

**Visvesvaraya Technological University**

**BELAGAVI, KARNATAKA - 590 014.**

**A MINI PROJECT REPORT**

ON

**“FLOWING FOUNTAIN”**

**USING OPENGL**

**Submitted By**

**NIKHITHA R KARANTH [4PM20CS064]**

**NOOR AYESHA NAAZ TRUPTHI K VAISHNAVI H S**

**[4PM20CS065]**

**[4PM20CS119]**

**[4PM20CS120]**

***Submitted in partial fulfillment of the requirements for the 6th Semester of***

**Bachelor of Engineering in**

**Computer Science and Engineering**

**UNDER THE GUIDANCE OF**

**Dr. Sunitha B S**

**Associate Professor, Dept. of CS&E.**

**PESITM, Shivamoga**

**Department of Computer Science & Engineering**

**July-2023**

**PES Institute of Technology & Management**

**NH-206, Sagar Road, Shimoga-577 204**

**(Affiliated to Visvesvaraya Technological University, Belagavi)**

**PES INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

**Shivamogga-577204**



**Department of Computer Science & Engineering**

**CERTIFICATE**

This is to certify that the project work entitled **“ Flowing Fountain”** is a bonafide work carried out by **Nikhitha R Karanth(4PM20CS064), Noor Ayesha Naaz(4PM20CS065), Trupthi K(4PM20CS119) and Vaishnavi H S(4PM20CS120)**

in partial fulfillment for the 6th Semester of Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum during the academic year 2023. It is certified that all corrections and 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 the Project Work prescribed for the said degree.

|  |  |
| --- | --- |
| **Dr Sunitha B S** | **Dr. Arjun U** |
| Associate Professor,Dept of CS&E. | HOD, Dept of CS&E |
| PESITM, Shivamogga | PESITM, Shivamogga |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name of the examiners** | **Signature with Date** |  |
|  | **1)……………………………** |  |  |
|  | **2)……………………………** |  |  |
|  |  |  |  |
|  |  |  |  |

**ABSTRACT**

The "CG Flowing Fountain" project is a computer graphics application developed using OpenGL. The project aims to create a visually appealing and interactive fountain simulation. It utilizes the power of OpenGL to render a 3D environment with realistic water effects and dynamic particle systems. The user can control various parameters of the fountain, such as its height, shape, and water flow. The application incorporates textures, models, and skybox rendering to enhance the visual experience. Through user input, the program handles events like adjusting the fountain settings and navigating the scene. The project utilizes OpenGL's capabilities for efficient rendering and provides an immersive and engaging virtual fountain experience for users to enjoy.

i

**ACKNOWLEDGEMENT**

I sincerely owe my gratitude to all the persons who helped and guided me in carrying out this project work. I would like to take this opportunity to thank them all.

First of all, I would like to express my immense gratitude to **Dr . Arjun U,** Head of the Department of CS& E, for his help and encouragement.

I owe my profound gratitude to my guide **Dr . Sunitha B.S.,** Associate Professor, Department of CS&E. for her able guidance, regular source of encouragement, and assistance throughout this project.

I express my sincere gratitude to **Dr . Chaitanya Kumar M.V.**, Principal, PESITM, for giving me this opportunity to enrich my knowledge. I am thankful to all the faculty members and the lab staff of the Department of Computer Science & Engineering, PESITM for their constant support.

I would like to thank my parents and friends for their moral support.

Thanking you,

Nikhitha R Karanth(4PM20CS064)

Noor Ayesha Naaz (4PM20CS065)

Trupthi K (4PM20CS119)

Vaishnavi H S (4PM20CS120)

ii

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **CONTENTS** |  |
| **Abstract** |  |  | **i** |
| **Acknowledgement** | | | **ii** |
| **Contents** |  |  | **iii** |
| **List of Figures** | |  | **iv** |
| **Chapters** |  |  |  |
| **1** | **Introduction** | | **2** |
| 1.1 | Computer Graphics | | 2 |
| 1.2 | Applications Of Computer Graphics | | 3 |
| 1.3 | History Of Opengl | | 3 |
|  | 1.3.1 | Functions Of Opengl | 4 |
|  | 1.3.2 | Features Of Opengl | 6 |
| **2** | **Requirement Specification** | | **7** |
| 2.1 | Objective Of The Project | | 7 |
| 2.2 | User Requirement | | 7 |
|  | 2.2.1 | Functional Requirement | 7 |
|  | 2.2.2 | Non-Functional Requirement | 8 |
| 2.3 | System Requirement | | 8 |
|  | 2.3.1 | Hardware Requirement | 8 |
|  | 2.3.2 | Software Requirement | 8 |
| **3** | **System Design and Implementation** | | **9** |
| 3.1 | Design Procedure | | 9 |
| 3.2 | Implementation | | 10 |
|  | 3.2.1 | Graphics Functions | 10 |
|  | 3.2.2 | Flowchart | 13 |
|  | 3.2.3 | Complete Project Code | 14 |
| **4** | **Results** | | **26** |
| 4.1 Snapshots | | | 26 |
| **5** | **Conclusion** | | **31** |
|  | **References** | | **32** |

iii

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Fig. No.** | **NAME** | **Page No**. |
| 1.3.1 | Library Organization of OpenGL | 05 |
| 3.1 | Flow Chart | 15 |
| 4.1 | First Window | 27 |
| 4.2 | Menu Page | 27 |
| 4.3 | Help Window | 28 |
| 4.4 | First output | 28 |
| 4.5 | Near output page | 29 |
| 4.6 | Away output page | 29 |
| 4.7 | Upside output page | 30 |
| 4.8 | Down View output page | 30 |
| 4.9 | Top view output page | 31 |
| 4.10 | Front view output page | 31 |

iv

**CHAPTER 1**

**INTRODUCTION**

The dominant characteristic of this new millennium is how computer and communication technologies have become dominant forces in our life. Activities as wide- ranging as film making, publishing, banking and education continue to undergo revolutionary changes as these technologies alter the ways in which we conduct our daily activities. This combination of computers, networks, and the complex human visual system, through computer graphics, has lead to new ways of displaying information, seeing virtual worlds, and communicating with people and machine.

