Visvesvaraya Technological University
Belgaum, Karnataka-590 014

A Project Report on

"SIMULATION OF TOWER OF HANOI PROBLEM USING OPENGL API"

Project Report submitted upon the completion of Mini Project

for the 6th Semester's

Computer Graphics Laboratory with Mini Project(18CSL67)

Submitted by

Soumya Melli 2JR18CS078

Sourabh G Patil 2JR18CS079

Under the Guidance of

Prof. Basavraj Madagouda

Jain Group of Institutions'

Jain College of Engineering and Research, Belagavi

Department of Computer Science and Engineering

2020-21

Jain Group of Institutions'

Jain College of Engineering and Research, Belagavi

Department of Computer Science and Engineering

CERTIFICATE

This is to certified that the project work entitled **"SIMULATION OF TOWER OF HANOI PROBLEM USING OPENGL API"** carried out by Ms. Soumya Melli, USN 2JR18CS078, and Mr. Sourabh Patil, USN 2JR18CS079, bonafide students of Jain College of Engineering and Research, Belagavi, in partial fulfillment for the Mini Project in the subject **Computer Graphics Laboratory with Mini Project** in **Computer Science and Engineering** department of the **Visvesvaraya Technological University, Belagavi** during the year 2020-2021. 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 Project work prescribed for the said subject.

Prof. Basavraj Madagouda
Guide

Dr. Pritam M
Head of Dept.

Examiner **Signature with date:**

1.

2.

ACKNOWLEDGEMENT

On the successful completion of this mini project work, we would like to acknowledge and extend our heartfelt gratitude to the following people who supported us to complete the mini project.

To our beloved Principal **Dr. S. V. Gorabal,** for providing an ideal atmosphere to pursue our objectives under his able administration.

We are also thankful to **Dr. Nandeesh Mathad,** Dean Academics, for creating right kind of milieu and giving moral support.

We are also thankful to **Dr. Pritam Dhumale,** Head of Computer Science and Engineering Department, who has given valuable suggestions during the work along with her moral support and encouragement.

Our sincere gratitude to our mini project coordinator and guide **Prof. Basavraj Madagouda** for providing valuable advice that helped us in the preparation of the mini project work, effective guidance and valuable suggestions right from the beginning of the mini project till its completion, without which this mini project work would not have been accomplished. We are greatly indebted to him.

We would like to express our heartfelt thanks to all the staff members of the department of Computer Science and Engineering for their constructive suggestion and constant encouragement.

We also thank to our parents and all our friends whole heartedly who have rendered their help, motivation and support to accomplish this mini project.

1. INTRODUCTION

The process of designing, 2D, 3D, and animating an object is known as Computer Graphics. Modelling, simulation, and visualisation of an object or a problem are all possible with computer graphics. Since the advent of photography and television, interactive computer graphics has been the most popular means of creating images. It also has the benefit of being able to create images not only of concrete real-world objects but also of abstract, synthetic objects such as mathematical surfaces and data with no apparent geometry, such as survey results, using a computer.

1.1 Objective

The aim of this project is to demonstrate how OpenGL can be used to simulate the Tower of Hanoi problem. The keyboard is the primary means of input in this project.

1.2 Motivation

The use of libraries supported by OpenGL is used in computer graphics to visually reflect the logic we are programming. This will help people understand how Algorithms and Data Structures function. This was the driving force behind selecting the Tower of Hanoi problem. With the principle of dynamic programming, recursion can be given a rigorous mathematical formalism, and Towers of Hanoi can be used as a recursion example when teaching programming.

2. LITERATURE SURVEY

2.1 OpenGL

OpenGL is a graphics rendering API that is independent of operating systems and produces high-quality colour images consisting of geometric and image primitives. This interface is made up of approximately 150 distinct commands that you can use to define the objects and operations used to create immersive three-dimensional applications.

Silicon Graphics Inc. (SGI) created OpenGL in 1992, and it is commonly used in CAD, augmented reality, science visualisation, knowledge visualisation, and flight simulation. It's also used in computer games, where it competes with Direct3D on Microsoft's Windows platform.

