Visvesvaraya Technological University

Belagavi, Karnataka – 590018

Project report on

TOWER OF HANOI

Submitted in partial fulfillment of the requirements for the course

COMPUTER GRAPHICS LABORATORY WITH MINI PROJECT

(18CSL67)

Submitted by

DARSHAN PS	**1JS20CS055**
CHIDANAND SV	**1JS20CS050**

Under the guidance of

Mrs. RASHMI BN **B. E, M. Tech**

Assistant professor, Dept. of CSE,

JSS Academy of Technical Education, Bengaluru

JSS Academy of Technical Education, Bengaluru – 560060,Department of Computer Science and Engineering

2022-2023

JSS Mahavidyapeetha, Mysuru

JSS Academy of Technical Education

JSS Campus, Uttarahalli – Kengeri Main Road, Bengaluru – 560060

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project work entitled "**TOWER OF HANOI**" is a bona fide work carried out by **Mr. DARSHAN PS(1JS20CS055)** and **Mr. CHIDANAND SV(1JS20CS050)** in partial fulfillment of the requirements for the course Computer graphics of 6th semester, Bachelor of engineering in Computer Science and engineering of the Visvesvaraya Technological University, Belagavi, during the academic year 2022 – 2023. It is certified that all corrections and suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the said degree.

Mrs. Rashmi BN. B.E, MTech	Dr. Mallikarjuna P.B. B.E, MTech,
Ph.D. Assistant professor, Dept. of CSE,	Asso Professor and Head, Dept. of CSE
JSSATE, Bengaluru	JSSATE, Bengaluru

Name of the examiners	Signature with date
i)	
ii)	

ACKNOWLEDGEMENTS

I express my humble greetings to his holiness **Jagadguru Sri Shivarathri Deshikendra Mahaswamijigalavaru** who has showered their blessings on us for framing our career successfully.

The completion of any project involves the efforts of many people. I’ve been lucky to have received a lot of help and support from Indian Railway aspirants and all other quarters, during the making of this project. I take this opportunity to acknowledge all those, whose guidance and encouragement helped me emerge successful.

I am thankful to the resourceful guidance, timely assistance and graceful gesture of my guide **Mrs. Rashmi BN.**, Assistant professor, Department of Computer Science and engineering, who helped me in every aspect of my project work.

I am also indebted to **Dr. Mallikarjuna P.B.**, Associate Professor and head of department of Computer Science and engineering for the facilities and support extended towards us.

I express our sincere thanks to our beloved principal, **Dr. Bhimasen Soragaon** for having supported us in our academic endeavors.

Last but not the least, I am pleased to express our heart full thanks to all the teaching and non- teaching staff of department of Computer and Science engineering and my friends whohave rendered their help, motivation and support.

DARSHAN PS (1JS20CS055)

CHIDANAND SV(1JS20CS050)

ABSTRACT

The Tower of Hanoi is a classical puzzle applied in the psychology of problem solving and skill learning.

In the standard wooden version, it consists of three vertical pegs and a variable number of disks, usually three to five, with increasing diameter.

The disks have a hole in the middle and are stacked on the left peg in the order of the diameter.

The task is to transfer the stack of disks to the right peg with a minimum of moves. Disks are moved from one peg to another, one at a time.

Each peg can serve as a temporary target for any disk. Larger disks must not be placed above smaller ones.

Performance is measured by the number of moves and/or the time required to complete the task. False moves can also be assessed and classified.

The minimum number of moves is $2^n - 1$, where n is the number of disks (e.g., 31 for the five-disk version).

Approximate mean performances on the first attempt are 22 moves in 2 minutes for the four-disk version and 64 moves in 10 minutes for the five-disk version.

In aged subjects, mean baseline performance declines.

CONTENTS

Chapter 1

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 touchsensitive 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

i. Scientific visualization

ii. Information visualization

iii. Computer vision

iv. Image processing

v. Computational geometry

vi. Computational topology

vii. Applied mathematics

1.4 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

Chapter 2

LITERATURE SURVEY

🎬 Inspiration and source of the idea:

The Tower of Hanoi (also called The problem of Benares Temple[1] or Tower of Brahma or Lucas' Tower[2] and sometimes pluralized as Towers, or simply pyramid puzzle[3]) is a mathematical game or puzzle consisting of three rods and a number of disks of various diameters, which can slide onto any rod.

The puzzle begins with the disks stacked on one rod in order of decreasing size, the smallest at the top, thus approximating a conical shape.

The objective of the puzzle is to move the entire stack to the last rod, obeying the following rules:

i. Only one disk may be moved at a time.

ii. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.

iii. No disk may be placed on top of a disk that is smaller than it.

With 3 disks, the puzzle can be solved in 7 moves. The minimal number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where n is the number of disks.

Chapter 3

SYSTEM REQUIREMENTS

System requirements are intended to communicate in precise way, the functions that the system must provide. To reduce ambiguity, they may be written in a structured form of natural language supplemented by tables and system models.

3.1 HARDWARE REQUIREMENTS

The physical components required are:

i. Processor - Pentium Pro

ii. Memory - 128MB RAM

iii. 40GB Hard Disk Drive

3.2 SOFTWARE REQUIREMENTS

The software used in building this program are as specified:-

i.Operating system – LINUX(UBUNTU 10.4)

ii. Tools : Eclipse

iii. Graphics Library – glut.h 🎬 OpenGL 2.0

Chapter 4

IMPLEMENTATION

A key to solving this puzzle is to recognize that it can be solved by breaking the problem down into a collection of smaller problems and further breaking those problems down into even smaller problems until a solution is reached. For example:

i. Label the pegs A, B, C

ii. Let n be the total number of discs

iii. Number the discs from 1 (smallest, topmost) to n (largest, bottommost)

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;