**1.1 COMPUTER GRAPHICS**

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. Computer Graphics become a powerful tool for the rapid and economical production of pictures. There is virtually no area in which Graphical displays cannot be used to some advantage so it is not surprising to find the use of CG so widespread.

OpenGL is a software interface that allows the programmer to create 2D and 3D graphics images. OpenGL is both a standard API and implementation of that API. The purpose of the project is to demonstrate concepts learnt in OpenGL .There are various in built OpenGL functions defined for development of a graphics application.

Although early application in engineering and science had to rely on expensive and cumbersome equipments, advances in computer technology have made interactive computer graphics a practical tool. Today Computer Graphics is found in a diverse area such as science, engineering, medicine, business, industry, art, entertainment, education and training. Computer graphics can be used as a generalized tool for drawing and creating pictures and stimulates the real world situations within a small computer window. User can create images by computers that are indistinguishable from photographs of real objects.

* 1. **APPLICATIONS OF COMPUTER GRAPHICS**

Nowadays Computer Graphics used in almost all the areas ranges from science, engineering, medicine, business, industry, government, art, entertainment, education and training.

* **CG in the field of CAD**

Computer Aided Design methods are routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft computers, textiles and many other applications.

* **CG in presentation Graphics**

Another major application area presentation graphics used to produce illustrations for reports or generate slides. Presentation graphics is commonly used to summarize financial, statistical, mathematical, scientific data for research reports and other types of reports.2D and 3D bar chart to illustrate some mathematical or statistical report.

* **CG in computer Art**

CG methods are widely used in both fine art and commercial art applications. Artists use a variety of computer methods including special purpose hardware, artist’s paintbrush program (lumena), other pain packages, desktop packages, math packages, animation packages that provide facility for designing object motion. Ex: cartoons decision is an example of computer art which uses CG.

* **Entertainment**

Computer graphics methods are now commonly used in making motion pictures, music, videos, games and sounds. Sometimes graphics objects are combined with the actors and live scenes.

* **Education and Training**

Computer generated models of physical financial, economic system is often as education aids. For some training application special systems are designed. Ex: specialized system is simulator for practice sessions or training of ship captain, aircraft pilots and traffic control.

* **Image Processing**

Although the methods used in CG image processing overlap, the 2 areas are concerned with fundamentally different operations. In CG a computer is used to create picture. Image processing on the other hand applies techniques to modify existing pictures such as photo scans, TV scans.

* **User Interface**

It is common for software packages to provide a graphical interface. A major component of a graphical interface is a window manager that allows a user to display multiple window area. Interface also displays menus, icons for fast selection and processing.

**1.3 HISTORY OF OPENGL**

William fetter was credited with coining the term Computer Graphics in 1960, to describe his work at Boeng. One of the first displays of computer animation was future world (1976), which included an animation of a human face and hand-produced by Carmull and Fred Parkle at University of Utah.

There are several international conferences and journals where the most significant results in computer-graphics are published. Among them are the SIGGRAPH and Euro graphics conferences and the Association for computing machinery (ACM) transaction on Graphics journals.

**1.3.1 FUNCTIONS OF OPENGL**

Most of our application will be designed to access OpenGL directly through functions in three libraries. Functions in the main GL (or OpenGL in windows) library have names that begin with the letters gl and are stored in a library usually referred to as GL (or OpenGL in windows). 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 but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with letters glu.

To interface with the window system and to get input from external devices into our programs, there is a need of at least one more system-specific library. For each major window system there is a system specific library that provides the “glue” between the window system and Open GL. For the X window system, this library is called GLX, for windows it is wgl, and for Macintosh it is agl. Rather than using a different library for each system we use a readily available library called the Open GL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system.

Fig below shows the organization of the libraries for an X Window System environment. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus can be recompiled with the GLUT library for other window systems.

OpenGL application Program

GLU

GL

GLUT

GLX

Xlib, Xtk

Frame Buffer

**Fig 1.3.1 Library organization of OpenGL**

**1.3.2 FEATURES OF OPENGL**

The main features of OpenGL are

* It provides 3D geometric objects such as lines, polygons, triangle, meshes, spheres, cubes, quadric surface, curved surfaces.
* It provides 3D modeling transformations and viewing functions to create views of 3D scenes using the idea of a virtual camera
* It supports high-quality rendering of scenes, including hidden-surface removal, multiple light sources, material types, transparency, textures, blending, fog.

**CHAPTER 2**

**REQUIREMENT SPECIFICATION**

In this section the various requirements that are essential for this project are specified. These requirements have to be fulfilled for execution of the project. The purpose, scope along with hardware and software requirements are given below.

**2.1 OBJECTIVE OF THE PROJECT**

The main purpose of this project is to design and implement “TIC-TAC-TOE” using OpenGL graphics using some of the built in functions. The project allows the user to learn about the aiming and shooting.

**2.2 USER REQIREMENT**

User requirements in project management should be specified in terms of two categories. They are as follows.

1. Functional requirements
2. Non-functional requirement

**2.2.1 Functional requirements:**

* Software should have proper, easy to use interface.
* Software should be user friendly and it should not take much time for the user to learn how to use.
* It should be possible to draw any geometric shapes like lines, triangles, polygon in an easy way.
* It should provide atleast these colors for usage. Red, Green, Blue, Yellow, White, Magenta, Black.
* It should provide white background to draw objects and pick buttons to pick the tools.

**2.2.2 Non-functional requirements:**

These are the constraints on the services of functions offered by the system,they include timing constraints, constraints on the development process and standards and processes. They are:

* The software should be easy to install into the computer.
* It should be very fast and no delay should be present in its operation.
* It should not cause system hanging during any error condition.
* It should be tolerant to user errors.
* It should not create any security problems for the computer.

