**IMPLEMENTATION**

**3.1 CODE:**

#include <GL/glut.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

#include <iostream.h>

#define PI 3.14152653597689786

#define RandomFactor 2.0

#define ESCAPE 27

#define TEXTID 3

unsigned int i;

int flag=0,f=2;

int vflag=0;

GLfloat xt=0.0,yt=0.0,zt=0.0;

GLfloat xangle=0.0,yangle=0.0,zangle=0.0;

GLfloat X[3];

GLint ListNum; //The number of the diplay list

GLfloat OuterRadius = 2.4; //reservoir

GLfloat InnerRadius = 2.0;

GLint NumOfVerticesStone = 6; // reservoir shape

GLfloat StoneHeight = 0.5;

GLfloat WaterHeight = 0.45;

struct SVertex{GLfloat x,y,z;};

///////////////////////////////fountain///////////////////////////////////

class CDrop{

private:

GLfloat time;

SVertex ConstantSpeed;

GLfloat AccFactor;

public:

void SetConstantSpeed (SVertex NewSpeed);

void SetAccFactor(GLfloat NewAccFactor);

void SetTime(GLfloat NewTime);

void GetNewPosition(SVertex \* PositionVertex); //increments time, gets the new position

};

void CDrop::SetConstantSpeed(SVertex NewSpeed){

ConstantSpeed = NewSpeed;

}

void CDrop::SetAccFactor (GLfloat NewAccFactor){

AccFactor = NewAccFactor;

}

void CDrop::SetTime(GLfloat NewTime){

time = NewTime;

}

void CDrop::GetNewPosition(SVertex \* PositionVertex){

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;

SVertex \* FountainVertices;

GLint Steps = 4; //a fountain has several steps, each with its own height

GLint RaysPerStep =8;

GLint DropsPerRay = 80;

GLfloat DropsComplete = Steps \* RaysPerStep \* DropsPerRay;

GLfloat AngleOfDeepestStep = 80;

GLfloat AccFactor = 0.011;

// Creating reservoir boundary

void CreateList(void){

SVertex \* Vertices = new SVertex[NumOfVerticesStone\*3]; //allocate mem for req vertices

ListNum = glGenLists(1);

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

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

Vertices[i].y = StoneHeight; //Top

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; //Top

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; //Bottom

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

glNewList(ListNum, GL\_COMPILE);

glBegin(GL\_QUADS); //ground quad:

glColor3ub(0,105,0);

glVertex3f(-OuterRadius\*10.0,0.0,OuterRadius\*10.0);

glVertex3f(-OuterRadius\*10.0,0.0,-OuterRadius\*10.0);

glVertex3f(OuterRadius\*10.0,0.0,-OuterRadius\*10.0);

glVertex3f(OuterRadius\*10.0,0.0,OuterRadius\*10.0);

for (int j = 1; j < 3; j++) { //stone

if (j == 1) glColor3f(1.3,0.5,1.2);

if (j == 2) glColor3f(0.4,0.2,0.1);

for (i = 0; i<NumOfVerticesStone-1; i++){

glVertex3fv(&Vertices[i+NumOfVerticesStone\*j].x);

glVertex3fv(&Vertices[i].x);

glVertex3fv(&Vertices[i+1].x);

glVertex3fv(&Vertices[i+NumOfVerticesStone\*j+1].x); }

glVertex3fv(&Vertices[i+NumOfVerticesStone\*j].x);

glVertex3fv(&Vertices[i].x);

glVertex3fv(&Vertices[0].x);

glVertex3fv(&Vertices[NumOfVerticesStone\*j].x); }

glEnd();

glTranslatef(0.0,WaterHeight - StoneHeight, 0.0); //The "water":

glBegin(GL\_POLYGON);

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

glVertex3fv(&Vertices[i+NumOfVerticesStone].x);

GLint m1,n1,p1;

m1=rand()%255;

n1=rand()%255;

p1=rand()%255;

glColor3ub(m1,n1,p1);