2.2 OpenGL Graphics Architecture

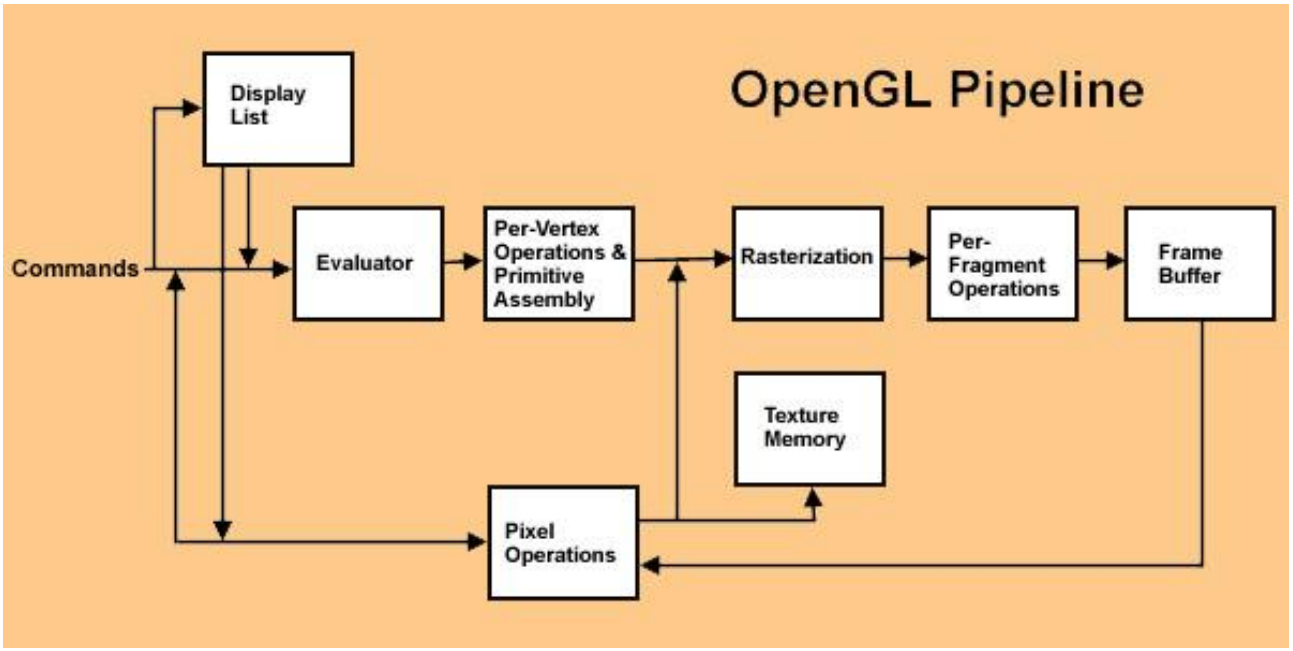

FIGURE 1 – OPENGL GRAPHICS ARCHITECTURE

1. **Display Lists:**

All data, whether it describes geometry or pixels, can be saved in a display list for current or later use. When a display list is executed, the retained data is sent from the display list just as if it were sent by the application in immediate mode.

2. **Evaluators:**

All geometric primitives are eventually described by vertices. Parametric curves and surfaces may be initially described by control points and polynomial functions called basis functions.

3. **Per Vertex Operations:**

For vertex data, next is the "per-vertex operations" stage, which converts the vertices into primitives. Some vertex data are transformed by 4 x 4 floating-point matrices. Spatial coordinates are projected from a position in the 3D world to a position on your screen.

4. **Primitive Assembly:**

Clipping, a major part of primitive assembly, is the elimination of portions of geometry which fall outside a half space, defined by a plane.

5. **Pixel Operation:**

While geometric data takes one path through the OpenGL rendering pipeline, pixel data takes a different route. Pixels from an array in system memory are first unpacked from one of a variety of formats into the proper number of components. Next the data is scaled, biased, and processed by a pixel map. The results are clamped and then either written into texture memory or sent to the rasterization step.

6. **Rasterization:**

Rasterization is the conversion of both geometric and pixel data into fragments. Each fragment square corresponds to a pixel in the frame buffer. Color and depth values are assigned for each fragment square.

7. **Fragment Operations:**

Before values are actually stored into the framebuffer, a series of operations are performed that may alter or even throw out fragments. All these operations can be enabled or disabled.

2.3 Tools required for OpenGL

We'll require a C/C++ compiler, such as GCC from MinGW or Cygwin (for Windows), Visual C/C++ Compiler, or anything else.