It should not overload the system

**2.3 SYSTEM REQUIREMENT**

**2.3.1 HARDWARE REQUIREMENTS:**

Processor: Intel 386 onwards compatible hardware

RAM: 256MB RAM (minimum)

Hard Disk: 30GB (minimum)

Monitor: VGA compatible

Keyboard: Standard 101 key keyboard

**2.3.2 SOFTWARE REQUIREMENTS:**

Operating System: Windows XP onwards

Language Tool: OpenGL

Compiler: Visual C++

**CHAPTER 3**

**SYSTEM DESIGN AND IMPLEMENTATION**

The design phase involves the collected requirements to be organized as modules to be implemented. Implementation involves the coding of various user defined functions along with inbuilt functions available in OpenGL.

The architecture of the software is explained below.

* The control starts from the main program; the software initializes some variables and environments and then enters the event driven loop.
* In the event loop, the software waits for an event to occur.
* Whenever there is an event, the appropriate action will be taken by the software.

**3.1 Design Procedure**

* For displaying menu in first, the display list is defined. For each different screen different display list is used. We have about 4 different screens defined in menu function.
* Structure is defined for each of the directions and a class is created for dropping the water via fountain. The Class declaration contains all data members and member functions.
* We have four main functions defined for velocity of water, time of flowing of water, acceleration for water and setting of new position of the droplets of water flowing via the fountain.
* Time and Position variable used along with the accelerating factor to determine the vertex position so we have water seems flowing at different levels. A fountain has several steps, each with its own height.
* Glutmenu function is used to make the project interesting. This allows to have user interaction. User can choose from a range of menu that contains help menu, different functionality and color chooser for water.
* Various key have been assigned with different work for user interaction like moving up and down.
* The most important part played in this project is the use of Display list. Also this is project we utilized the concept of OOP principle.
* The next work is to create the reservoir which is done by taking outer and inner radius and doing so mathematical work. The important part in this was to draw a perfect circular cylinder shape for reservoir.
* To make the Fountain looks real, with water flow we need to shaply apply a good mathematical algorithm. For accomplish this task we defined speed factor as well as random generation of water. Also different angle of water was needed to make it flow (in your eyes), we defined the angles for them. All this clubbed in a function and was used in for loop so it (flowing fountain’s water) continuously flows.
* We have defined a function that allows to have different color for the flowing water in the fountain.

**3.2 Implementation**

**3.2.1 Graphics Functions:**

**void glClear(GL\_COLOR\_BUFFER\_BIT**)**:**

The glClear() function clears a particular buffer or combination of buffers. A buffer is a storage area for image information. The red, green, and blue components of a drawing actually have separate buffers, but they are usually collectively referred to as the color.

**void glBegin(glEnum mode):**

Initiates the new primitive of the type mode and starts the collection of vertices. Values of mode include GL\_POLYGON, GL\_POINTS, etc.,

**void glLoadIdentity( ):**

Sets the current matrix to identity.

**void glutPostRedisplay(void (\*func) (void) ):**

Marks the current window need to be redrawn.

**void** g**lutDisplayFuncd (void (\*func) (void) )):**

Registers the display callback be executed after the current windows.

**void glutMainLoop( ):**

Current program enters an event processing loop.

**void glutInit(int argc, char \*\*argv ):**

glutInit is used to initialize the GLUT library.

**void glutReshapeFunc(void \*f(int width,int height )):**

Registers the reshape callback f. The callback function returns the height and width of the new window. The reshape callback invokes display callback.

**void glutIdleFunc(void (\*f)(void)):**

Registers the display callback function f i.e., executed whenever there are no other events to be handled.

**void glutSwapBuffers( ):**

Swaps the front and back end.

**void gluOrtho2D(GLdouble left,GLdouble right,GLdouble bottom,GLdouble top):**

Defines a two-dimensional viewing rectangle in the plane Z=0.

**void glutInit(int \*\*argc,char \*\*argv):**

Initializes GLUT.

**void glutInitWindowSize(int width, int height):**

Specifies the initial height and width of windows in pixel.

**void glutInitDisplayMode(unsigned int mode):**

Request a display with the properties in mode.

**int glutCreateWindow(char \*title):**

Creates a window on the display returns an identifier

**void glFlush( ):**

Forces any buffered OpenGL commands to execute.