// glColor3f(1.0,1.0,1.0); }

glEnd();

glEndList(); }

GLfloat GetRandomFloat(GLfloat range){

return (GLfloat)rand() / (GLfloat)RAND\_MAX \* range \* RandomFactor; }

void randcolor()

{

GLint a,b,c;

a=rand()%101;

b=rand()%101;

c=rand()%101;

X[0]=(GLfloat)a/100.0;

X[1]=(GLfloat)b/100.0;

X[2]=(GLfloat)c/100.0;

}

void DrawFountain(void) {

if(flag==0)

glColor3f(1,1,1);

else if(flag==1)

glColor3fv(X);

else if(flag==2)

glColor3f(0.0,1.0,0.0);

else

glColor3f(0.0,1.0,1.0);

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

{

FountainDrops[i].GetNewPosition(&FountainVertices[i]);

}

glDrawArrays( GL\_POINTS,0,DropsComplete);

glutPostRedisplay();

}

void help(int id) {

glutPostRedisplay(); }

//key board functions

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

{

if(f==0) //main page

{ switch ( key )

{

case 13:

case '1': f=3; break; //fountain

case 'h':

case '2': f=1; break; //help

case '3': //exit

case '4':

case 'b': f=2; break;

case ESCAPE: exit(0);

glutPostRedisplay();

}

}

else if(f==1) //help page

{ if(key=='b'||key=='B')

f=0;

else

f=3;

glutPostRedisplay();

}

else if(f==2) //cover page

{ f=0;

}

else // fountain page

{

switch ( key )

{

case ESCAPE : exit(0); break;

case 't': case 'T':

vflag=3; //top view

glutPostRedisplay();

break;

case 'f': case 'F':

vflag=33; //top view

glutPostRedisplay();

break;

case 'd': case 'D':

vflag=2; // Move down

glutPostRedisplay();

break;

case 'u': case 'U':

vflag=22; // Move up

glutPostRedisplay();

break;

case 'a': case 'A':

vflag=1; // Move away

glutPostRedisplay();

break;

case 'n': case 'N':

vflag=11; // Move near

glutPostRedisplay();

break;

case 'b': case 'B': //back

f=0;

glutPostRedisplay();

break;

case 'h': case 'H': //help

f=1;

glutPostRedisplay();

break;

default:break; }

}//end of else

}

void Display(void){

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

glLoadIdentity();

glClearColor(0,0,100,1.0);

glTranslatef(0.0,0.0,-6.0);

glTranslatef(0.0,-1.3,0.0);

if(vflag==1) { //far

zt-=0.06; }

glTranslatef(xt,yt,zt);

if(vflag==11) { //near

zt+=0.06; }

glTranslatef(xt,yt,zt);

if(vflag==2){ //down

yt -= 0.05; }

glTranslatef(xt,yt,zt);

if(vflag==22){ //up

yt += 0.05; }

glTranslatef(xt,yt,zt

if(vflag==3){ //angular

if(xangle<=80.0)

xangle += 5.0; }

if(vflag==33){ //angular

if(xangle>=-5)

xangle -= 5.0; }

glColor3f(1.0,0.0,0.0);

glRotatef(xangle,1.0,0.0,0.0);

vflag=0;

glRotatef(45.0,0.0,1.0,0.0);

glPushMatrix();

glCallList(ListNum);

glPopMatrix();

DrawFountain();

glFlush(); //Finish rendering

glutSwapBuffers(); //Swap the buffers ->make the result of rendering visible

}

glutSwapBuffers();

}

void Dis()

{

if(f==0)

menu1();

else if(f==1)

menu2();

else if(f==2)

cover();

else

Display();

}

void Reshape(int x, int y)