We need the following sets of libraries in programming OpenGL:

- **Core OpenGL (GL)**: Core OpenGL models an object using a series of geometric primitives such as point, line, and polygon.
- **OpenGL Utility Library (GLU)**: GLU is designed on top of the core OpenGL and has useful services as well as further construction templates.
- **OpenGL Utilities Toolkit (GLUT)**: Enables interaction with the operating system (such as creating a window, handling key and mouse inputs).
- **OpenGL Extension Wrangler Library (GLEW)**: GLEW is an open-source C/C++ extension loading library that runs on all platforms. GLEW uses powerful run-time mechanisms to figure out which OpenGL extensions are supported on the target platform.

3 DATA FLOW DIAGRAM

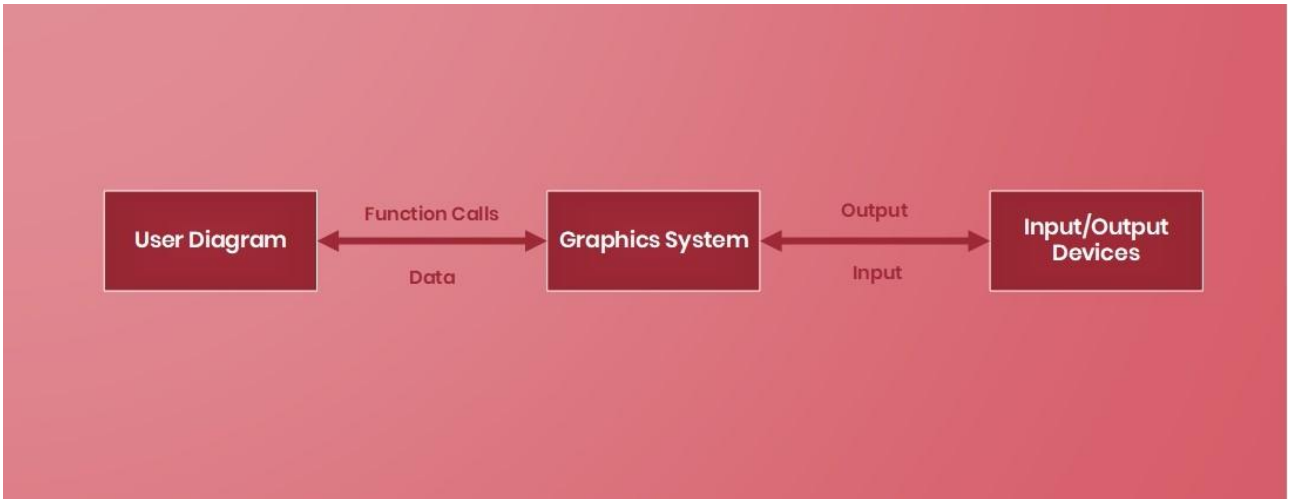

FIGURE #2 – GRAPHICS FUNCTION FLOW

The interaction with the windows is initialized using glutInit() OpenGL API. The display mode-double buffer and depth buffer is, various call back functions for drawing and redrawing, for mouse and keyboard interfaces, input and calculate functions for various mathematical calculations, the window position and size are also initialized and create the window to display the output.

4 METHODOLOGY

4.1 Problem explanation

Three rods and n discs make up the Tower of Hanoi, a mathematical puzzle. The aim of the puzzle is to shift the whole stack to a different rod by following these basic rules:

1. At a given time, only one disc can be transferred.
2. Each move consists of taking the upper disk from one of the pegs and placing it on top of another peg i.e., a disk can only be moved if it is the uppermost disk on a stack.
3. A smaller disc cannot be mounted on top of a larger disc.

4.2 Algorithm

The key to solving this puzzle is to understand that it can be broken down into a series of smaller problems, which can then be broken down further into still smaller problems before a solution is found. For instance:

P.T.O

FIGURE #3 – ALGORITHM FOR THE TOWER OF HANOI

This is a recursive algorithm that can be used for N−1 discs once again. The whole method involves a limited number of steps since the algorithm is necessary at some stage for N = 1. The step of moving a single disc from peg A to peg C, is trivial. With the theory of dynamic programming, this technique can be given a robust mathematical formalism, and it is often used as a recursion example when teaching programming.