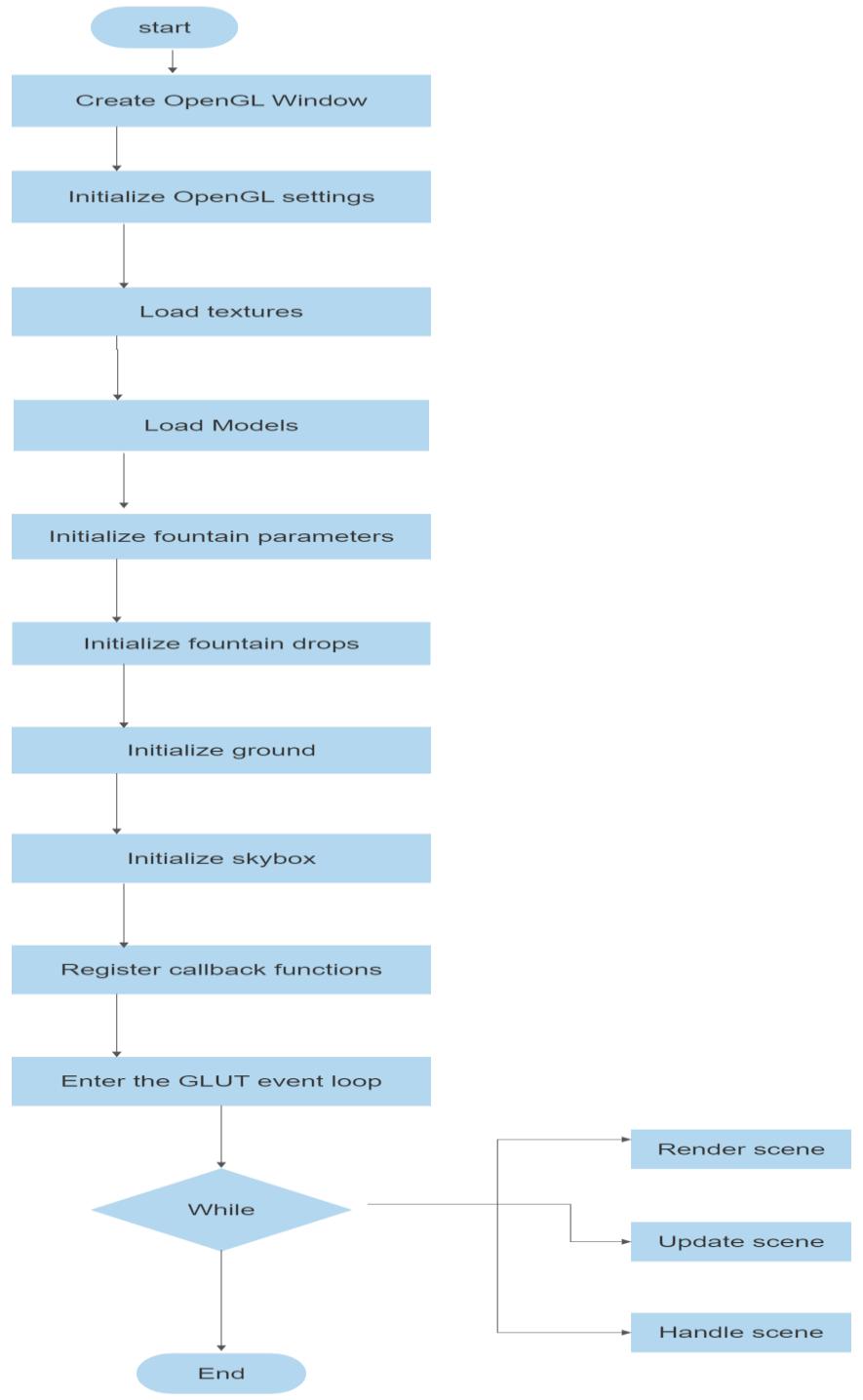
**void glutDisplayFunc(void (\*func)(void)):**

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

**void glutBitmapCharacter(void \*font, int char):**

Renders a character with ASCII code char at the current raster position using the raster font given by font

**3.2.2 Flowchart**



**Fig 3.2.2** **FLOW CHART**

Figure 3.2.2 shows the flow chart of flowing fountain which shows us the steps to be followed to build project.

**3.2.3 Complete Project Code**

#include <GL/glut.h> // Include the GLUT library

#include <stdlib.h> // Include the standard library

#include <math.h> // Include the math library

#include <stdio.h> // Include the standard input/output library

#include <string.h> // Include the string library

#include <iostream> // Include the input/output stream library

#define PI 3.14152653597689786 // Define the value of PI

#define RandomFactor 2.0 // Define the random factor

#define ESCAPE 27 // Define the escape key

#define TEXTID 3 // Define the text ID

unsigned int i; // Declare an unsigned integer variable

int flag = 0, f = 2; // Declare and initialize flag and f variables

int vflag = 0; // Declare and initialize vflag variable

GLfloat xt = 0.0, yt = 0.0, zt = 0.0; // Declare and initialize xt, yt, zt variables GLfloat xangle = 0.0, yangle = 0.0, zangle = 0.0; // Declare and initialize xangle, yangle, zangle variables

GLfloat X[3]; // Declare an array of three GLfloat variables

GLint ListNum; // Declare a GLint variable

GLfloat OuterRadius = 2.4; // Declare and initialize OuterRadius variable

GLfloat InnerRadius = 2.0; // Declare and initialize InnerRadius variable

GLint NumOfVerticesStone = 6; // Declare and initialize NumOfVerticesStone variable

GLfloat StoneHeight = 0.5; // Declare and initialize StoneHeight variable

GLfloat WaterHeight = 0.45; // Declare and initialize WaterHeight variable

struct SVertex // Define a structure called SVertex {

GLfloat x, y, z; // Declare x, y, z variables of type GLfloat

};

class CDrop // Define a class called CDrop {

private:

GLfloat time; // Declare a private GLfloat variable called time

SVertex ConstantSpeed; // Declare a private SVertex variable called ConstantSpeed

GLfloat AccFactor; // Declare a private GLfloat variable called AccFactor

public:

void SetConstantSpeed(SVertex NewSpeed); // Declare a public member function called SetConstantSpeed

void SetAccFactor(GLfloat NewAccFactor); // Declare a public member function called SetAccFactor

void SetTime(GLfloat NewTime); // Declare a public member function called SetTime

void GetNewPosition(SVertex \*PositionVertex); // Declare a public member function called GetNewPosition

};

void CDrop::SetConstantSpeed(SVertex NewSpeed) // Define the SetConstantSpeed member function of the CDrop class

{

ConstantSpeed = NewSpeed;

}

void CDrop::SetAccFactor(GLfloat NewAccFactor) // Define the SetAccFactor member function of the CDrop class

{

AccFactor = NewAccFactor;

}

void CDrop::SetTime(GLfloat NewTime) // Define the SetTime member function of the CDrop class

{

time = NewTime

}

void CDrop::GetNewPosition(SVertex \*PositionVertex) // Define the GetNewPosition member function of the CDrop class

{

SVertex Position;

time += 0.15;

Position.x = ConstantSpeed.x \* time;

Position.y = ConstantSpeed.y \* time - AccFactor \* time \* time;

Position.z = ConstantSpeed.z \* time;

PositionVertex->x = Position.x;

PositionVertex->y = Position.y + WaterHeight;

PositionVertex->z = Position.z;

if (Position.y < 0.0)

{

time = time - int(time);

if (time > 0.0)

time -= 1.0;

}

}

CDrop \*FountainDrops; // Array of fountain drops

SVertex \*FountainVertices; // Array of fountain vertices

GLint Steps = 4; // Number of steps in the fountain

GLint RaysPerStep = 8; // Number of rays per step

GLint DropsPerRay = 80; // Number of drops per ray

GLfloat DropsComplete = Steps \* RaysPerStep \* DropsPerRay;

