Program 9

9. Write a program to model a car like figure using display lists and move a car from one end of the screen to other end. User is able to control the speed with mouse.

```
#include <GL/glut.h> // Header File For The GLUT Library

#include <GL/gl.h> // Header File For The OpenGL32 Library

#include <GL/glu.h> // Header File For The GLu32 Library

//#include <unistd.h> // Header File For sleeping.

/* ASCII code for the escape key. */

#define ESCAPE 27

/* The number of our GLUT window */

int window;

/* rotation angle for the triangle. */

float rtri = 0.0f;

/* rotation angle for the quadrilateral. */

float rquad = 0.0f;

/* A general OpenGL initialization function. Sets all of the initial parameters. */

// We call this right after our OpenGL window is created.

void InitGL(int Width, int Height)

{

 // This Will Clear The Background Color To Black

 glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

 glClearDepth(1.0); // Enables Clearing Of The Depth Buffer

 glDepthFunc(GL_LESS); // The Type Of Depth Test To Do

 glEnable(GL_DEPTH_TEST); // Enables Depth Testing

 glShadeModel(GL_SMOOTH); // Enables Smooth Color Shading

 glMatrixMode(GL_PROJECTION);
```

```
 glLoadIdentity(); // Reset The Projection Matrix
 gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,100.0f);
 glMatrixMode(GL_MODELVIEW);
}
/* The function called when our window is resized (which shouldn't happen, because we're fullscreen)
*/
void ReSizeGLScene(int Width, int Height)
{
 if (Height==0) // Prevent A Divide By Zero If The Window Is Too Small
 Height=1;
 glViewport(0, 0, Width, Height); // Reset The Current Viewport And Perspective Transformation
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,100.0f);
 glMatrixMode(GL_MODELVIEW);
}
float ballX = -0.5f;
float ballY = 0.0f;
float ballZ = 0.0f;
void drawBall(void) {
 glColor3f(0.0, 1.0, 0.0); //set ball colour
 glTranslatef(ballX,ballY,ballZ); //moving it toward the screen a bit on creation
 //glRotatef(ballX,ballX,ballY,ballZ);
 glutSolidSphere (0.3, 20, 20); //create ball.
 glTranslatef(ballX+1.5,ballY,ballZ); //moving it toward the screen a bit on creation
 glutSolidSphere (0.3, 20, 20); //
```

```
 }
/* The main drawing function. */
void DrawGLScene()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear The Screen And The Depth Buffer
 glLoadIdentity(); // Reset The View
 glTranslatef(rtri,0.0f,-6.0f); // Move Left 1.5 Units And Into The Screen 6.0

 //glRotatef(rtri,1.0f,0.0f,0.0f); // Rotate The Triangle On The Y axis
 // draw a triangle (in smooth coloring mode)
 glBegin(GL_POLYGON); // start drawing a polygon
 glColor3f(1.0f,0.0f,0.0f); // Set The Color To Red
 glVertex3f(-1.0f, 1.0f, 0.0f); // Top left
 glVertex3f(0.4f, 1.0f, 0.0f);

 glVertex3f(1.0f, 0.4f, 0.0f);

 glColor3f(0.0f,1.0f,0.0f); // Set The Color To Green
 glVertex3f( 1.0f,0.0f, 0.0f); // Bottom Right
 glColor3f(0.0f,0.0f,1.0f); // Set The Color To Blue
 glVertex3f(-1.0f,0.0f, 0.0f);// Bottom Left
 //glVertex3f();
 glEnd(); // we're done with the polygon (smooth color interpolation)
 drawBall();
 rtri+=0.005f; // Increase The Rotation Variable For The Triangle
```