}
```

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);

4.1 DESIGN:

The flow chart describes how the path travels.

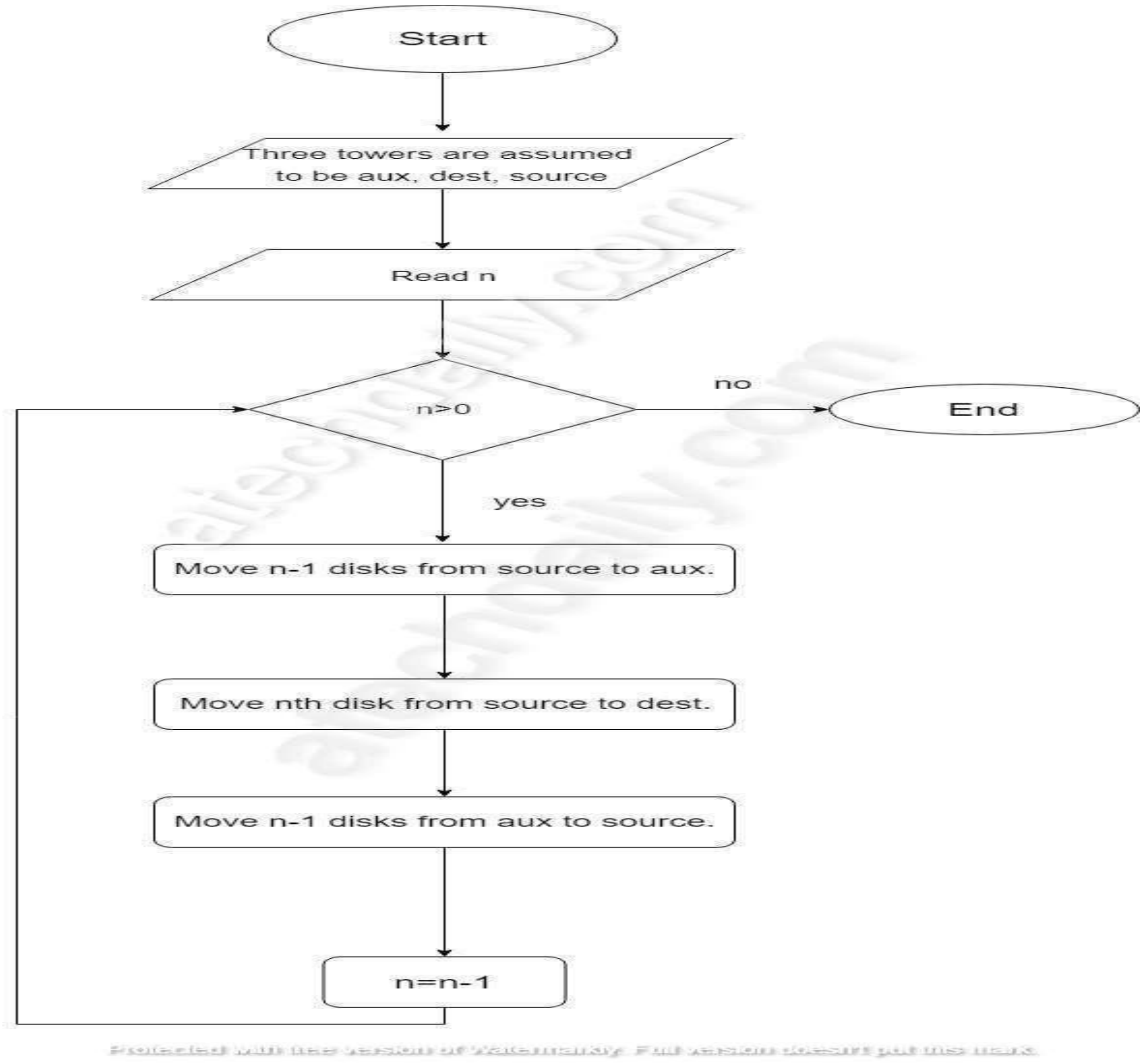

Start
Three towers are assumed to be aux, dest, source
Read n
n>0
no
End
yes
Move n-1 disks from source to aux.
Move nth disk from source to dest.
Move n-1 disks from aux to source.
n=n-1

Fig 4.1: Data flow diagram.

4.2 ALGORITHM:

In Tower Of Hanoi puzzle we have three towers and some disks. We have to move this disk from intial tower to destination tower using aux tower.

For an example lets take we have **two disks** and we want to move it from source to destination tower.

So the approach we will follow

1. First, we will move the top disk to aux tower.

2. Then we will move the next disk (which is the bottom one in this case) to the destination tower.And at last, we will move the disk from aux tower to the destination tower.

Similarly if we will have n number of disk then our aim is to move bottom which is disk n from source to destination. And then put all other **(n-1)** disks onto it.

Steps we will follow is

Step 1 − Move n-1 disks from source to aux

Step 2 − Move nth disk from source to dest

Step 3 − Move n-1 disks from aux to dest

Means to move **n** > **1** disks from tower 1 to tower 2, using auxiliary tower 3

Step 1- Move n – 1 disks from Tower 1 to tower 3

Step 2 – Move the nt^{h} disk from Tower 1 to Tower 2

Step 3 – Move n – 1 disk from Tower 3 to Tower 2

4.3 SOURCE CODE

```
#include<windows.h>
#include<GL/glut.h>
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>

int pos[16] = {10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85};
int peg[3] = {50,150,250};
int moves[10000][3];
int max_moves;
int POLES[3][7];
int top[3]={-1,-1,-1};
int NUM_DISKS=3;
int cnt,counter,speed=10;
int line1=90,line2=85;
float ycoordinate;
int lightflag=1,animationFlag=1;

void push(int p,int disk)
{
    POLES[p][++top[p]] = disk;
}

void pop(int p)
{
    top[p]--;
}

void tower(int n,int src,int temp,int dst)
{
    if(n>0)
    {
        tower(n-1,src,dst,temp);
        moves[cnt][0] = n;
        moves[cnt][1] = src;
        moves[cnt][2] = dst;
        cnt++;
        tower(n-1,temp,src,dst);
    }
}

void drawPegs()
{
    int i;
    glColor3f(0.5,0.0,0.1);
    glRectf(50,0,53,68);
    glRectf(150,0,153,68);
      glRectf(250,0,253,68);
    for(i=0;i<3;i++)
    {
        glPushMatrix();
        glTranslatef(peg[i],5,0);
        glRotatef(-90,1,0,0);
            glRectd(-50,-60,50,-25);
        glPopMatrix();
    }

}

void printString(char *text)
{
    int len=strlen(text),i;
    for(i=0;i<len;i++)
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,text[i]);
}

void drawText()
{
    glColor3f(1,1,1);
    glRasterPos3f(-70,line1,0);
    printString("Move  :");
    char str[5];
    sprintf(str, "%d", counter);
    glRasterPos3f(-40,line1,0);
    printString(str);
    glRasterPos3f(-70,line2,0);
    printString("Disk");
    char str1[10];
    sprintf(str1, "%d", moves[counter][0]);
    glRasterPos3f(-50,line2,0);
    printString(str1);
    glRasterPos3f(-40,line2,0);
    printString("from");
    char src[2];
    if(moves[counter][1]==0)strcpy(src,"A");
    else if(moves[counter][1]==1)strcpy(src,"B");
    else strcpy(src,"C");
    glRasterPos3f(-20,line2,0);
    printString(src);
    glRasterPos3f(-10,line2,0);
    printString("to");
    char dst[2];
    if(moves[counter][2]==0)strcpy(dst,"A");
    else if(moves[counter][2]==1)strcpy(dst,"B");
    else strcpy(dst,"C");
    glRasterPos3f(0,line2,0);
    printString(dst);
    glColor3f(0.6,0.7,0.8);
    glBegin(GL_POLYGON);
        glVertex3f(-75,93,-5);
        glVertex3f(-75,83,-5);
        glVertex3f(10,83,-5);
        glVertex3f(10,93,-5);
    glEnd();
    glColor3f(1,0,0);
    glRasterPos3f(peg[0],70,0);
    glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,'A');
    glRasterPos3f(peg[1],70,0);
    glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,'B');
    glRasterPos3f(peg[2],70,0);
    glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,'C');
}