GLfloat AngleOfDeepestStep = 80;

GLfloat AccFactor = 0.011;

// Create display list for stone void CreateList(void)

{

SVertex \*Vertices = new SVertex[NumOfVerticesStone \* 3]; ListNum = glGenLists(1);

for (GLint i = 0; i < NumOfVerticesStone; i++)

{

Vertices[i].x = cos(2.0 \* PI / NumOfVerticesStone \* i) \* OuterRadius; Vertices[i].y = StoneHeight;

Vertices[i].z = sin(2.0 \* PI / NumOfVerticesStone \* i) \* OuterRadius; for (i = 0; i<NumOfVerticesStone; i++)

{

Vertices[i + NumOfVerticesStone \* 1].x = cos(2.0 \* PI / NumOfVerticesStone

\* i) \* InnerRadius;

Vertices[i + NumOfVerticesStone \* 1].y = StoneHeight;

Vertices[i + NumOfVerticesStone \* 1].z = sin(2.0 \* PI / NumOfVerticesStone

\* i) \* InnerRadius;

}

for (i = 0; i<NumOfVerticesStone; i++){

Vertices[i + NumOfVerticesStone \* 2].x = cos(2.0 \* PI / NumOfVerticesStone

\* i) \* OuterRadius;

Vertices[i + NumOfVerticesStone \* 2].y = 0.0;

Vertices[i + NumOfVerticesStone \* 2].z = sin(2.0 \* PI / NumOfVerticesStone

\* i) \* OuterRadius;

}

glNewList(ListNum, GL\_COMPILE); glBegin(GL\_TRIANGLE\_STRIP);

for (GLint i = 0; i < NumOfVerticesStone \* 2 + 1; i++)

{

if (i < NumOfVerticesStone)

glTexCoord2f(0.0, 0.0);

else if (i % 2 == 0)

glTexCoord2f(0.0, 1.0);

else

glTexCoord2f(1.0, 1.0);

glVertex3f(Vertices[i % NumOfVerticesStone].x,Vertices[i % NumOfVerticesStone].y,Vertices[i % NumOfVerticesStone].

if (i < NumOfVerticesStone)

glTexCoord2f(1.0, 0.0);

else if (i % 2 == 0)

glTexCoord2f(0.0, 1.0);

else

glTexCoord2f(1.0, 1.0);

glVertex3f(Vertices[i % NumOfVerticesStone + NumOfVerticesStone].x,

Vertices[i % NumOfVerticesStone + NumOfVerticesStone].y;

Vertices[i % NumOfVerticesStone + NumOfVerticesStone].z);

}

glEnd();

glEndList();

delete[] Vertices;

}

// Initialize the fountain drops

void InitFountain()

{

SVertex Speed;

GLfloat v = 1.2;

FountainDrops = new CDrop[DropsComplete];

FountainVertices = new SVertex[DropsComplete];

for (int s = 0; s < Steps; s++)

{

for (int r = 0; r < RaysPerStep; r++)

{

for (int d = 0; d < DropsPerRay; d++)

{

Speed.x = v \* sin((AngleOfDeepestStep / Steps \* s +

AngleOfDeepestStep / (Steps \* RaysPerStep) \* r) \* PI / 180) \*

cos(2 \* PI \* d / DropsPerRay)

(GLfloat)rand() / RAND\_MAX \* RandomFactor;

Speed.y = v \* cos((AngleOfDeepestStep / Steps \* s +

AngleOfDeepestStep / (Steps \* RaysPerStep) \* r) \* PI / 180)

(GLfloat)rand() / RAND\_MAX \* RandomFactor;

Speed.z = v \* sin((AngleOfDeepestStep / Steps \* s +

AngleOfDeepestStep / (Steps \* RaysPerStep) \* r) \* PI / 180) \* sin(2 \* PI \* d / DropsPerRay) +

(GLfloat)rand() / RAND\_MAX \* RandomFactor;

FountainDrops[s \* RaysPerStep \* DropsPerRay + r \* DropsPerRay + d].SetConstantSpeed(Speed)

FountainDrops[s \* RaysPerStep \* DropsPerRay + r \* DropsPerRay + d].SetAccFactor(AccFactor);

FountainDrops[s \* RaysPerStep \* DropsPerRay + r \* DropsPerRay + d].SetTime((GLfloat)rand() / RAND\_MAX);

}

}

}

}

// Draw the fountain void DrawFountain()

{

glPushMatrix();

glBindTexture(GL\_TEXTURE\_2D, TextureID); glColor4f(1.0f, 1.0f, 1.0f, 0.8f);

for (int i = 0; i < DropsComplete; i++)

{

FountainVertices[i] = FountainDrops[i].GetPosition();

FountainDrops[i].Move();

}

glEnableClientState(GL\_VERTEX\_ARRAY);

glVertexPointer(3, GL\_FLOAT, 0, FountainVertices);

glDrawArrays(GL\_POINTS, 0, DropsComplete);

glDisableClientState(GL\_VERTEX\_ARRAY);

glBindTexture(GL\_TEXTURE\_2D, 0);

glPopMatrix();

}

// Render the scene void RenderScene()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

// Set the camera

glMatrixMode(GL\_MODELVIEW); glLoadIdentity();

gluLookAt(0.0, 20.0, 80.0, 0.0, 20.0, 0.0, 0.0, 1.0, 0.0);

// Draw the fountain

DrawFountain();

// Draw the ground DrawGround();

// Draw the skybox DrawSkybox();

glutSwapBuffers();

}

// Update the scene

void UpdateScene(int value)

{

glutPostRedisplay();

glutTimerFunc(30, UpdateScene, 0);

}

int main(int argc, char \*argv[])

{

// Initialize GLUT and create a window glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH); glutInitWindowSize(800, 600);

glutCreateWindow("Fountain Simulation");

// Initialize the scene InitGL(); InitTextures(); InitGround()