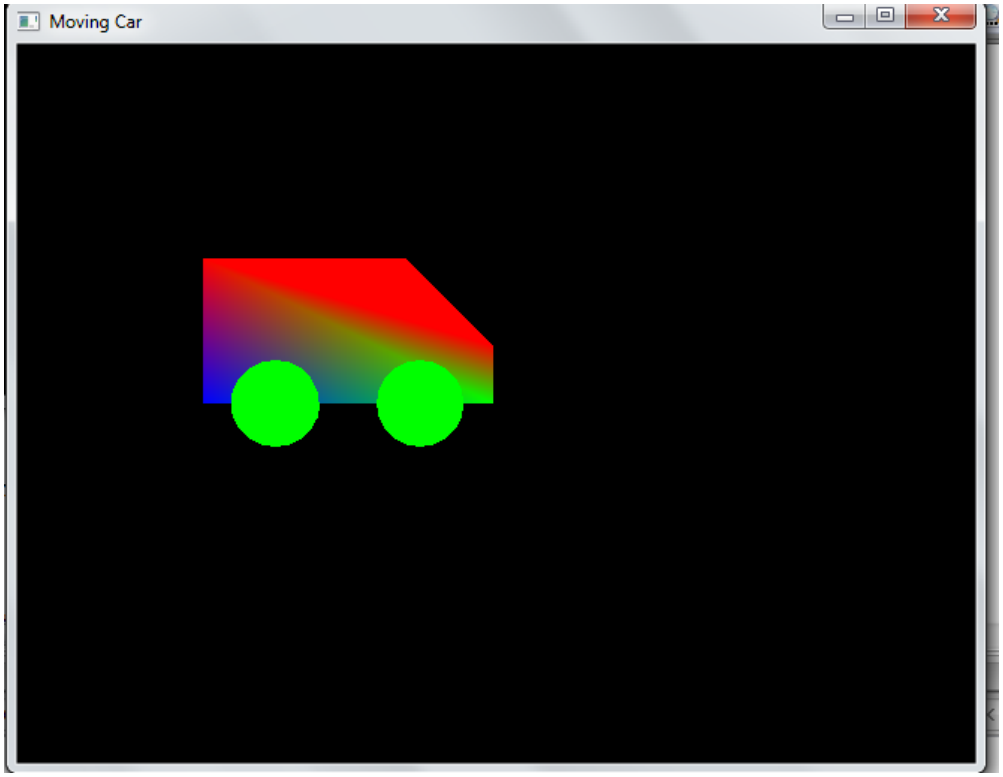
```
 if(rtri>2)
 rtri=-2.0f;
 rquad-=15.0f; // Decrease The Rotation Variable For The Quad
  // swap the buffers to display, since double buffering is used.
 glutSwapBuffers();
}
/* The function called whenever a key is pressed. */
void keyPressed(unsigned char key, int x, int y)
{
 /* sleep to avoid thrashing this procedure */
 // usleep(100);
 /* If escape is pressed, kill everything. */
 if (key == ESCAPE)
 {
 /* shut down our window */
 glutDestroyWindow(window);

 /* exit the program...normal termination. */
 exit(0);
 }
}
int main(int argc, char **argv)
{
 glutInit(&argc, argv);
 glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH);
```

```
/* get a 640 x 480 window */
glutInitWindowSize(640, 480);
/* the window starts at the upper left corner of the screen */
glutInitWindowPosition(0, 0);
/* Open a window */
window = glutCreateWindow("Moving Car");
/* Register the function to do all our OpenGL drawing. */
glutDisplayFunc(&DrawGLScene);
/* Go fullscreen. This is as soon as possible. */
//glutFullScreen();
/* Even if there are no events, redraw our gl scene. */
glutIdleFunc(&DrawGLScene);
/* Register the function called when our window is resized. */
glutReshapeFunc(&ReSizeGLScene);
/* Register the function called when the keyboard is pressed. */
glutKeyboardFunc(&keyPressed);
/* Initialize our window. */
 InitGL(640, 480);

/* Start Event Processing Engine */
glutMainLoop();
return 1;
}
```

OUTPUT:



Program 10

10.Write a program to create a color cube and spin it using OpenGL transformations.

```
#include <GL/glut.h>

GLfloat vertices[8][3] = { {-1.0,-1.0,1.0},{1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0},

 {-1.0,-1.0,-1.0}, {1.0,-1.0,-1.0}, {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}

 };

GLfloat colors[8][3] = { {0.0,0.0,1.0}, {1.0,0.0,1.0}, {1.0,1.0,1.0}, {0.0,1.0,1.0},

{0.0,0.0,0.0},{1.0,0.0,0.0}, {1.0,1.0,0.0}, {0.0,1.0,0.0}

};

GLfloat theta[] = {0.0,0.0,0.0};

GLint axis = 2;

GLdouble viewer[]= {0.0, 0.0, 5.0}; /* initial viewer location */

void polygon(int a, int b, int c , int d)

{

glBegin(GL_POLYGON);

glColor3fv(colors[a]);

glVertex3fv(vertices[a]);

glColor3fv(colors[b]);

glVertex3fv(vertices[b]);

glColor3fv(colors[c]);

glVertex3fv(vertices[c]);

glColor3fv(colors[d]);

glVertex3fv(vertices[d]);

glEnd();

}
```