{

if (y == 0 || x == 0) return; //Nothing is visible then, so return

//Set a new projection matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(50.0,(GLdouble)x/(GLdouble)y,0.10,20.0);

glMatrixMode(GL\_MODELVIEW);

glViewport(0,0,x,y); //Use the whole window for rendering

glPointSize(GLfloat(x)/600.0); //Adjust point size to window size

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(1024,768);

glutInitWindowPosition(0,0);

glutCreateWindow("Flowing Fountain");

glEnable(GL\_DEPTH\_TEST);

glClearColor(0,0,100,1.0);

glBlendFunc(GL\_SRC\_ALPHA,GL\_ONE\_MINUS\_SRC\_ALPHA);

glEnable(GL\_LINE\_SMOOTH);

glEnable(GL\_BLEND);

glLineWidth(2.0);

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

InitFountain();

CreateList();

glutDisplayFunc(Dis);

glutReshapeFunc(Reshape);

glutKeyboardFunc(NormalKey);

//add the menu and submenus

int sub\_menu=glutCreateMenu(colours);

glutAddMenuEntry("RANDOM",1);

glutAddMenuEntry("GREEN",2);

glutAddMenuEntry("BLUE",3);

int sub\_menu2=glutCreateMenu(flow);

glutAddMenuEntry("LOW",8);

glutAddMenuEntry("MEDIUM",10);

glutAddMenuEntry("HIGH",20);

int sub\_menu3=glutCreateMenu(level);

glutAddMenuEntry("3 LEVELS",3);

glutAddMenuEntry("4 LEVELS",4);

glutAddMenuEntry("5 LEVELS",5);

int sub\_menu4=glutCreateMenu(help);

glutAddMenuEntry("KEYBOARD CONTROLS:",0);

glutAddMenuEntry("Move Near: n",1);

glutAddMenuEntry("Move Away: a",2);

glutAddMenuEntry("Move Down: d",3);

glutAddMenuEntry("Move Up: u",4);

glutAddMenuEntry("Vertical 360: x",5);

glutAddMenuEntry("EXIT",6);

glutCreateMenu(CMain);

glutAddSubMenu("Colors",sub\_menu);

glutAddSubMenu("Help",sub\_menu4);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);//attached to right click

glutIdleFunc(Dis); //sets the global idle callback

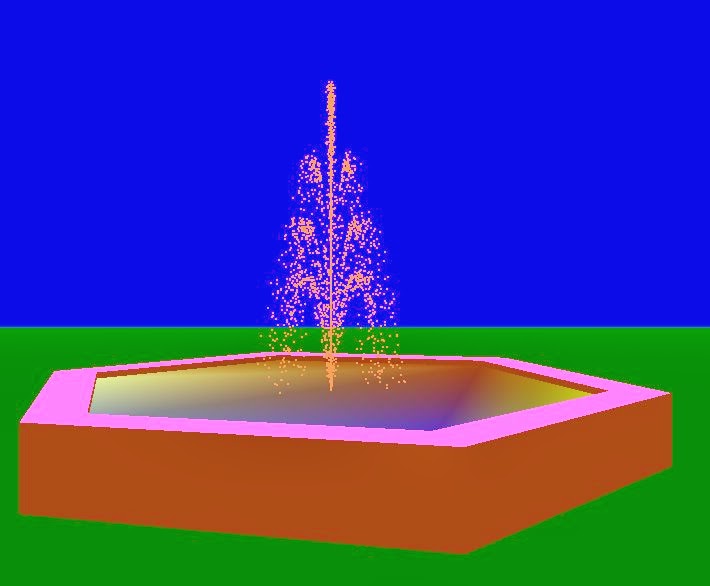
glutMainLoop();