// Register callback functions glutDisplayFunc(RenderScene); glutReshapeFunc(ChangeSize); glutTimerFunc(30, UpdateScene, 0);

// Enter the GLUT event loop glutMainLoop();

// Clean up

delete[] GroundVertices;

delete[] SkyboxVertices;

delete[] FountainDrops;

delete[] FountainVertices;

return 0;

}

void SetConstantSpeed(SVertex NewSpeed) : The purpose of this function is to set the value of a member variable called ConstantSpeed in the CDrop class.

void SetAccFactor(GLfloat NewAccFactor) : The purpose of this function is to set the value of a member variable called AccFactor in the CDrop class. The AccFactor variable is of type GLfloat.

void SetTime(GLfloat NewTime) : The purpose of this function is to update the value of a member variable called time in the CDrop class. The time variable is of type GLfloat, which is typically used to represent floating-point numbers in OpenGL or similar graphics libraries.

void GetNewPosition(SVertex \* PositionVertex) : The purpose of this function is to calculate and update the position of a CDrop object based on its current speed, acceleration factor, time, and water height.

void CreateList(void) : The `CreateList` function creates an OpenGL display list that defines the geometry and colors of a stone shape.

void InitFountain(void) : The function `InitFountain` initializes a fountain effect in OpenGL. It creates an array of drops and vertices, calculates various factors and angles for each drop, sets their properties, and enables the vertex array for rendering the fountain effect.

void randcolor() : The function randcolor generates random values for three variables a, b, and c within the range of 0 to 100. It then assigns these values divided by 100 to the elements of an array X, resulting in random color values between 0.0 and 1.0 for each RGB component.

void DrawFountain(void) : The function `DrawFountain` is responsible for rendering the fountain effect in OpenGL. It sets the color based on the value of the `flag` variable, updates the positions of the fountain drops, draws the drops as points using `glDrawArrays`, and triggers a redisplay using `glutPostRedisplay`.

void Reshape(int x, int y) : The `Reshape` function is responsible for handling the window reshape event in OpenGL. It sets up the projection matrix with a perspective view, updates the model-view matrix, adjusts the viewport dimensions, and sets the point size based on the window size.

void colours(int id)

void flow(int id)

void level(int id)

void help(int id)

void CMain(int id)

void NormalKey(GLubyte key, GLint x, GLint y)

void DrawTextXY(double x, double y, double z, double scale, const char\*s)

voidDisplay(void)

void menu1()

void menu2()

void cover()

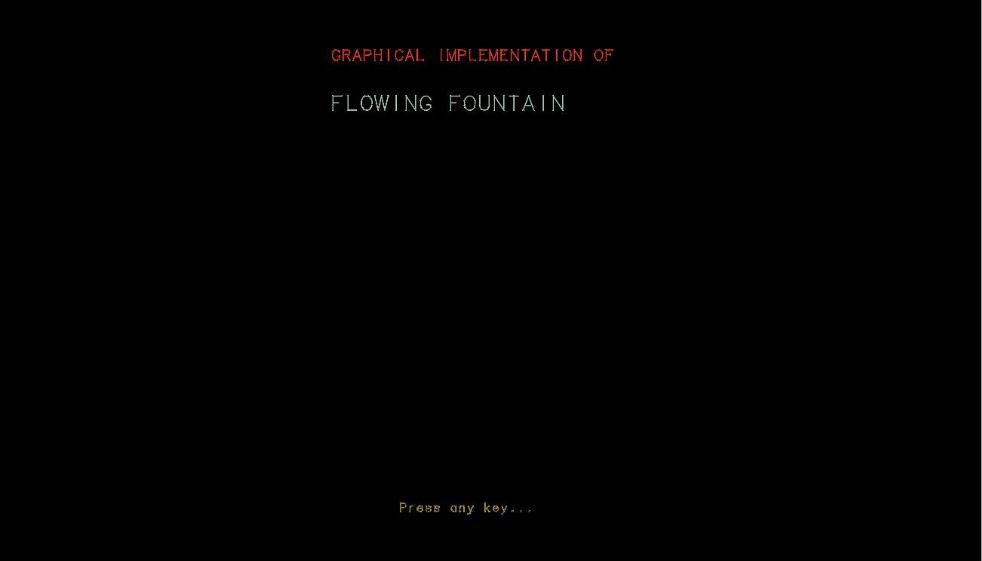
void Dis()

**CHAPTER 4**

**RESULT**

**4.1 SNAPSHOTS**

Results obtained after the completion of the project are shown in this chapter. The snapshots are visual illustration of the project.

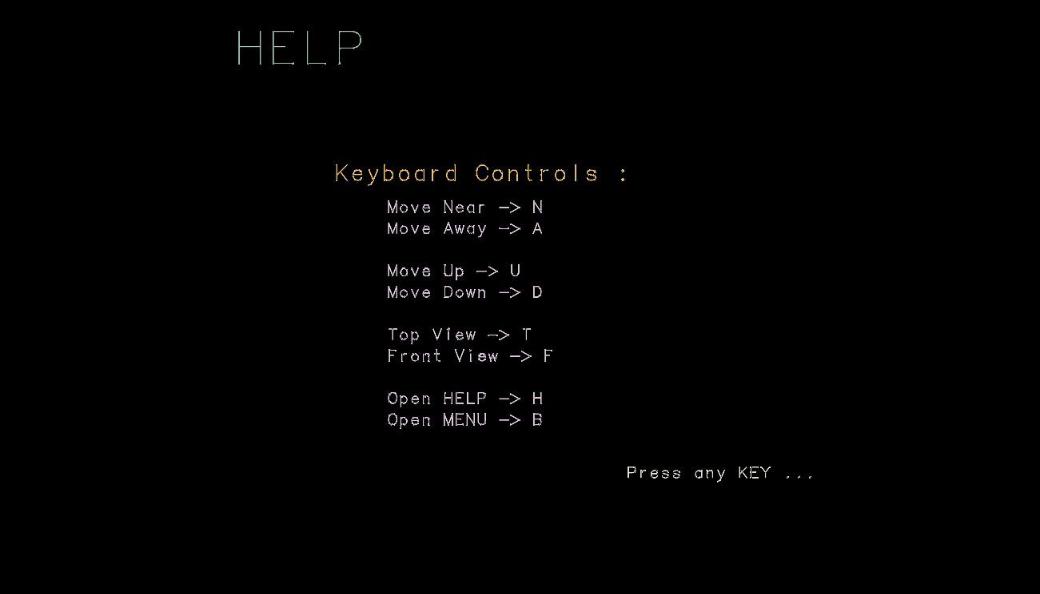


**Fig 4.1:First Window**

Figure 4.1 shows the Beginning Look of the project.