void drawSolved()
{
    glColor3f(1,1,0);
    glRasterPos3f(-60,87,0);
    printString("Solved !!");
    glColor3f(0.6,0.7,0.8);
    glBegin(GL_POLYGON);
        glVertex3f(-75,93,-5);
        glVertex3f(-75,83,-5);
        glVertex3f(10,83,-5);
        glVertex3f(10,93,-5);
    glEnd();
    glColor3f(1,0,0);
    glRasterPos3f(peg[0],70,0);
    glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,'A');
    glRasterPos3f(peg[1],70,0);
    glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,'B');
    glRasterPos3f(peg[2],70,0);
    glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,'C');
}

void display()
{
    int i,j,k;
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    if(lightflag)glEnable(GL_LIGHTING);
    glPushMatrix();
    gluLookAt(0,ycoordinate,0,0,0,-1,0,1,0);
    drawPegs();
    for(i=0;i<3;i++)
    {
        k=0;
        for(j=0;j<=top[i];j++)
        {
            glPushMatrix();
            glTranslatef(peg[i],pos[k++],0);
            glRotatef(90,1,0,0);
            glColor3f(0.1*POLES[i][j],0.3*POLES[i][j],0);
            glutSolidTorus(1.5, 7*POLES[i][j], 4, 4);
            glPopMatrix();
        }
    }
    glPopMatrix();
    glDisable(GL_LIGHTING);
    if(counter==max_moves)
        drawSolved();
    else
        drawText();
    if(lightflag)glEnable(GL_LIGHTING);
    glutSwapBuffers();
}
```

```
void lighting()
{
    GLfloat shininess[] = {50};
    GLfloat white[] = {0.6,0.6,0.6,1};
    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIF
FUSE);
    GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
    GLfloat light_position[] = {100,60, 10, 0.0 };
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, white);
    glMaterialfv(GL_FRONT, GL_SPECULAR, white);
    glMaterialfv(GL_FRONT, GL_SHININESS, shininess);
    glEnable(GL_LIGHT0);
}

void init()
{
    glClearColor(0.6,0.7,0.8,0);
    glColor3f(1,0,0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-80,350,-10,100,-100,100);
    glMatrixMode(GL_MODELVIEW);
    glEnable(GL_DEPTH_TEST);
    glShadeModel(GL_SMOOTH);
    lighting();
}

void animate(int n,int src,int dest)
{
    int i;
    if(speed<=0)speed=1;
    for(i=pos[top[src]+1];i<90;i+=speed)
    {
        glPushMatrix();
        glTranslatef(peg[src],i,0);
        glRotatef(85,1,0,0);
        glColor3f(0.1*n,0.3*n,0);
        glutSolidTorus(1.5, 7*n, 4, 4);
        glPopMatrix();
        glutSwapBuffers();
        display();
    }
    if(peg[src]<peg[dest])
        for(i=peg[src];i<=peg[dest];i+=speed)
        {
            glPushMatrix();
            glTranslatef(i,90,0);
            glRotatef(85,1,0,0);
            glColor3f(0.1*n,0.3*n,0);
            glutSolidTorus(1.5, 7*n, 4, 4);
            glPopMatrix();
            glutSwapBuffers();
            display();
        }
    else
        for(i=peg[src];i>=peg[dest];i-=speed)
        {
            glPushMatrix();
            glTranslatef(i,90,0);
            glRotatef(85,1,0,0);
            glColor3f(0.1*n,0.3*n,0);
            glutSolidTorus(1.5, 7*n, 4, 4);
            glPopMatrix();
            glutSwapBuffers();
            display();
        }
    for(i=70;i>pos[top[dest]+1];i-=speed)
    {
        glPushMatrix();
        glTranslatef(peg[dest],i,0);
        glRotatef(85,1,0,0);
        glColor3f(0.1*n,0.3*n,0);
        glutSolidTorus(1.5, 7*n, 4, 4);
        glPopMatrix();
        glutSwapBuffers();
        display();
    }
}

void mouse(int btn,int mode,int x,int y)
```

```
{
    if(btn==GLUT_LEFT_BUTTON && mode == GLUT_DOWN)
    {
        if(counter<max_moves)
        {
            pop(moves[counter][1]);
            if(animationFlag)

    animate(moves[counter][0],moves[counter][1],moves[counter][2]);
            push(moves[counter][2],moves[counter][0]);
            counter++;
        }
    }
    glutPostRedisplay();
}

void restart()
{
    int i;
    memset(POLES,0,sizeof(POLES));
    memset(moves,0,sizeof(POLES));
    memset(top,-1,sizeof(top));
    cnt=0,counter=0;
    ycoordinate=0.1;
    max_moves = pow(2,NUM_DISKS)-1;
    for(i=NUM_DISKS;i>0;i--)
    {
        push(0,i);
    }
    tower(NUM_DISKS,0,1,2);
}

void processMenuMain1(int option)
{

}

void processMenuNumDisks(int option)
{
    NUM_DISKS=option;
    restart();
    glutPostRedisplay();
```

```
}

void createGLUTMenus1()
{
    int menu = glutCreateMenu(processMenuNumDisks);
    glutAddMenuEntry("3",3);
    glutAddMenuEntry("4",4);
    glutAddMenuEntry("5",5);
    glutAddMenuEntry("6",6);
    glutAddMenuEntry("7",7);

    glutCreateMenu(processMenuMain1);
    glutAddSubMenu("Number of Disks",menu);

    glutAttachMenu(GLUT_RIGHT_BUTTON);
}

void strokeString(float x,float y,float sx,float sy,char *string,int width)
{
    char *c;
    glLineWidth(width);
    glPushMatrix();
    glTranslatef(x,y,0);
    glScalef(sx,sy,0);
    for(c=string; *c != '\0'; c++)
    {
        glutStrokeCharacter(GLUT_STROKE_ROMAN, *c);
    }
    glPopMatrix();
}

void initfirst()
{
    glClearColor(0.6,0.7,0.8,0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0,1000,0,1000,-1,1);
    glMatrixMode(GL_MODELVIEW);
}

void first()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0,0,0);
    strokeString(250,850,0.4,0.45,"TOWER OF HANOI",6);
    strokeString(260,350,0.3,0.35,"NUMBER OF DISKS:",3);

    glColor3f(0.3,0,0.5);
    char str[5];
    sprintf(str, "%d", NUM_DISKS);
    strokeString(680,350,0.3,0.35,str,3);