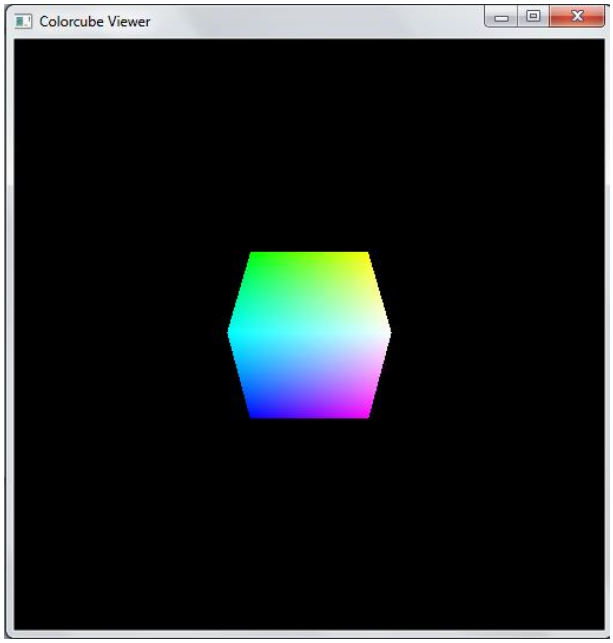
```
void colorcube()
{
polygon(0,3,2,1); // front face – counter clockwise
polygon(4,5,6,7); // back face – clockwise
polygon(2,3,7,6); // front face – counter clockwise
polygon(1,5,4,0); // back face – clockwise
polygon(1,2,6,5); // front face – counter clockwise
polygon(0,4,7,3); // back face – clockwise
}
void display(void)
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
// Update viewer position in modelview matrix
glLoadIdentity();
gluLookAt(viewer[0],viewer[1],viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
glRotatef(theta[0], 1.0, 0.0, 0.0);
glRotatef(theta[1], 0.0, 1.0, 0.0);
glRotatef(theta[2], 0.0, 0.0, 1.0);
colorcube();
glFlush();
glutSwapBuffers();
}
void mouse(int btn, int state, int x, int y)
{
if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
```

```
axis = 0;
if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN)
axis = 1;
if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
axis = 2;
theta[axis]+= 2.0;
if( theta[axis] > 360.0 ) theta[axis]-= 360.0;
display();
}
void keys(unsigned char key, int x, int y)
{
if(key == 'x') viewer[0]-= 1.0;
 if(key == 'X') viewer[0]+= 1.0;
 if(key == 'y') viewer[1]-= 1.0;
 if(key == 'Y') viewer[1]+= 1.0;
 if(key == 'z') viewer[2]-= 1.0;
 if(key == 'Z') viewer[2]+= 1.0;
 display();
}
void myReshape(int w, int h)
{
glViewport(0, 0, w, h);
/* Use a perspective view */
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
```