****

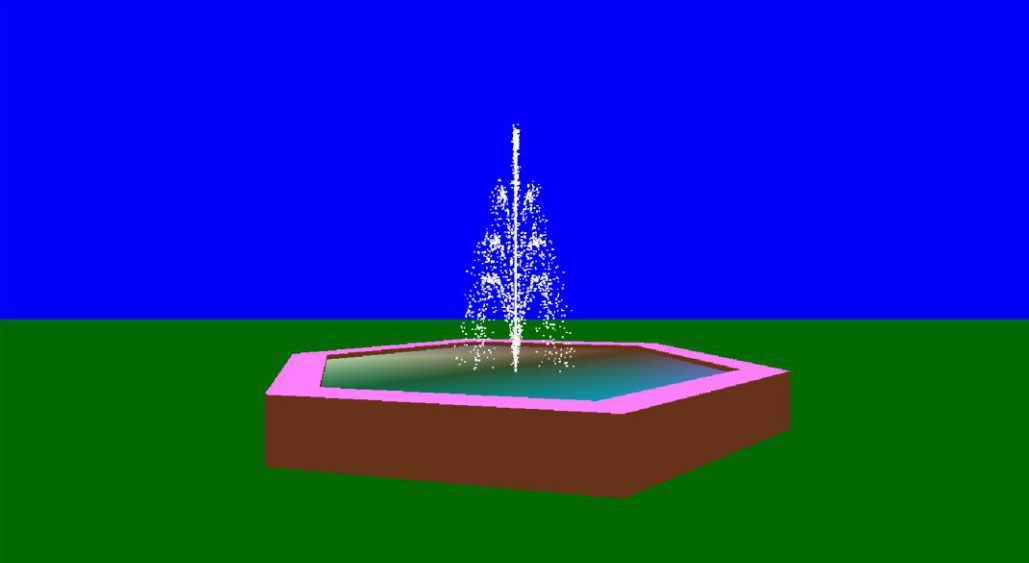
**Fig 4.2 : Menu page**

Figure 4.2 shows the Menu page of the project which contain options to Proceed, Help ,exit and Back.

**Fig 4.3**: **Display of the player Options.**

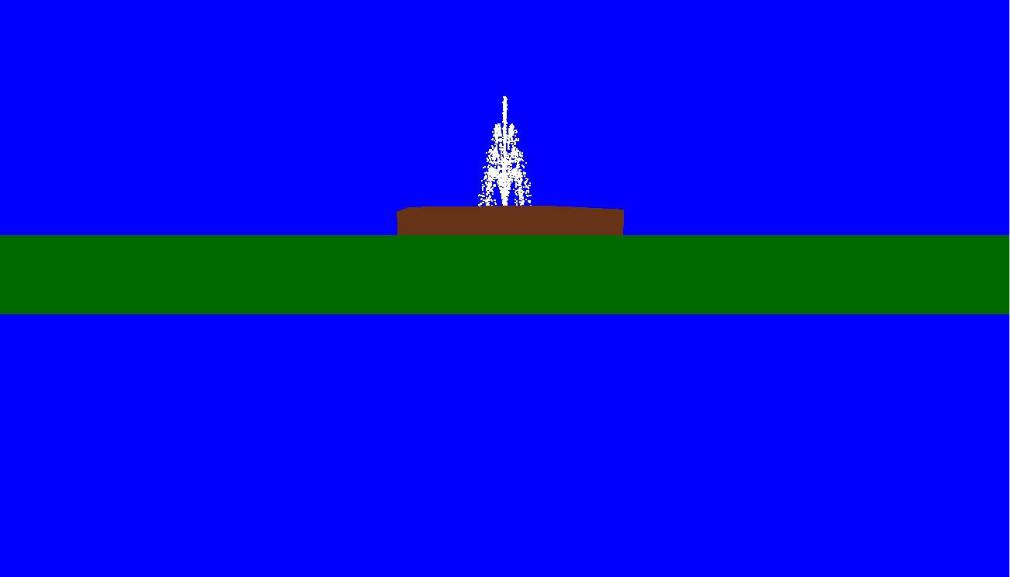
**Fig 4.3: Help window**

Figure 4.3 Shows pops up when you click (2) Help in the Menu page, which helps you to guide the keyboard controls of the project