    glColor3f(0.5,0,0);
    strokeString(200,200,0.3,0.3,"Press ENTER button to START",2);
    glutSwapBuffers();
}

void keyboard3(unsigned char c, int x, int y)
{
    switch (c)
    {
        case 'A':
         animationFlag=1;
        break;
        case 'a':
         animationFlag=0;
        break;

        case 'b': glClearColor(0,0,0,0);break;
        case 'w': glClearColor(1,1,1,0);break;
        case 'p': glClearColor(0.6,0.7,0.8,0);break;
        case 'e': exit(0);

        case 'l':
         glDisable(GL_LIGHTING);
         lightflag=0;
        break;
        case 'L':
         glEnable(GL_LIGHTING);
         lightflag=1;
        break;

        case 'r': restart();
    }
    glutPostRedisplay();
}

void keyboard(unsigned char c, int x, int y)
{
 switch(c)
 {
        case 13:
             restart();
             init();
             glutDisplayFunc(display);
             glutKeyboardFunc(keyboard3);
             createGLUTMenus1();
             glutMouseFunc(mouse);
        break;
        case 'e': exit(0);
 }
 glutPostRedisplay();
}

int main(int argc,char** argv)
{
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
 glutInitWindowSize(1024,720);
 glutInitWindowPosition(100,100);
 glutCreateWindow("Tower of Hannoi");
 initfirst();
 glutDisplayFunc(first);
 glutKeyboardFunc(keyboard);