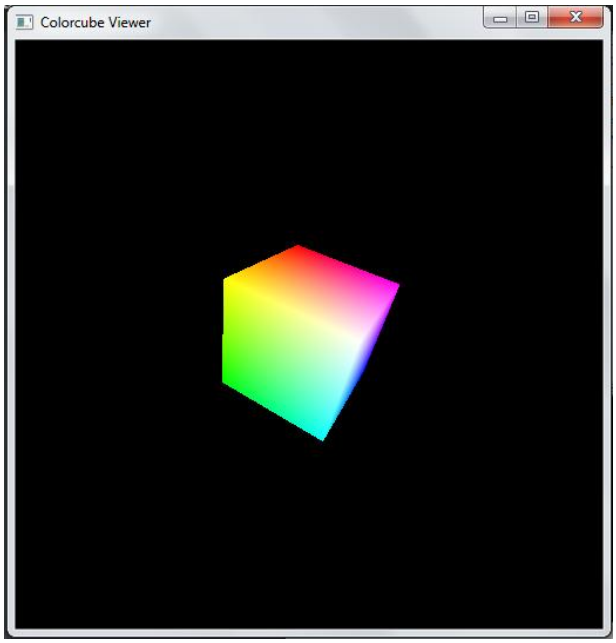
```
if(w<=h)
glFrustum(-2.0, 2.0, -2.0 *(GLfloat) h/(GLfloat) w,2.0*(GLfloat) h/(GLfloat) w, 2.0, 20.0);
else
glFrustum(-2.0, 2.0, -2.0 *(GLfloat)w/(GLfloat) h,2.0* (GLfloat) w / (GLfloat) h, 2.0, 20.0);
glMatrixMode(GL_MODELVIEW);
}
int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
glutInitWindowSize(500, 500);
glutCreateWindow("Colorcube Viewer");
glutReshapeFunc(myReshape);
glutDisplayFunc(display);
glutMouseFunc(mouse);
glutKeyboardFunc(keys);
glEnable(GL_DEPTH_TEST);
glutMainLoop();
}
```

Output:

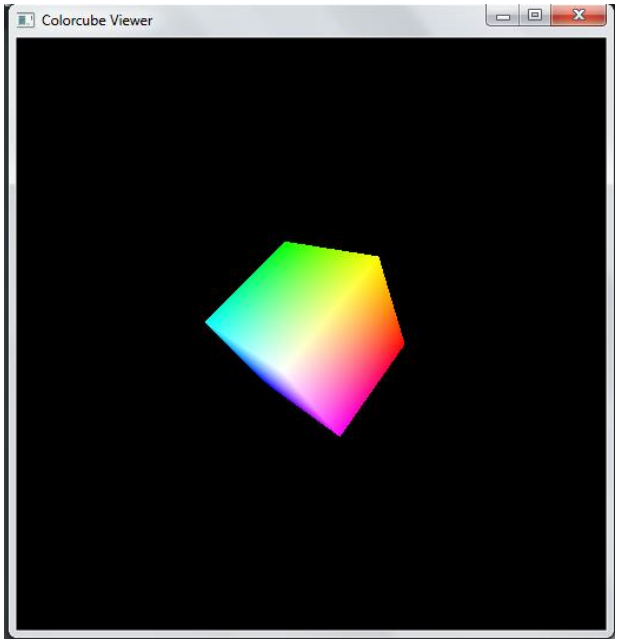
Rotation through X axis



Perspective view with Key ‘x’



Perspective view with Key ‘y’



Program 11

11. Create a menu with three entries named curves, colors and quit. The entry curves has a sub menu which has four entries namely Limacon, Cardiod, Three-Leaf, and Spiral. The color menu has sub menu with all eight colors of RGB color model. Write program to create the above hierarchical menu and attach appropriate services to each menu entries with mouse buttons.

```
#include<gl/glut.h>

#include<math.h>

#include<stdio.h>

struct screenPt {

int x;

int y;

};

typedef enum { limacon = 1, cardioid = 2, threeLeaf = 3, spiral = 4 } curveName;

int w = 600, h = 500;

int curve = 1;

int red = 0, green = 0, blue = 0;

void myinit(void) {

glClearColor(1.0, 1.0, 1.0, 1.0);

glMatrixMode(GL_PROJECTION);

gluOrtho2D(0.0, 200.0, 0.0, 150.0);

}

void lineSegment(screenPt p1, screenPt p2) {

glBegin(GL_LINES);

glVertex2i(p1.x, p1.y);

glVertex2i(p2.x, p2.y);

glEnd();

glFlush();
```

```
}
void drawCurve(int curveNum) {
const double twoPi = 6.283185;
const int a = 175, b = 60;
float r, theta, dtheta = 1.0 / float(a);
int x0 = 200, y0 = 250;
screenPt curvePt[2];
curve = curveNum;
glColor3f(red, green, blue);
curvePt[0].x = x0;
curvePt[0].y = y0;
glClear(GL_COLOR_BUFFER_BIT);
switch (curveNum) {
case limacon: curvePt[0].x += a + b; break;
case cardioid: curvePt[0].x += a + a; break;
case threeLeaf: curvePt[0].x += a; break;
case spiral: break;
default: break;
}
theta = dtheta;
while (theta < twoPi) {
switch (curveNum) {
case limacon: r = a * cos(theta) + b; break;
case cardioid: r = a * (1 + cos(theta)); break;
case threeLeaf: r = a * cos(3 * theta); break;
```

```
case spiral: r = (a / 4.0) * theta; break;
default: break;
}
curvePt[1].x = x0 + r * cos(theta);
curvePt[1].y = y0 + r * sin(theta);
lineSegment(curvePt[0], curvePt[1]);
curvePt[0].x = curvePt[1].x;
curvePt[0].y = curvePt[1].y;
theta += dtheta;
}
}
void colorMenu(int id) {
switch (id) {
case 0:
break;
case 1:
red = 0;
green = 0;
blue = 1;
break;
case 2:
red = 0;
green = 1;
blue = 0;
break;
```

```
case 4:
red = 1;
green = 0;
blue = 0;
break;
case 3:
red = 0;
green = 1;
blue = 1;
break;
case 5:
red = 1;
green = 0;
blue = 1;
break;
case 6:
red = 1;
green = 1;
blue = 0;
break;
case 7:
red = 1;
green = 1;
blue = 1;
break;
```