4.3 Flowchart

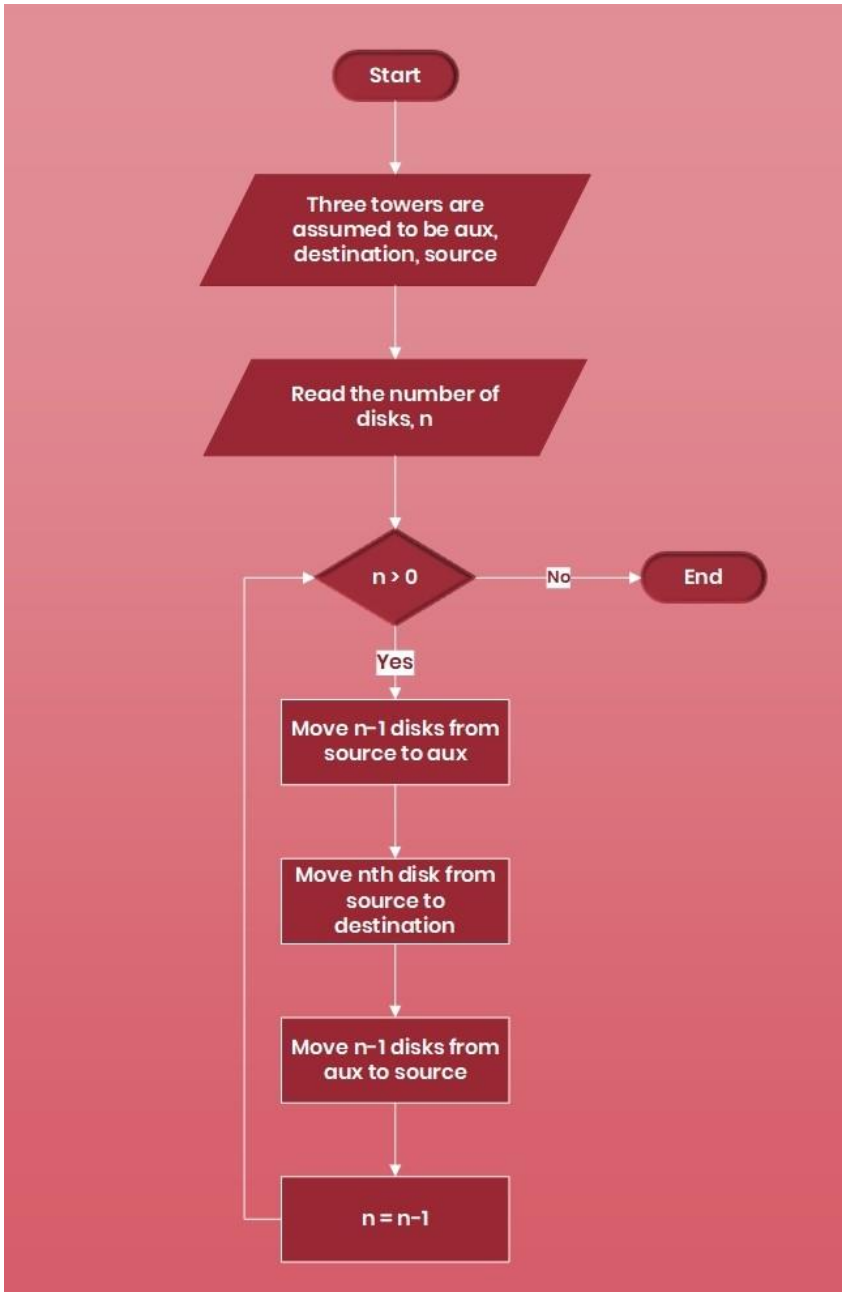

FIGURE #4 – FLOWCHART FOR TOWER OF HANOI

4.4 Built-In Functions

- **void glClear(glEnum mode);**

Clears the buffers namely color buffer and depth buffer. Mode refers to GL_COLOR_BUFFER_BIT or DEPTH_BUFFER_BIT.

- **void glutBitmapCharacter(void *font, int character);**

Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font.

The fonts used are-

  - GLUT_BITMAP_HELVETICA_18.
  - GLUT_BITMAP_TIMES_ROMAN_24.

- **void glutInit(int *argc, char **argv);**

Initializes GLUT; the arguments from main are passed in and can be used by the application.

- **void glutCreateWindow(char *title);**

Creates a window on display; the string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

- **void glutMainLoop();**

Causes the program to enter an event-processing loop.

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

Registers the display function func that is executed when the window needs to be redrawn.

- **void glColor3[b I f d ub us ui](TYPE r, TYPE g, TYPE b, alpha)**

Sets the present RGB colors. Valid types are bytes(b), int(i), float(f), double(d), unsigned byte(ub), unsigned short(us), and unsigned int(ui). The maximum and minimum value for floating point types are 1.0 and 0.0 respectively, whereas the maximum and minimum values of the discrete types are those of the type, for eg, 255 and 0 for unsigned bytes.