 glutMainLoop();
 return 0;
}
```

Chapter 5

Snap Shots

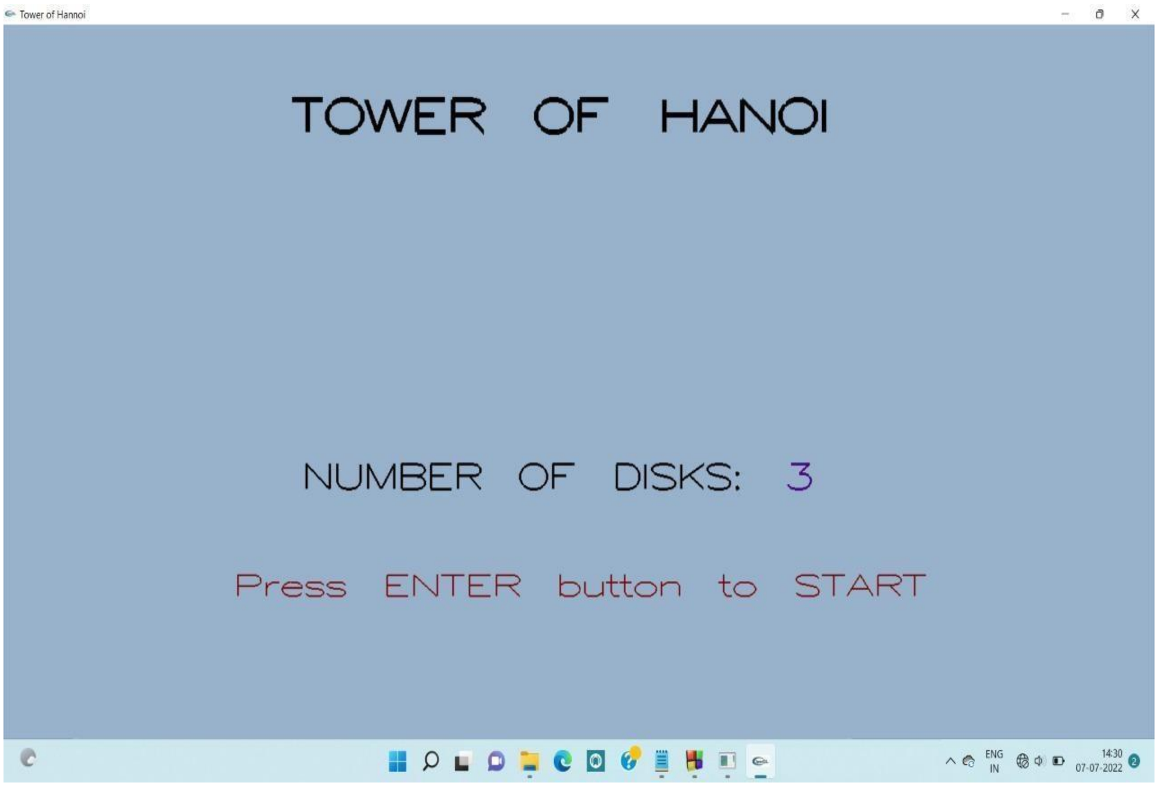


Fig 5.1: Display Page showing the name of the program and the number of disks.

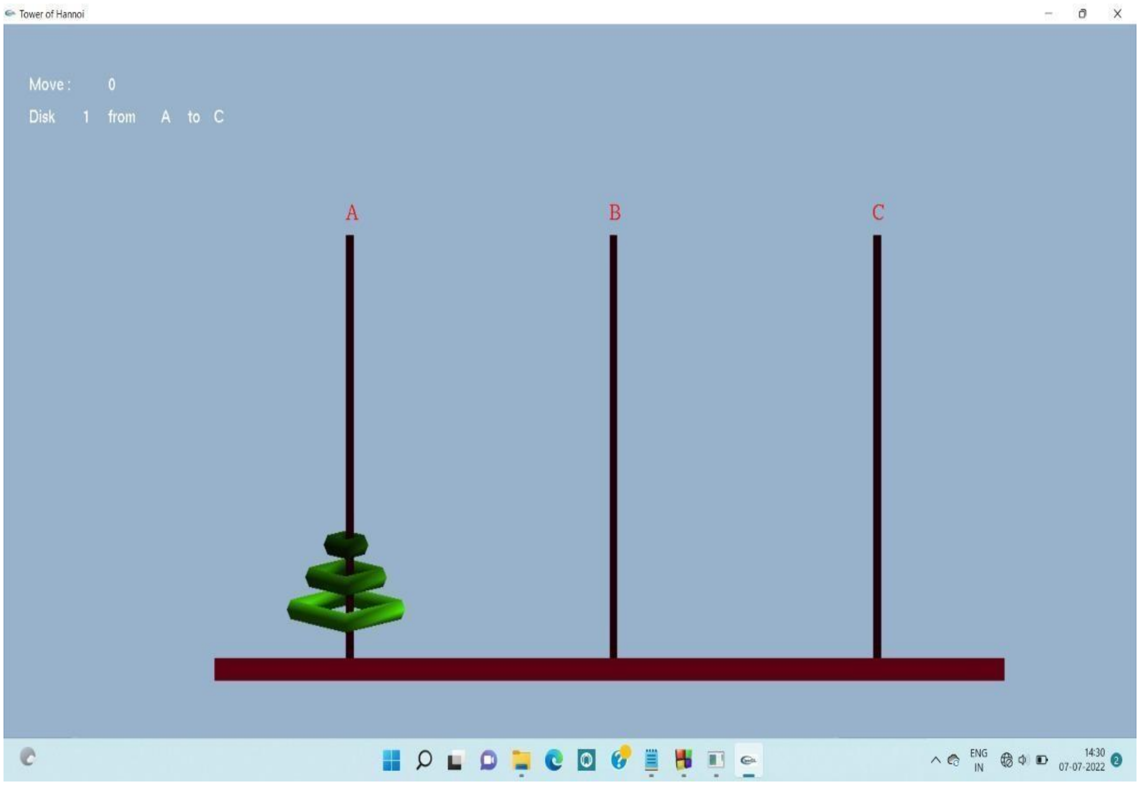


Fig 5.2: Move 0 - Disk 1 from A to C.

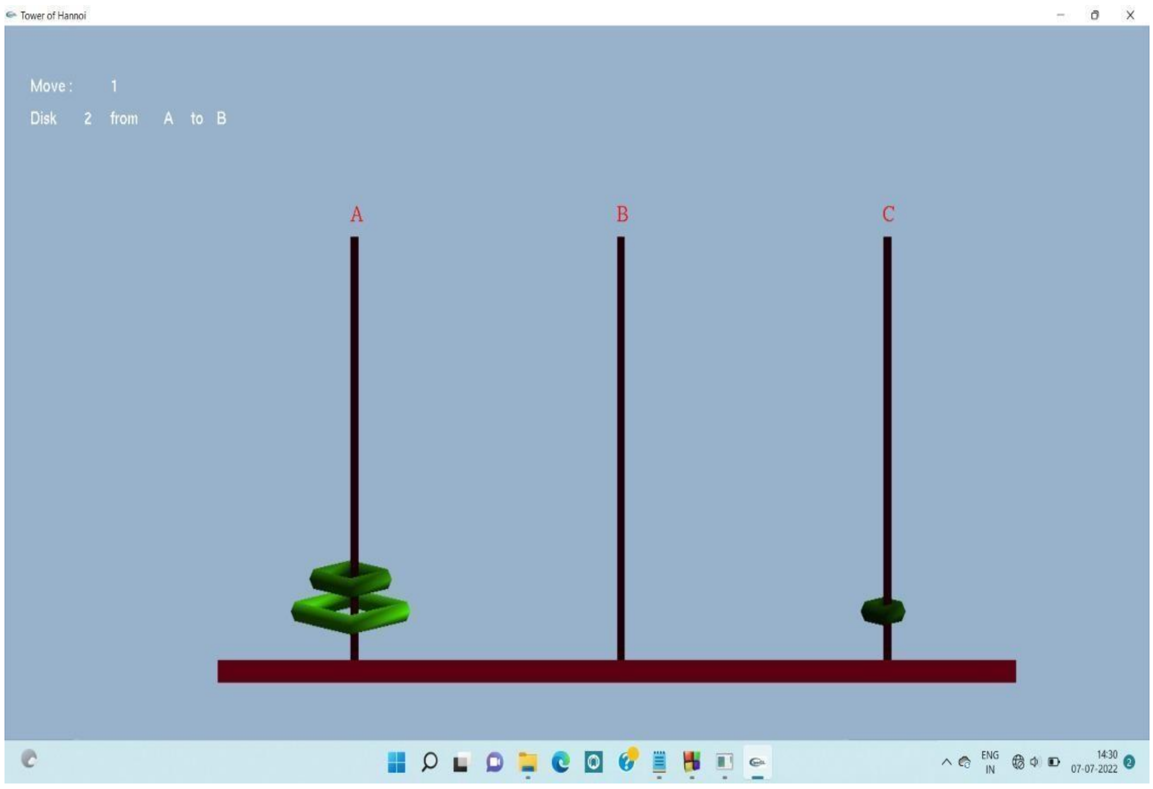


Fig 5.3: Move 1 - Disk 2 from A to B.

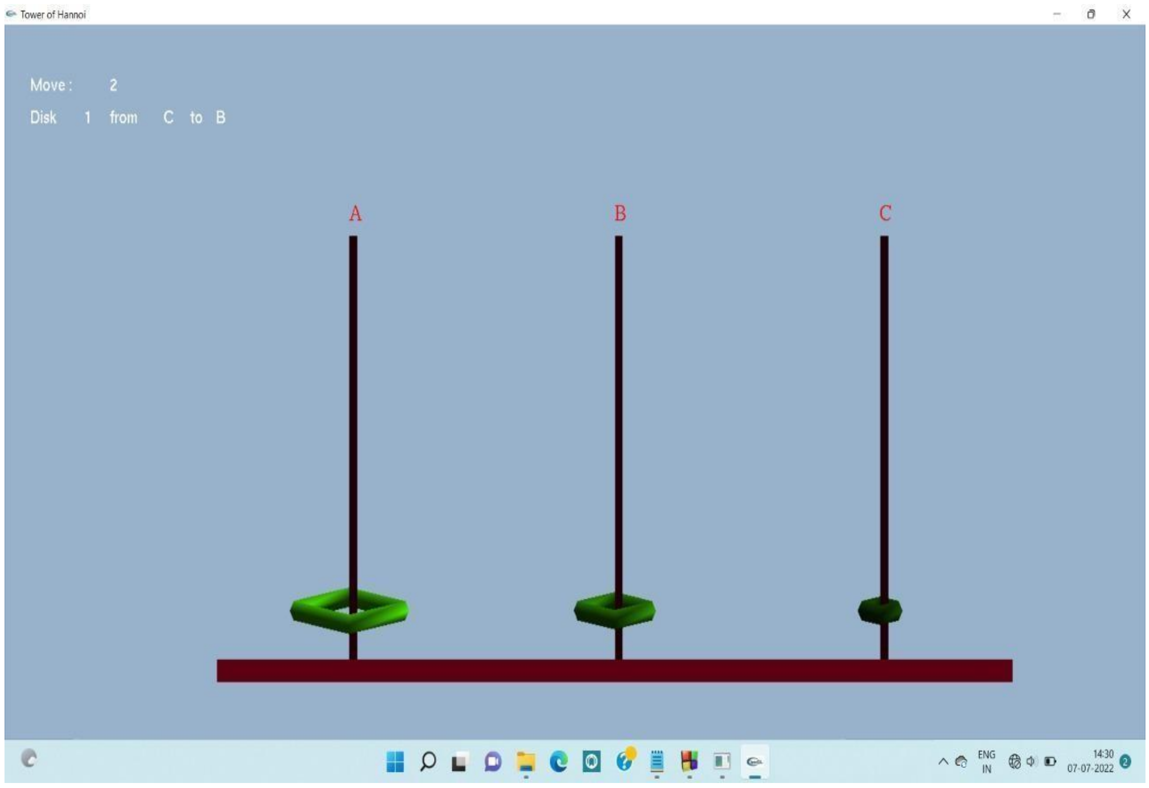


Fig 5.4: Move 2 - Disk 1 from C to B.

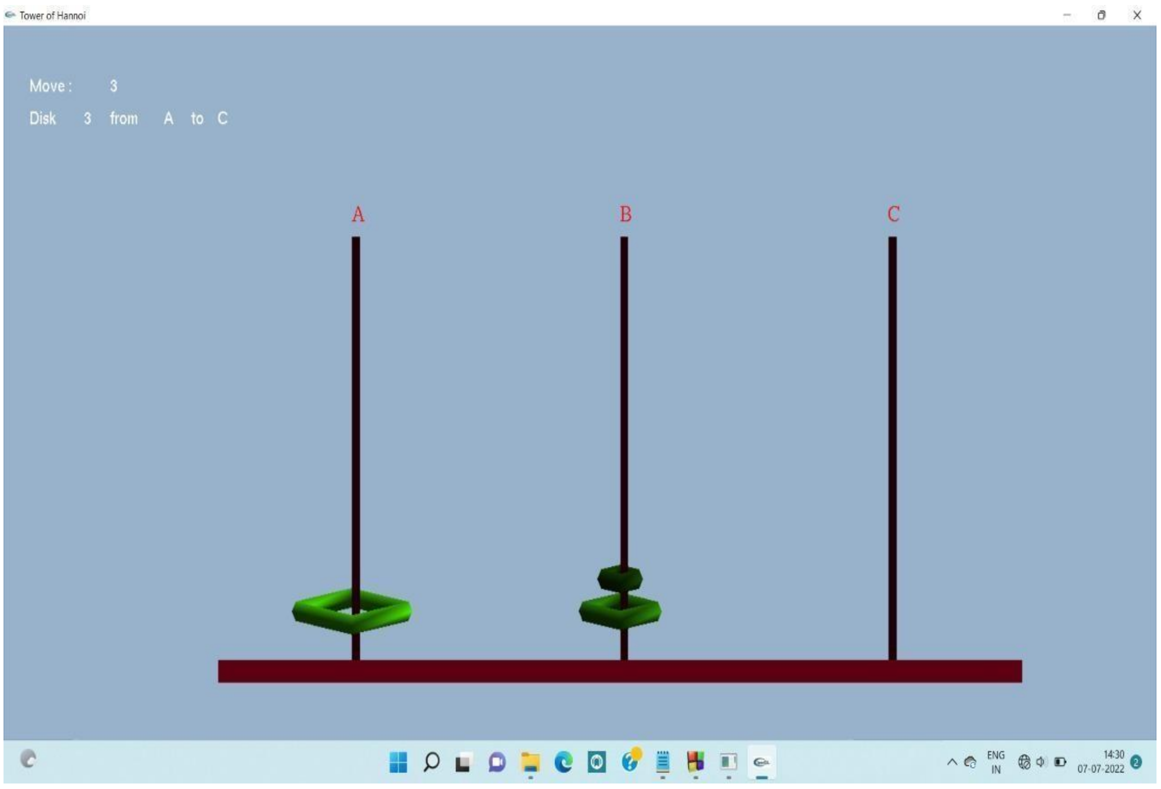


Fig 5.5: Move 3 - Disk 3 from A to C.

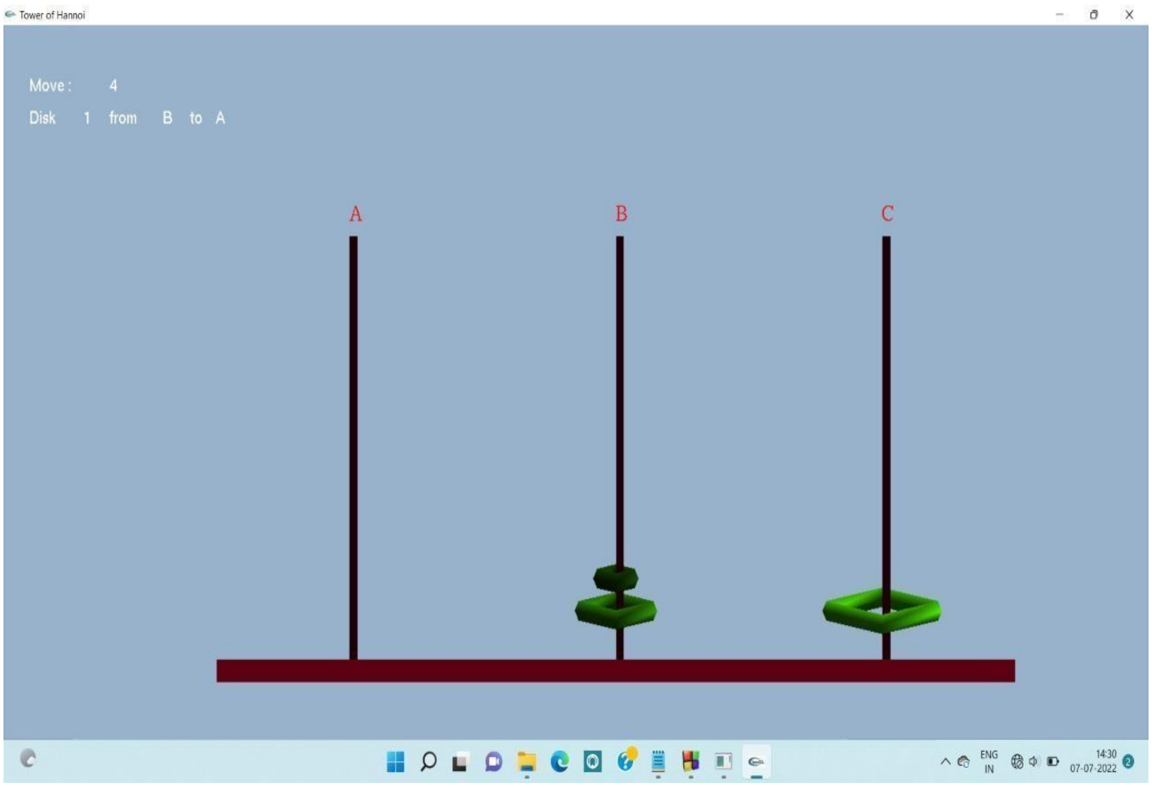
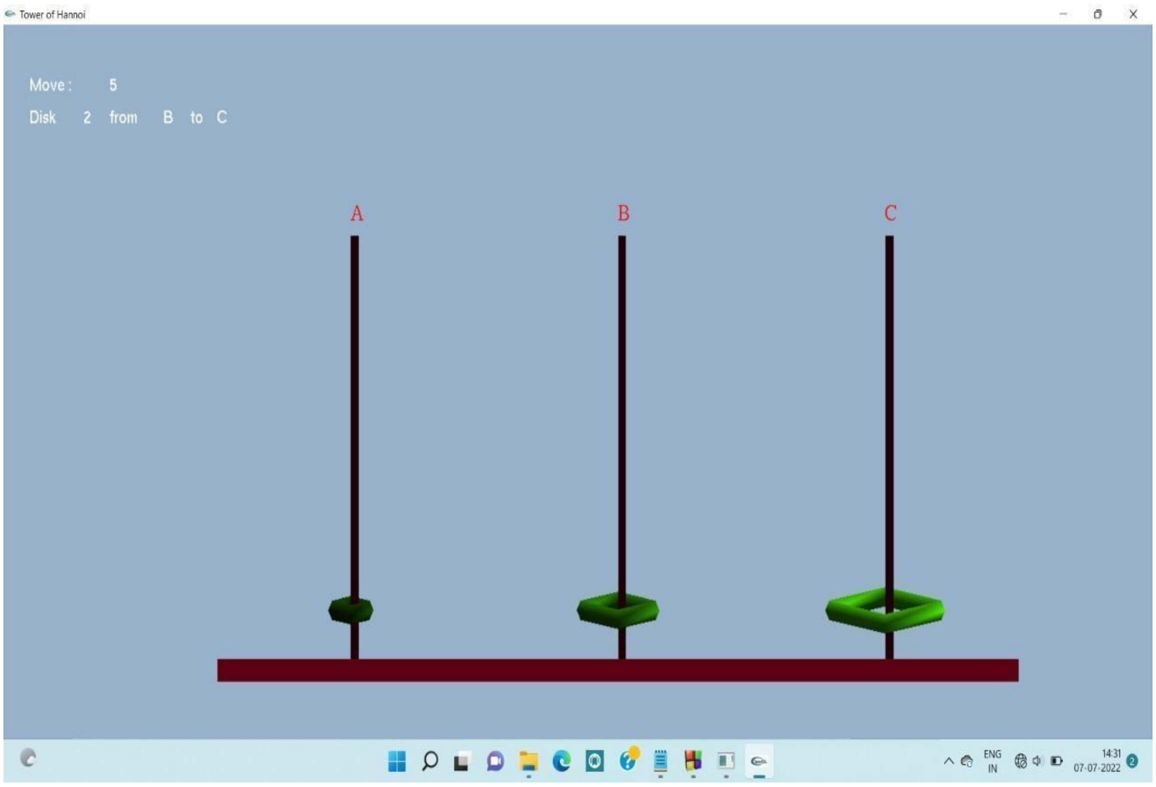
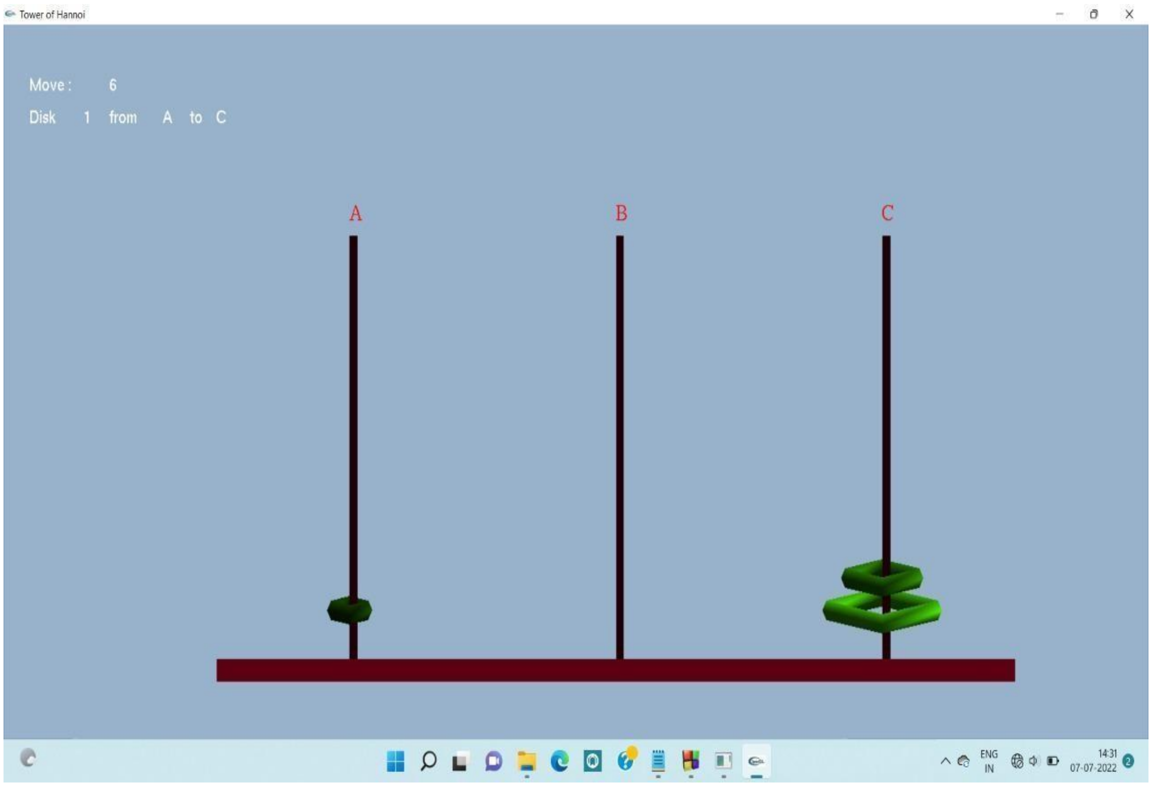