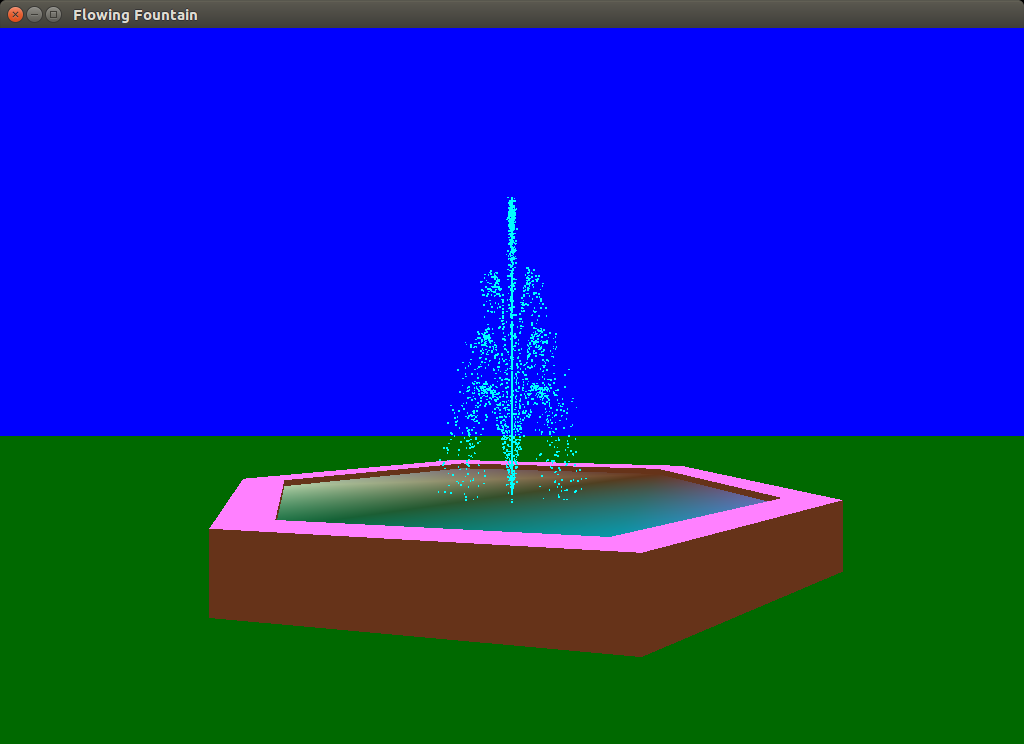
return 0;

