```
#include <glut.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <iostream>
#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 the required
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);
//stone:
for (int j = 1; j < 3; j++)
{
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();
//The "water":
glTranslatef(0.0,WaterHeight - StoneHeight, 0.0);
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 InitFountain(void)
{
//This function needn't be and isn't speed optimized
FountainDrops = new CDrop [ (int)DropsComplete ];
FountainVertices = new SVertex [ (int)DropsComplete ];
SVertex NewSpeed;
GLfloat DropAccFactor; //different from AccFactor because of the random change
GLfloat TimeNeeded;
GLfloat StepAngle; //Angle, which the ray gets out of the fountain with
GLfloat RayAngle; //Angle you see when you look down on the fountain
GLint i,j,k;
for (k = 0; k <Steps; k++)
{
for (j = 0; j < RaysPerStep; j++)
{
for (i = 0; i < DropsPerRay; i++)
{
DropAccFactor = AccFactor + GetRandomFloat(0.0005);
StepAngle = AngleOfDeepestStep + (90.0-AngleOfDeepestStep)
* GLfloat(k) / (Steps-1) + GetRandomFloat(0.2+0.8*(Steps-k-1)/(Steps-1));
//This is the speed caused by the step:
NewSpeed.x = cos ( StepAngle * PI / 180.0) * (0.2+0.04*k);
NewSpeed.y = sin ( StepAngle * PI / 180.0) * (0.2+0.04*k);
//This is the speed caused by the ray:
RayAngle = (GLfloat)j / (GLfloat)RaysPerStep * 360.0;
//for the next computations "NewSpeed.x" is the radius. Care! Dont swap the two
//lines, because the second one changes NewSpeed.x!
NewSpeed.z = NewSpeed.x * sin ( RayAngle * PI /180.0);
NewSpeed.x = NewSpeed.x * cos ( RayAngle * PI /180.0);
//Calculate how many steps are required, that a drop comes out and falls down again
TimeNeeded =NewSpeed.y/ DropAccFactor;
FountainDrops[i+j*DropsPerRay+k*DropsPerRay*RaysPerStep].SetConstantSpeed ( NewSpeed
);
FountainDrops[i+j*DropsPerRay+k*DropsPerRay*RaysPerStep].SetAccFactor
(DropAccFactor);
FountainDrops[i+j*DropsPerRay+k*DropsPerRay*RaysPerStep].SetTime(TimeNeeded * i
/DropsPerRay);
}
}
}
```

```
//Tell OGL that we'll use the vertex array function
glEnableClientState(GL_VERTEX_ARRAY);

//Pass the data position
glVertexPointer( 3, //x,y,z-components
GL_FLOAT, //data type of SVertex
0, //the vertices are tightly packed
FountainVertices);
}
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 colours(int id)
{
flag=id;
if(flag==1)
randcolor();
glutPostRedisplay();
}
void flow(int id)
{
RaysPerStep=id;
glutPostRedisplay();
}
void level(int id)
{
Steps=id;
glutPostRedisplay();
}
void help(int id)
{
glutPostRedisplay();
}
void CMain(int id)
{
}
//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 // funtain 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
}
// functrion to display text
void DrawTextXY(double x,double y,double z,double scale,char *s)
{
int i;
glPushMatrix();
glTranslatef(x,y,z);
glScalef(scale,scale,scale);
for (i=0;i < strlen(s);i++)