Fig 5.6: Move 4 - Disk 1 from B to A.



Move : 5
Disk 2 from B to C

Fig 5.7: Move 5 - Disk 2 from B to C.



Move : 6
Disk 1 from A to C

Fig 5.8: Move 6 - Disk 1 from A to C.

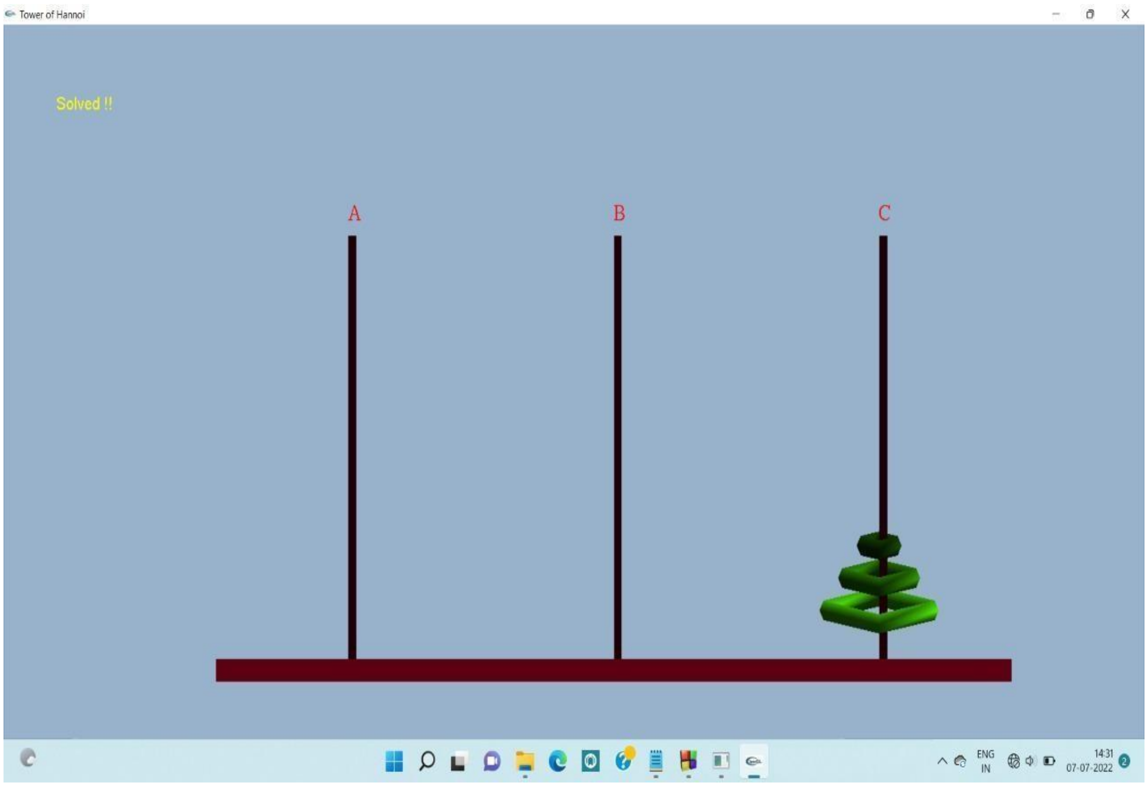
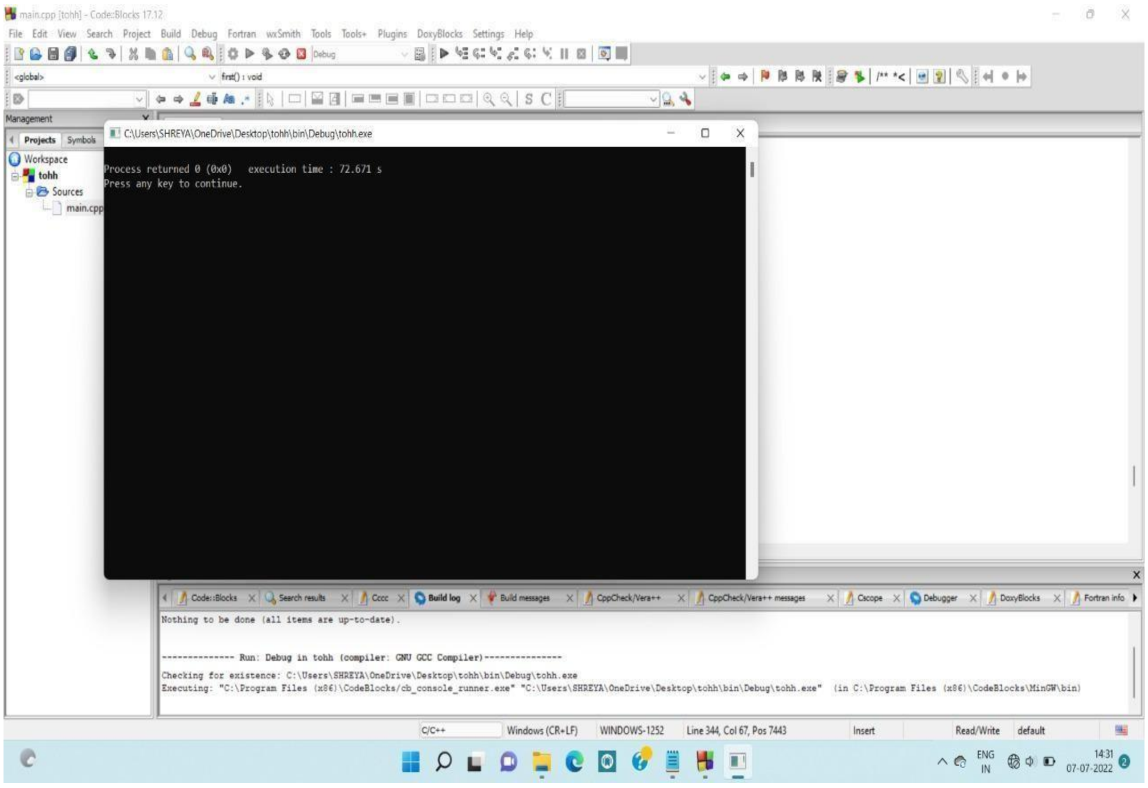


Fig 5.9: Move 7 - Solved.


main.cpp [tohh] - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
first() : void
Management
Projects Symbols
Workspace
tohh
Sources
main.cpp
C:\Users\SHREYA\OneDrive\Desktop\tohh\bin\Debug\tohh.exe
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info
C/C++ Windows (CR+LF) WINDOWS-1252 Line 344, Col 67, Pos 7443 Insert Read/Write default
14:31
07-07-2022