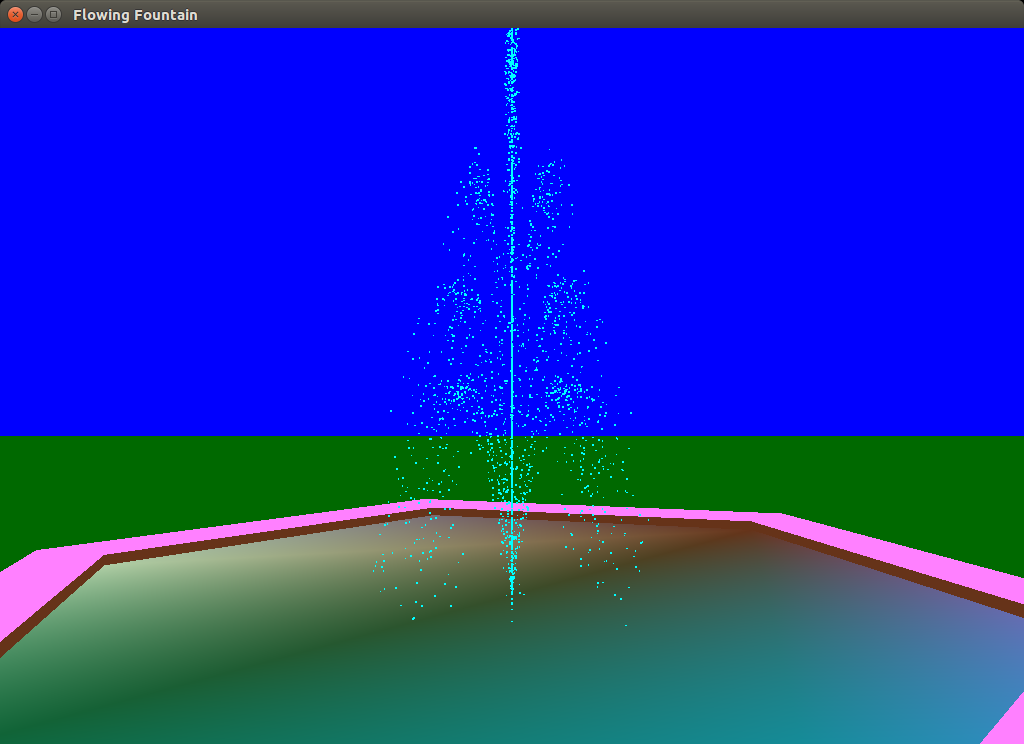
}

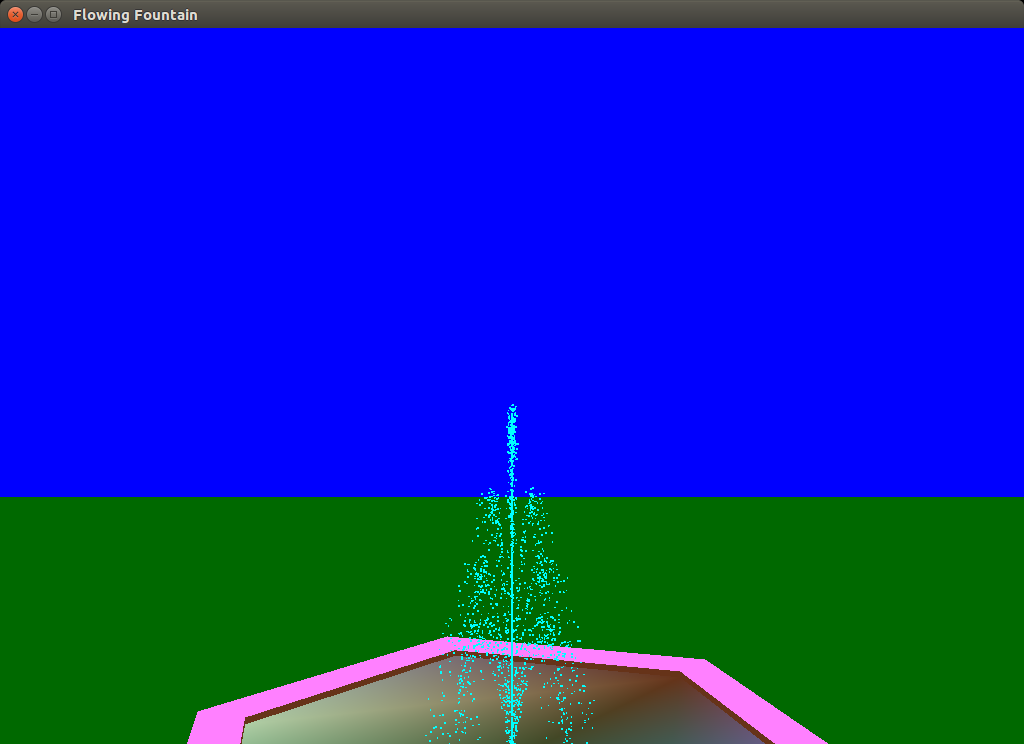
**CHAPTER 4**

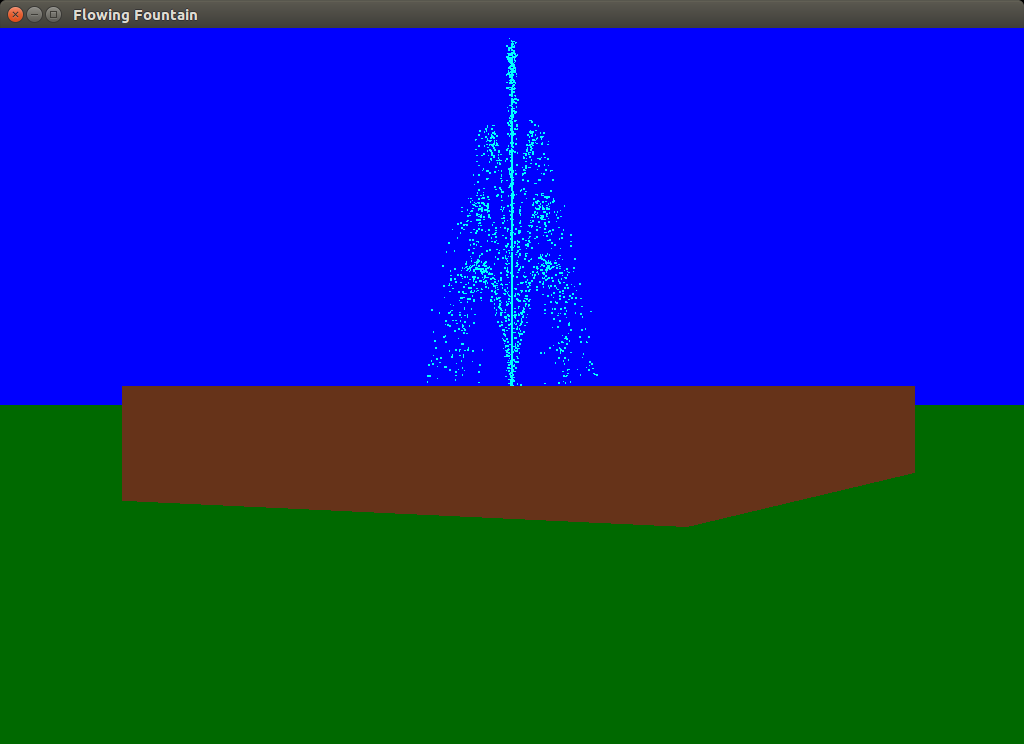
**RESULTS**

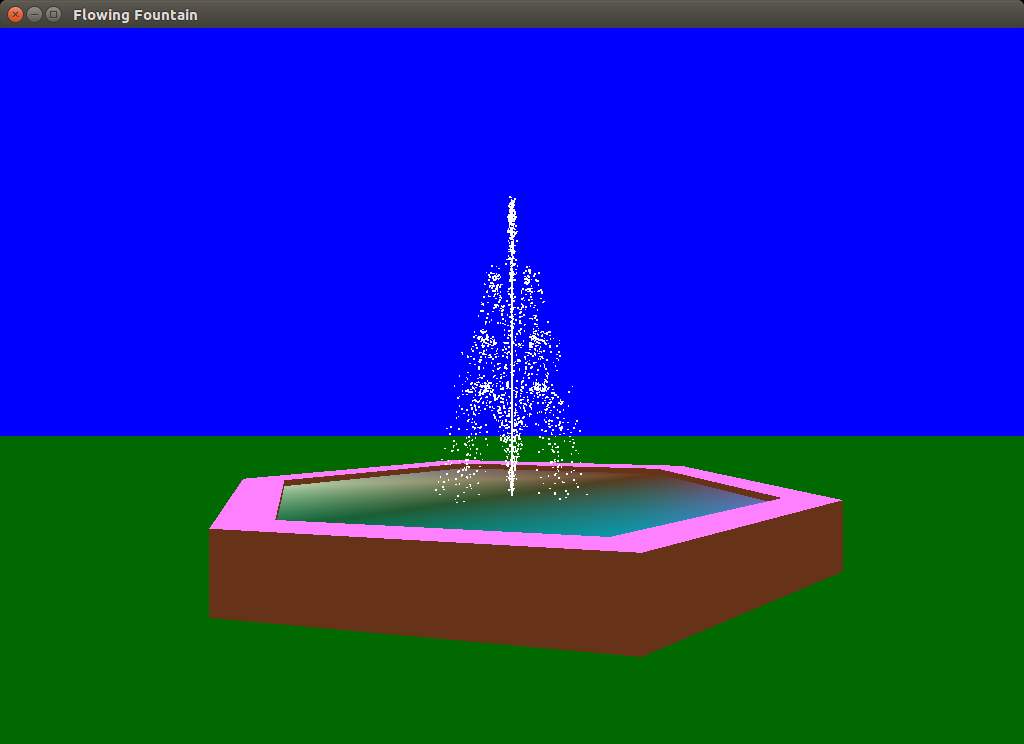


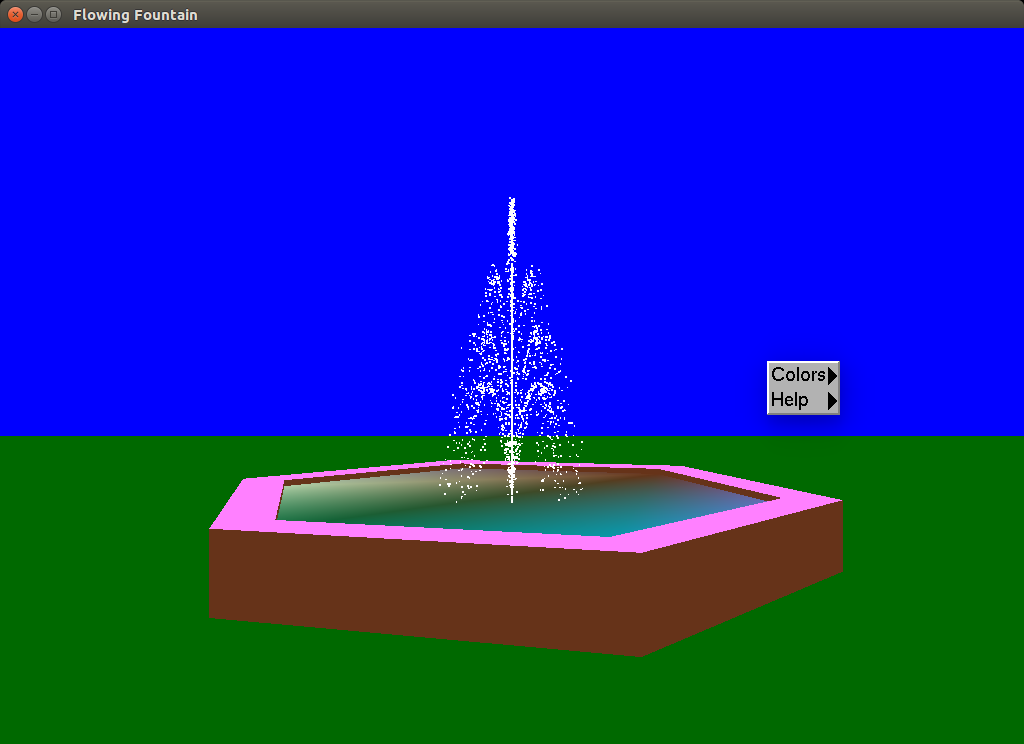














**CHAPTER-5**

**CONCLUSION**

We have shown computer graphics is a method of image formation that should be related to classical method of image formation. We have also introduced OpenGL API and the applied basic concepts. The historical development of graphics APIs and graphical method illustrates the starting in 3 dimensions.