glutStrokeCharacter(GLUT_STROKE_MONO_ROMAN,s[i]);
// glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,s[i]);
glPopMatrix();
}
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();
glFlush(); //Finish rendering
glutSwapBuffers(); //Swap the buffers ->make the result of rendering visible
}
//display menu
void menu1()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
glClearColor(0,0,0,0.0);
glTranslatef(0.0,0.0,-6.0);
glTranslatef(0.0,-1.3,0.0);
glColor3f(1.00,0.20,0.10);
glLoadName(TEXTID);
DrawTextXY(-2.7,3.5,0.0,0.003," FOUNTAIN ");
glColor3f(0.6,0.8,0.7);
DrawTextXY(-1.25,2.4,0.0,0.0014," MENU ");
glColor3f(1.0,0.8,0.4);
DrawTextXY(-1.25,2.1,0.0,0.001," 1 : PROCEED ");
DrawTextXY(-1.25,1.9,0.0,0.001," 2 : HELP ");
DrawTextXY(-1.25,1.7,0.0,0.001," 3 : EXIT ");
DrawTextXY(-1.25,1.5,0.0,0.001," 4 : BACK");
glFlush(); //Finish rendering
glutSwapBuffers();
}
void menu2()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
glClearColor(0,0,0,1.0);
```

```
glTranslatef(0.0,0.0,-6.0);
glTranslatef(0.0,-1.3,0.0);
glColor3f(0.6,0.8,0.7);
DrawTextXY(-2.7,3.5,0.0,0.003," HELP ");
glColor3f(1.0,0.8,0.4);
DrawTextXY(-1.75,2.4,0.0,0.0014," Keyboard Controls : ");
glColor3f(0.9,0.8,0.9);
DrawTextXY(-1.25,2.1,0.0,0.001," Move Near -> N ");
DrawTextXY(-1.25,1.9,0.0,0.001," Move Away -> A ");
DrawTextXY(-1.25,1.5,0.0,0.001," Move Up -> U ");
DrawTextXY(-1.25,1.3,0.0,0.001," Move Down -> D ");
DrawTextXY(-1.25,0.9,0.0,0.001," Top View -> T ");
DrawTextXY(-1.25,0.7,0.0,0.001," Front View -> F ");
DrawTextXY(-1.25,0.3,0.0,0.001," Open HELP -> H ");
DrawTextXY(-1.25,0.1,0.0,0.001," Open MENU -> B ");
glColor3f(0.9,0.9,0.8);
DrawTextXY(1,-0.4,0.0,0.001," Press any KEY ... ");
glFlush(); //Finish rendering
glutSwapBuffers();
}
void cover()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
glClearColor(0,0,0,0.0);
glTranslatef(0.0,0.0,-6.0);
glTranslatef(0.0,-1.3,0.0);
glColor3f(1.00,0.20,0.10);
glLoadName(TEXTID);
DrawTextXY(-1.7,3.5,0.0,0.001," GRAPHICAL IMPLEMENTATION OF ");
glColor3f(0.6,0.8,0.7);
```

```
DrawTextXY(-1.75,3,0.0,0.0014," FLOWING FOUNTAIN ");
glColor3f(0.7,0.6,0.1);
DrawTextXY(-3.25,1.5,0.0,0.0007," Submitted by :- ");
glColor3f(1.0,0.5,0.0);
DrawTextXY(-2.5,1.2,0.0,0.001," First Name ");
DrawTextXY(1,1.2,0.0,0.001," Second Name ");
glColor3f(0.7,0.8,0.6);
DrawTextXY(-2.5,0.95,0.0,0.001," (USN 1) ");
DrawTextXY(1,0.95,0.0,0.001," (USN 2) ");
glColor3f(0.7,0.6,0.1);
DrawTextXY(-1.25,0,0.0,0.0007," Under the guidance of : ");
glColor3f(1.0,0.8,0.4);
DrawTextXY(-1.25,-.2,0.0,0.001," Name of Guide");
DrawTextXY(-1,-.5,0.0,0.0007," Designation ");
DrawTextXY(-1,-.7,0.0,0.001," College Name");
glColor3f(0.3,0.3,0.3);
DrawTextXY(-1,-1,0.0,0.0008," Press any key... ");
glFlush(); //Finish rendering
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
//Adjust point size to window size
glPointSize(GLfloat(x)/600.0);
}
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;
}
```