```
Process returned 0 (0x0)   execution time : 72.671 s
Press any key to continue.
```

```
Nothing to be done (all items are up-to-date).


-------------- Run: Debug in tohh (compiler: GNU GCC Compiler)---------------
Checking for existence: C:\Users\SHREYA\OneDrive\Desktop\tohh\bin\Debug\tohh.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "C:\Users\SHREYA\OneDrive\Desktop\tohh\bin\Debug\tohh.exe"  (in C:\Program Files (x86)\CodeBlocks\MinGW\bin)
```

Fig 5.10: Program terminated.

Chapter 6

CONCLUSION

We found designing and developing this project as a very interesting and learning experience.

It helped us to learn about computer graphics, design of Graphical User Interfaces, interface to the user, user interaction handling and screen management. T

his project witnesses the fact that using OpenGL we can develop complete 3D objects and we can animate them.

This program implements projection and viewport transformation functions, viewing and modelling transformation functions.

Thus, we can create programs with higher complexity using the various inbuilt functions supported by OpenGL.

Chapter 7

REFERENCES

6.1 Interactive Computer Graphics

6.2 Edward Angel

6.3 Official OPENGL Documentation at https://www.opengl.org/documentation/

6.4 OpenGL Programming Guide: The Official Guide to Learning OpenGLDave Shreiner.