- **void glutInitWindowSize(int width, int height);**

  Specifies the initial height and width of the window in pixels.

4.5 Module Description

- **void push(int p, int disk)**

  To push the disk into the rods.

- **void pop(int p)**

  To pop the disk from the rod

- **void tower(int n, int src, int temp, int dst)**

  Initializes the Source, Aux and the destination towers.

- **void drawcylinder()**

  Function to draw the cylinder shape of the disks.

- **void drawPegs()**

  Function to draw the Disks(Pegs).

- **void printString(char *text)**

  Function to render characters in the Helvetica font.

- **voidprintS(char *text)**

  Function to render characters in the Times Roman font.

- **void drawText()**

Function to display text on the display window.

- **void displayName()**

  Function to display the Name, USN of the student.

- **void drawSolved()**

  Function to display the text after the simulation is done.

- **void display()**

  The display handler for the window.

- **void lighting()**

  Function that maintains the lighting model of the project.

- **void mouse(int btn, int mode, int x, int y)**

  Function to provide input to the program from the Mouse.

- **void restart()**

  Function to restart the simulation.

- **void processMenuRestart(int option)**

  Menu Option for restart.

- **void processMenuExit(int option)**

  Menu Option to exit.

- **void processMenuSolveCompletely(int option)**

  Menu option to solve the problem completely.

- **void processMenuNumDisks(int option)**

Menu option to select the number of disks for the simulation.

- **void createGLUTMenus2()**

  Function to create a Menu.

- **void init()**

  Function to initialize the display parameters.

