations:

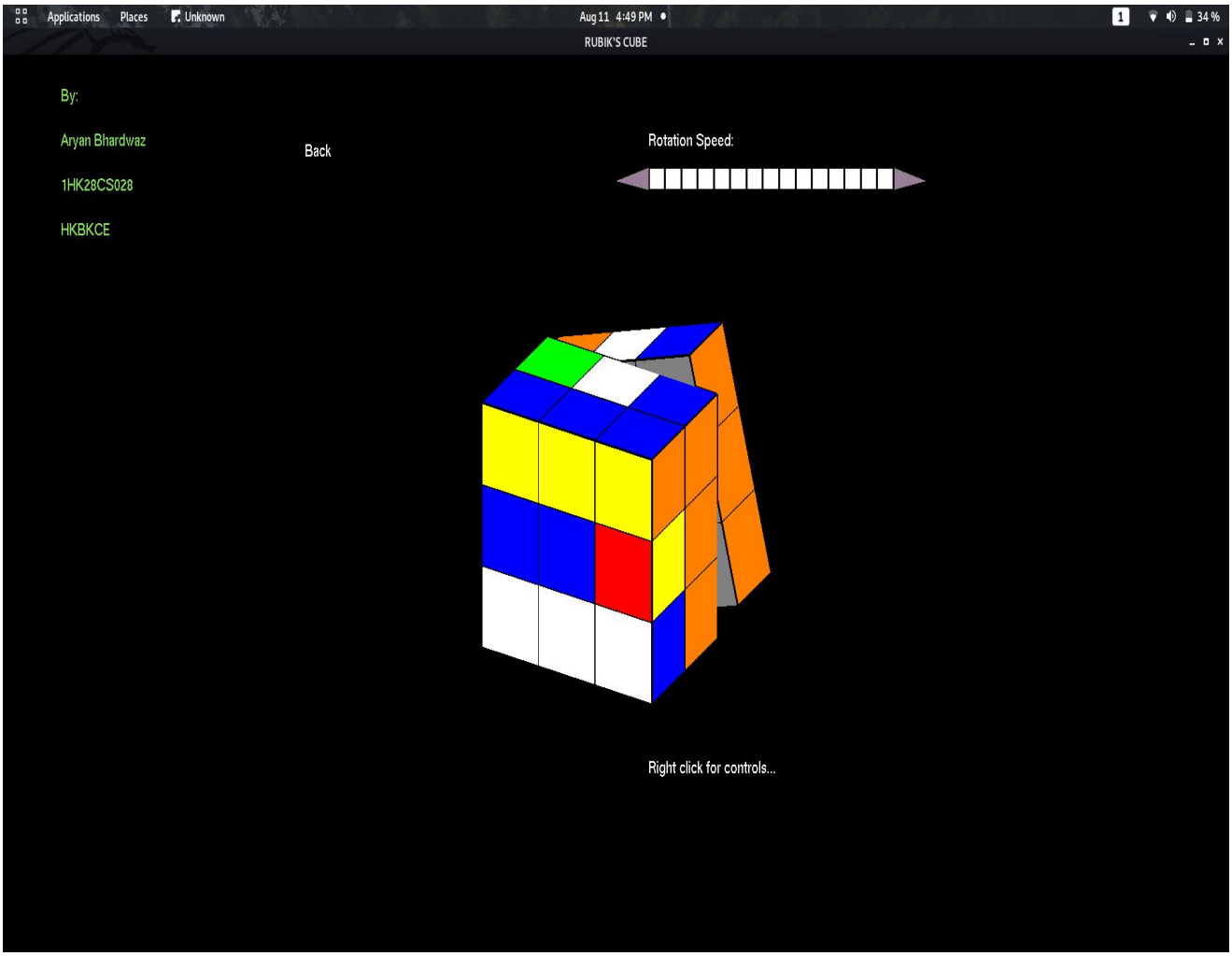


Fig 5.8: Back Rotations:

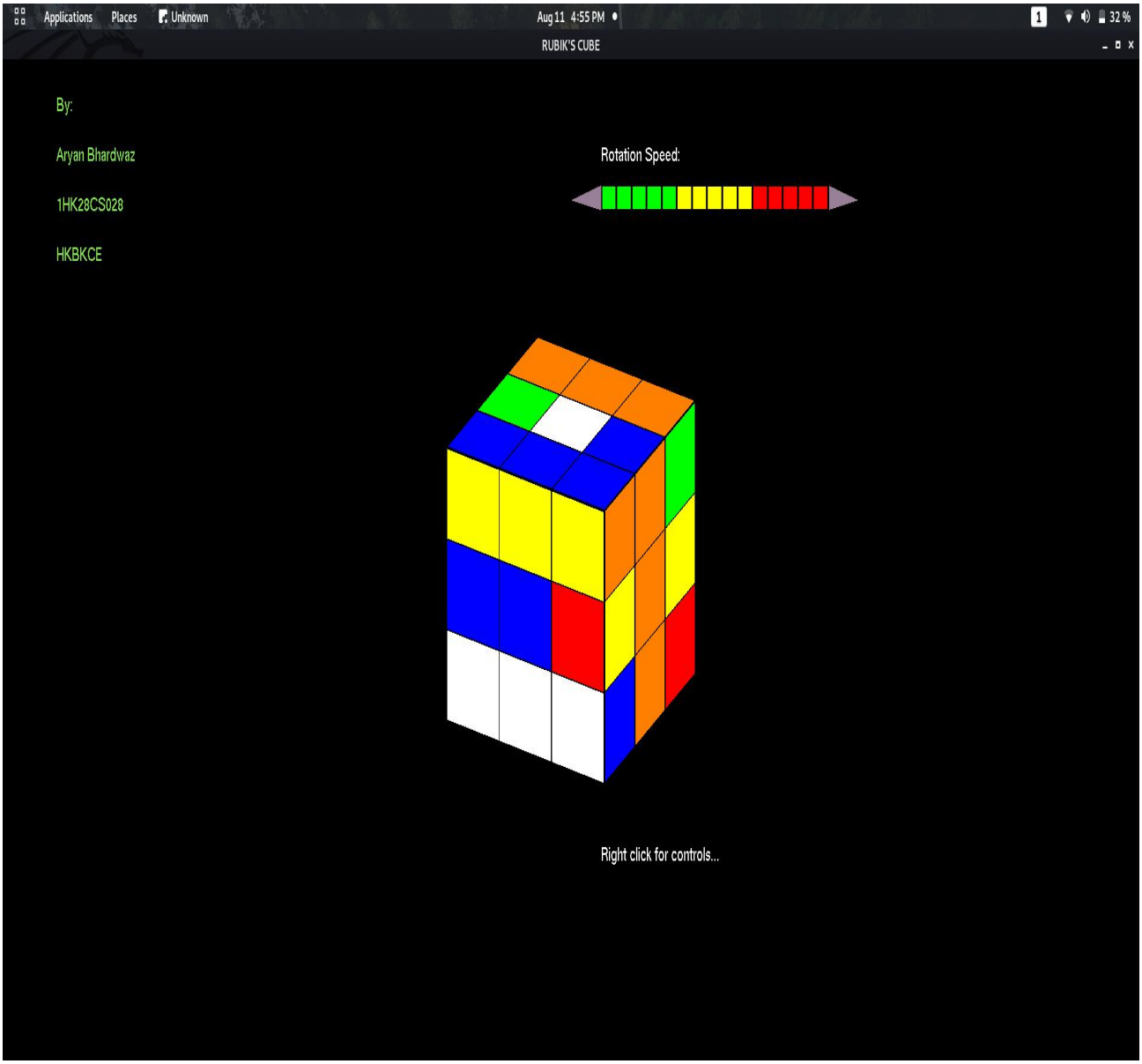


Fig 5.9: Maximum Rotation Speed:

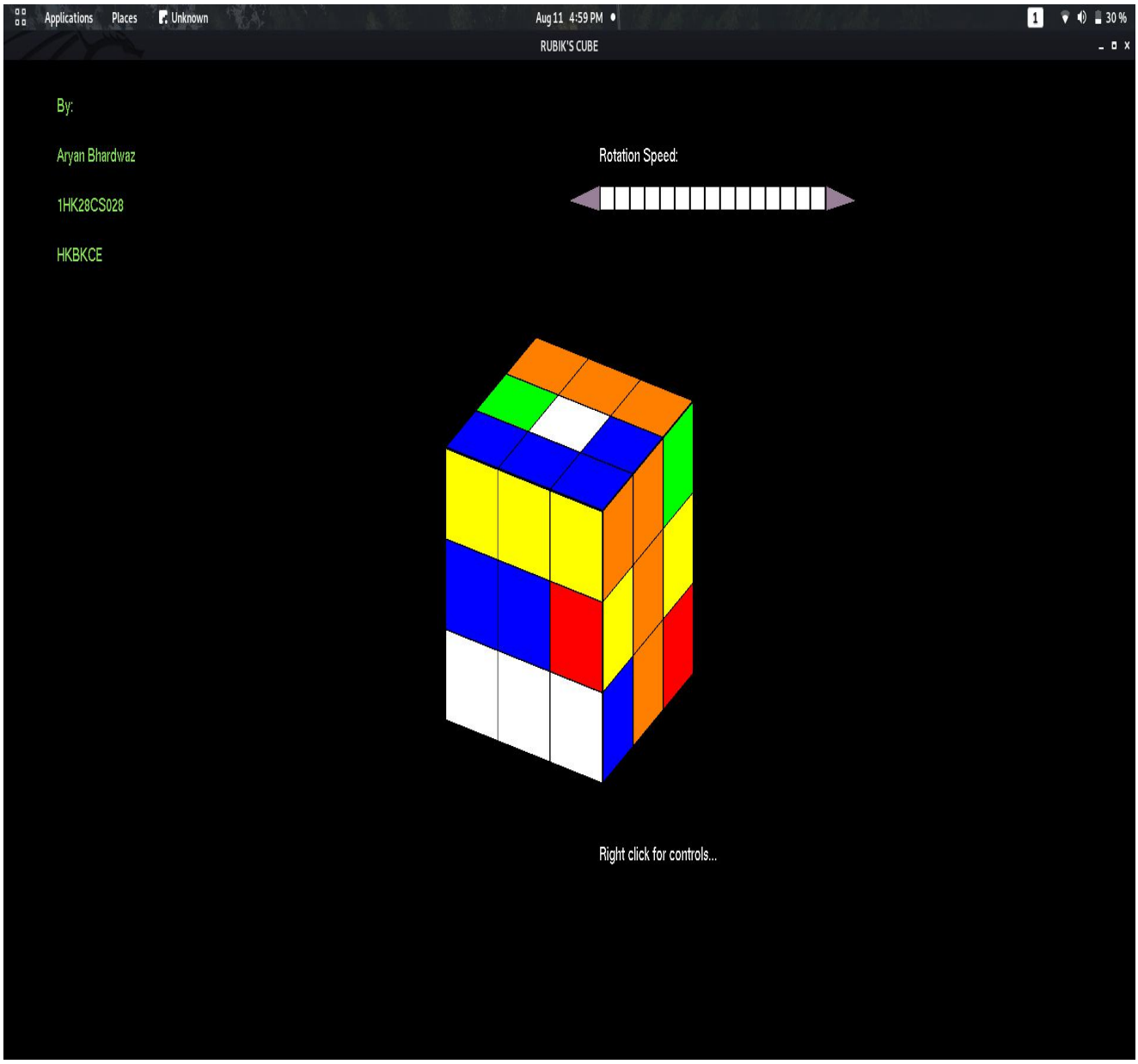


Fig 5.10: Minimum Rotation Speed

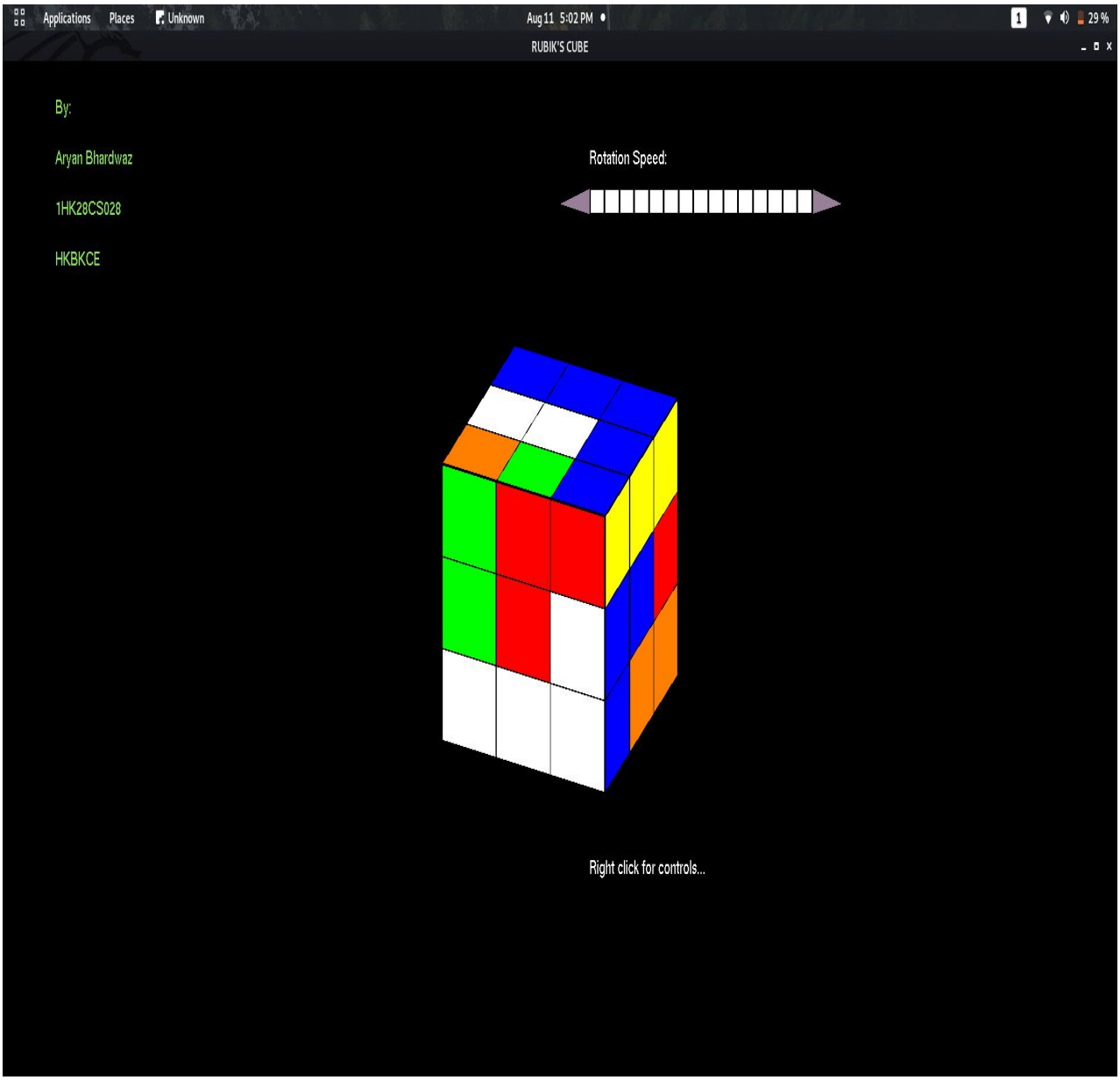


Fig 5.11: Rotations through X-Axix:

CHAPTER 6:

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION:

We successfully implemented the Puzzle Slider game with tiles and frame.The extra features include Player’s name and a clock that starts at the beginning of the game. Also, a high score list – Hall Of Fame has been included. To make the high score list permanent, a text file is running at the back end**.**We were able to successfully implement a hardware accelerated Rubik’s Cube Solver on the FPGA. Our accelerated version was significantly faster than running the Thistlewaite’s Algorithm on the FPGA without the acceleration blocks. After trying different a few different VGA display options we settled on a VGA display that we believe is the most user friendly to make it as easy as possible for the user to enter the inputs.

6.2 FUTURE ENHANCEMENTS

Implement texture mapping to map images onto the tiles instead of the current Roman Numerals. Adding cheat codes like most commercial games, so that the player can enter them and the fully solved puzzle is displayed. This can also be used to view the effects when a player makes it to the Hall of Fame. Give the option of allowing the user to put any image of his choice onto the tiles. This again involves Texture mapping and image processing. Implementing different levels with increasing levels of difficulty. There are several things we would like to add to our project design in the future that we would not have had time to implement in this class. Among those things would include support for non-traditional Rubik’s Cube variations. We would like to add animation to show the solution to the cube rather than just changing

colors on the screen to show what the next cube state should look like. In addition we would like to use color sensors to allow the user to enter the color inputs of the cubies without having to use the keyboard to enter values in individually.

BIBLIOGRAPHY:

[1] Edward Angel’s Interactive Computer Graphics Pearson Education 5th Edition

[2] Interactive computer Graphics --A top down approach using open GL--by Edward Angle

[3] Jackie .L. Neider, Mark Warhol, Tom.R.Davis, "OpenGL Red Book", Second Revised Edition, 2005.

[4] Donald D Hearn and M.Pauline Baker, "Computer Graphics with OpenGL", 3rd Edition.

Links:

www.codeproject.com

www.vtupulse.com

www.stackoverflow.com

www.khronos.org/opengl/wiki/