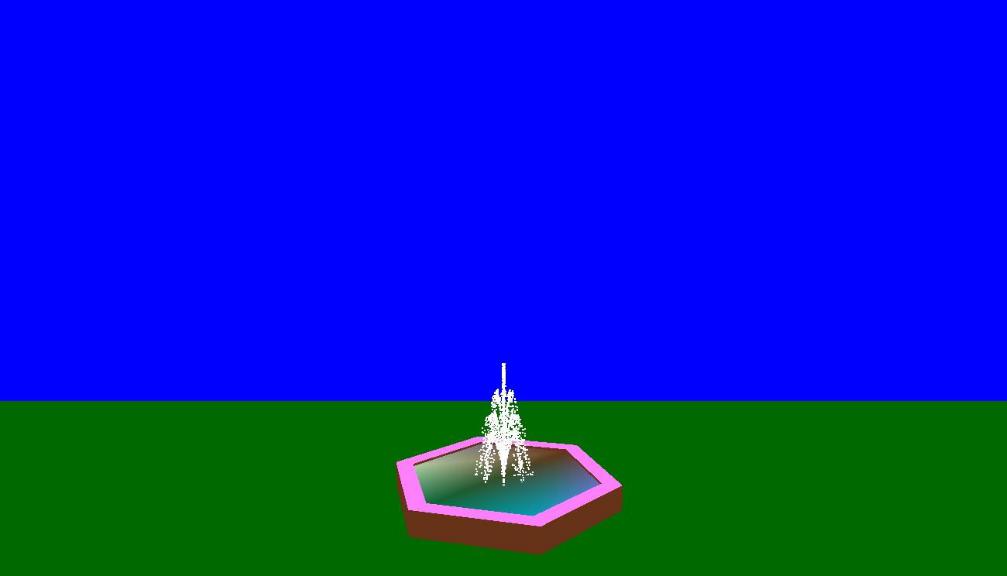
**Fig 4.6: Away output page**

Figure 4.6 This window shows the Distant look of fountain in the project.



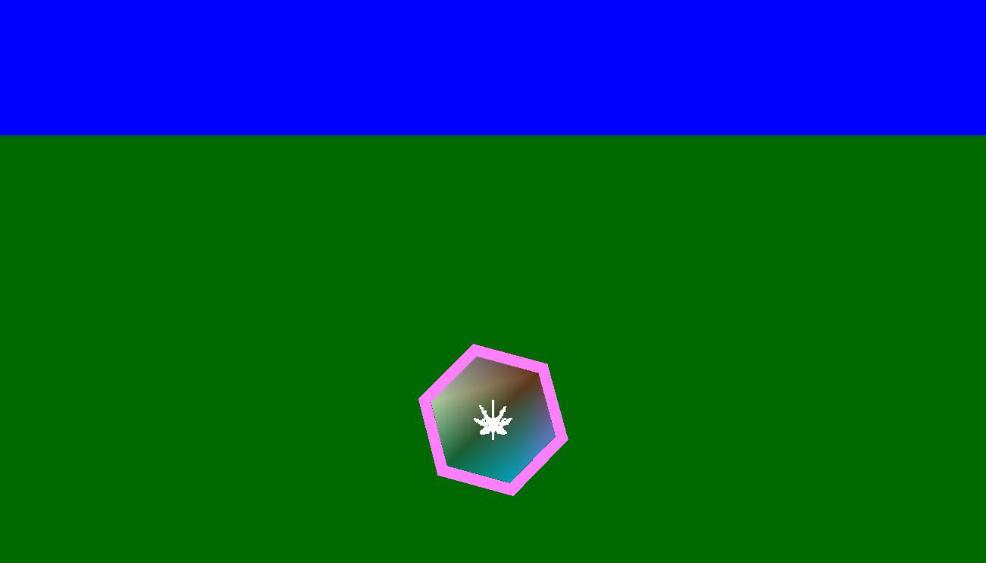
**Fig 4.7: Upside output page**

Figure 4.7 shows the Fountains upside View in the project.



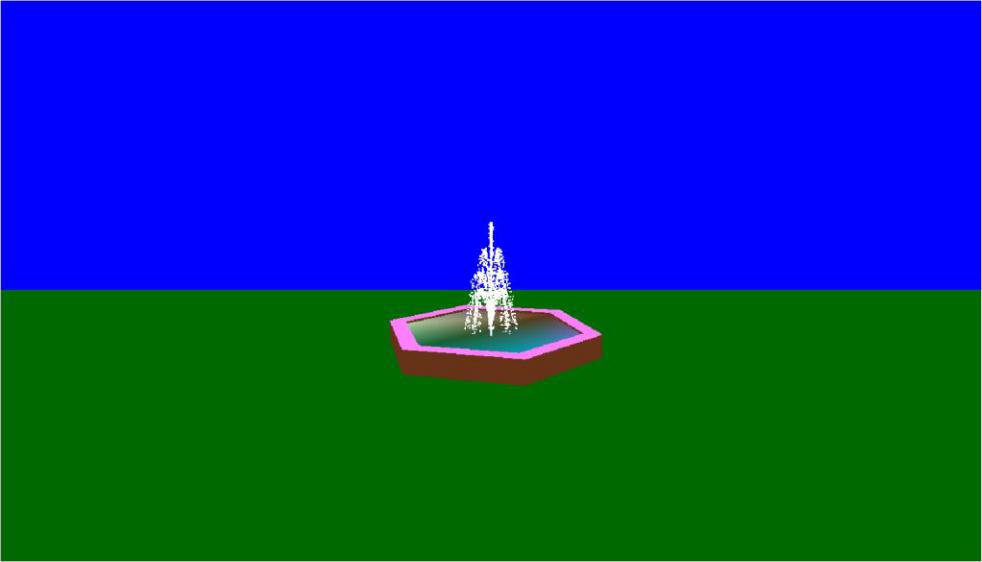
**Fig 4.8: Down View output page**

Figure 4.8 shows the Fountains Down View in the project.



**Fig 4.9 Top view output page**

Figure4.9 shows the Top View of Fountain in project.



**Fig 4.10:Front view page**

Figure 4.10 shows the Front View of the fountain in project.

**CHAPTER 5**

**CONCLUSION**

The OpenGL project developed is a sincere effort towards design and implementation of Computer graphics. The OpenGL package is user friendly and provides an easy interaction for the user. This project is an attempt at understanding the basic functionality of a computer graphics package and mastering it. It was a great learning experience to learn a graphics package and develop a graphics application.

**REFERENCES**

**BOOKS:**

* F.S. Hill Jr.: Computer Graphics Using OpenGL, 2nd Edition, Pearson education, 2001.
* James D. Foley, Andries Van Dam, Steven K. Feiner, John F. Hughes, Computer Graphics, Addison-wesley 1997.

**WEB RESOURCES:**

* [www.google.com](http://www.google.com)
* [www.opengl.org](http://www.opengl.org)
* [www.glProgramming.com](http://www.glProgramming.com)