P.T.O

5 OUTPUT

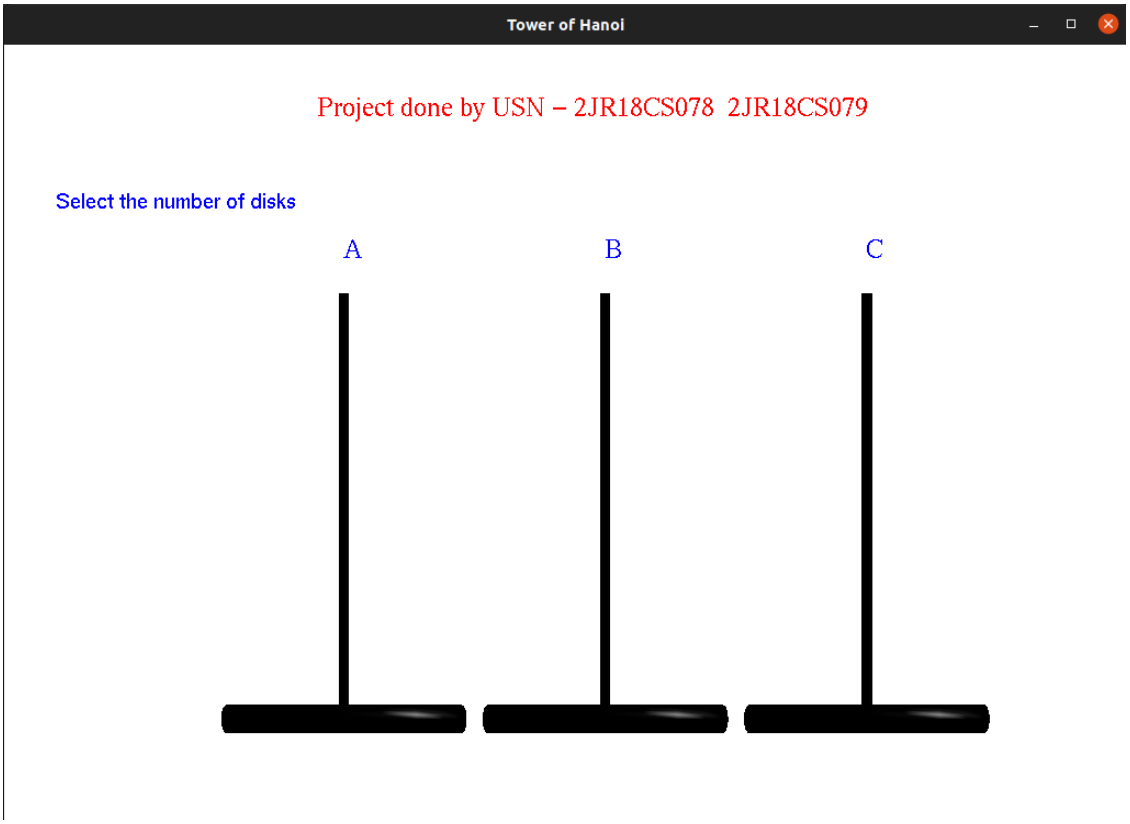

FIGURE #5 – INITIAL SCREEN

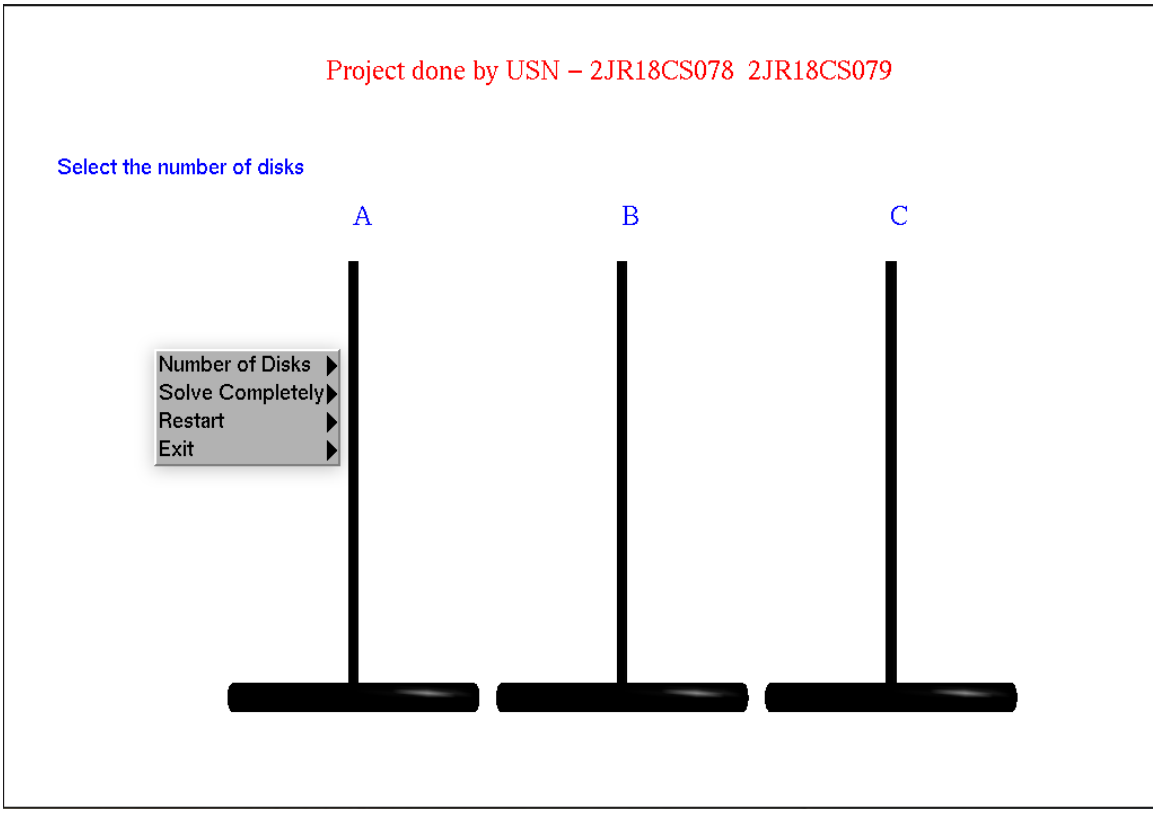

FIGURE #6 – MENU

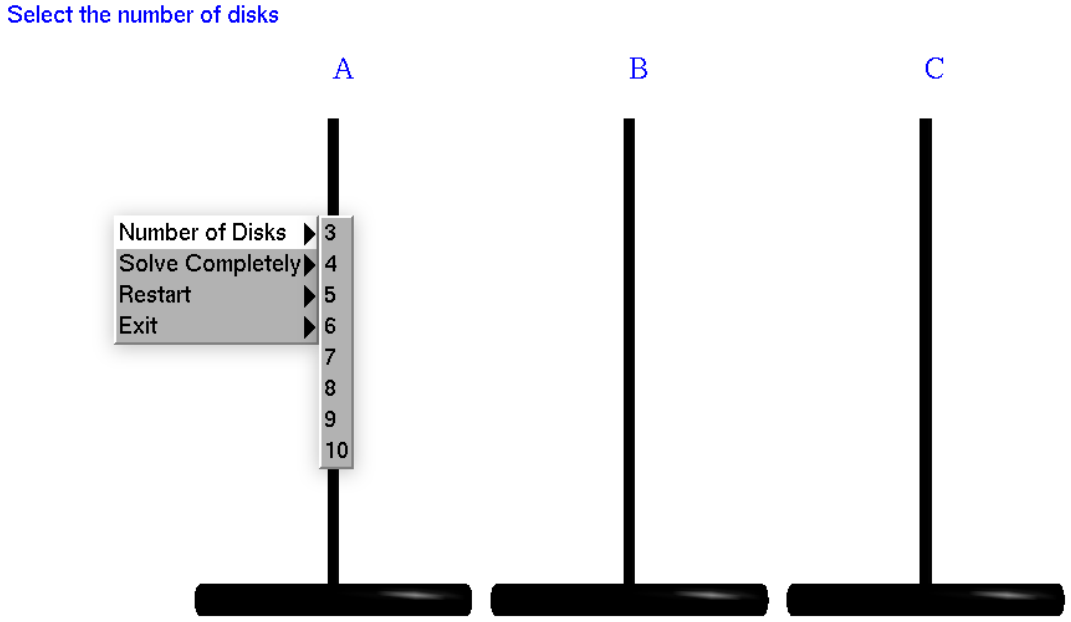

FIGURE #7 – SELECT THE NUMBER OF DISKS

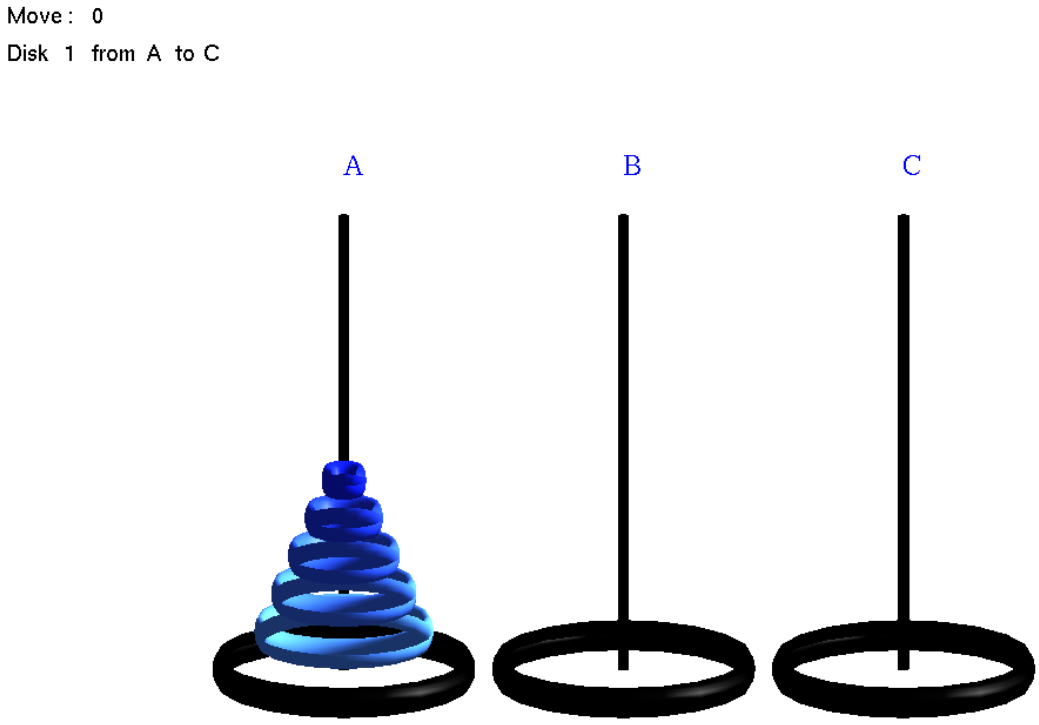

FIGURE #8 – FIVE NUMBER OF DISKS SELECTED

Move : 23

Disk 4 from B to C

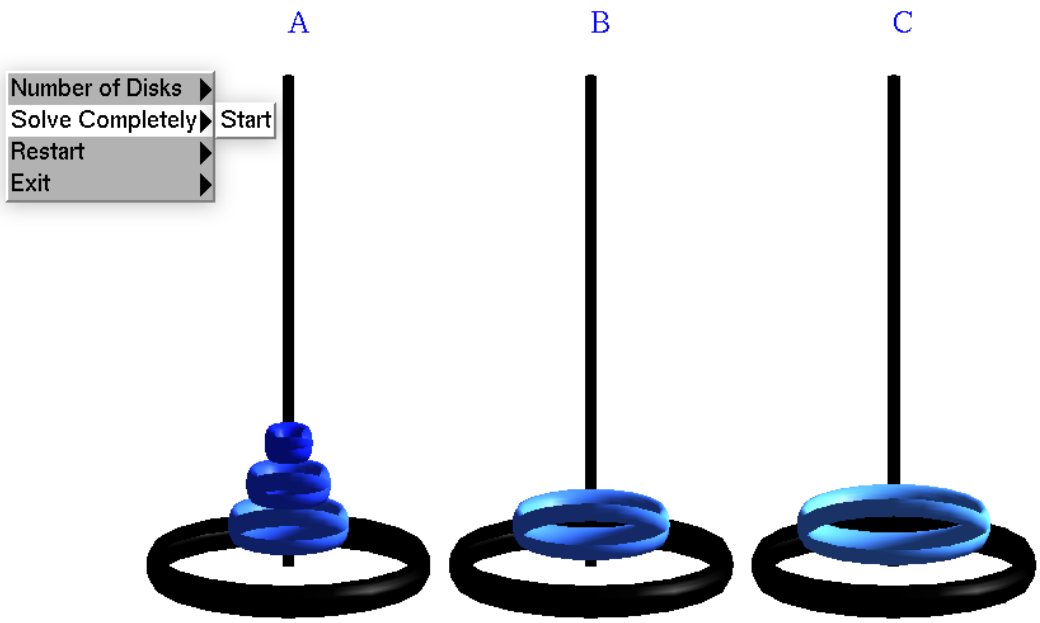

FIGURE #9 – TOWER OF HANOI UNDER SIMULATION

Project done by USN – 2JR18CS078, 2JR18CS079

Solved !!Total number of moves taken = 31

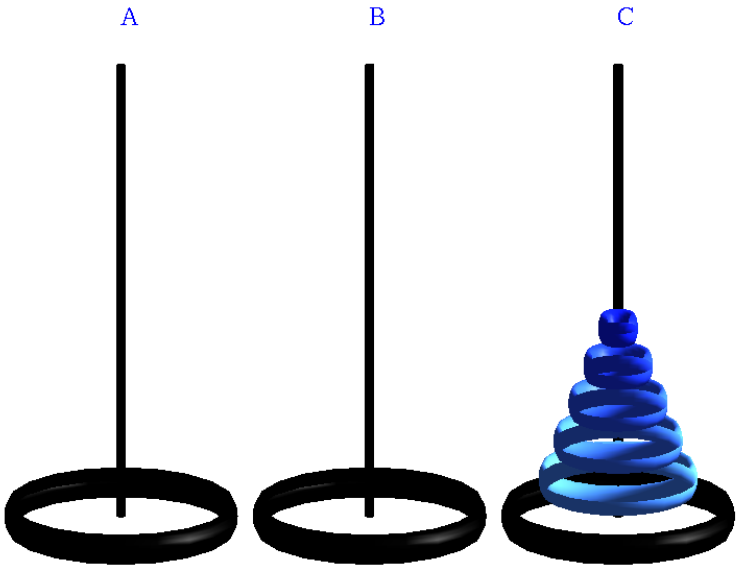

FIGURE #10 – TOWER OF HANOI AFTER SIMULATION

6 CONCLUSION

The project, Tower of Hanoi Simulation was created and implemented using the OpenGL graphics software system, which has become a widely accepted standard for developing graphics applications.

Usage of Open GL functions and primitives are well understood and henceforth can be applied for real time applications.

This project is both informative and entertaining. This project gave a chance to understand the subject's different principles in depth as well as provided a platform to make one's creativity and vision come true. Further animation can be incorporated to improve the overall appearance and feel of the project.

7 REFERENCES

1. Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd / 4th Edition, Pearson Education,2011.

2. Edward Angel: Interactive Computer Graphics- A Top-Down approach with OpenGL, 5th edition. Pearson Education, 2008.

3. Tower of Hanoi recursion game algorithm. - https://www.hackerearth.com/blog/developers/tower-hanoi-recursion-game-algorithm-explained/
4. OpenGL - 2D and 3D Vector Graphics by Victor Gordan.

   https://youtu.be/45MIykWJ-C4

5. OpenGL Documentation – https://www.opengl.org/documentation