The interactive aspects make the field of computer graphics exciting and fun. Although our API, OpenGL, is independent of any operating or window system, we recognize that any program must have at least minimal interaction with the rest of the computer system. We handled simple interactions by using a simple tool kit, GLUT, whose API provides the necessary additional functionality, without being dependent on a particular operating or window system. From the application programmer’s perspective, various characteristics of interactive graphics are shared by most systems. We see the graphic part of the system as a server, consisting of a raster display, a keyboard, and a pointing device. Thus interactive computer graphics is a powerful tool with unlimited applications.

Hence, we conclude that OpenGL remains the standard programmer’s interface both for writing application programs and developing high-level products of multiplatform applications. OpenGL supports application ranging from large scientific visualizations to cell phone games.

**CHAPTER-6**

**BIBLIOGRAPHY**

1. Web references

* <http://www.opengl.org/sdk/docs/man/xhtml/>
* <http://www.opengl.org/resources/libraries/glut/>
* https://en.m.wikibooks.org/OpenGL

1. Book references

* **Interactive Computer Graphics- A Top-Down Approach with OpenGL**-Edward Angel, 5th Edition, Addison-Wesley, 2008.
* **Computer Graphics Using OpenGL**– F.S. Hill,Jr.  2nd Edition, Pearson Education, 2001.