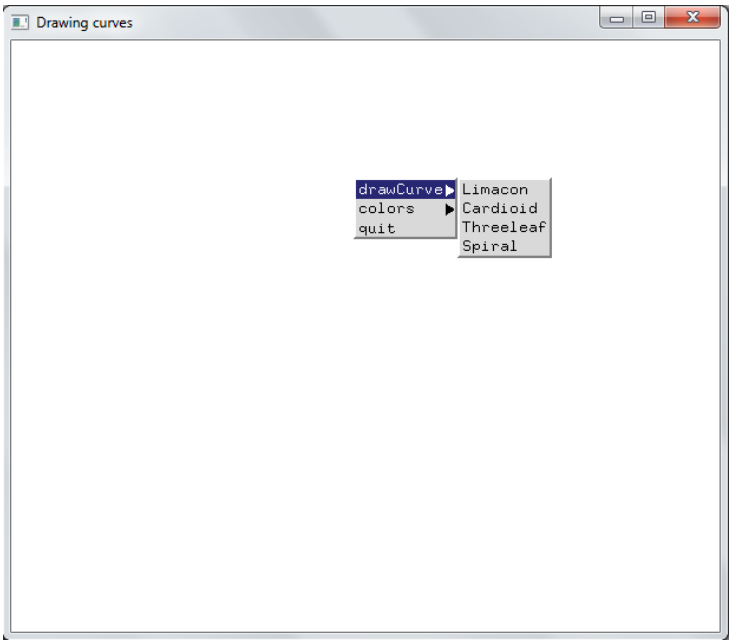
```
default:
break;
}
drawCurve(curve);
}
void main_menu(int id) {
switch (id) {
case 3: exit(0);
default: break;
}
}
void mydisplay() {
}
void myreshape(int nw, int nh) {
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0, (double)nw, 0.0, (double)nh);
glClear(GL_COLOR_BUFFER_BIT);
}
int main(int argc, char** argv) {
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(w, h);
glutInitWindowPosition(100, 100);
glutCreateWindow("Drawing curves");
```

```
int curveId = glutCreateMenu(drawCurve);
glutAddMenuEntry("Limacon", 1);
glutAddMenuEntry("Cardioid", 2);
glutAddMenuEntry("Threeleaf", 3);
glutAddMenuEntry("Spiral", 4);
glutAttachMenu(GLUT_LEFT_BUTTON);
int colorId = glutCreateMenu(colorMenu);
glutAddMenuEntry("Red", 4);
glutAddMenuEntry("Green", 2);
glutAddMenuEntry("Blue", 1);
glutAddMenuEntry("Black", 0);
glutAddMenuEntry("Yellow", 6);
glutAddMenuEntry("Cyan", 3);
glutAddMenuEntry("Magenta", 5);
glutAddMenuEntry("white", 7);
glutAttachMenu(GLUT_LEFT_BUTTON);
glutCreateMenu(main_menu);
glutAddSubMenu("drawCurve", curveId);
glutAddSubMenu("colors", colorId);
glutAddMenuEntry("quit", 3);
glutAttachMenu(GLUT_LEFT_BUTTON);
myinit();
glutDisplayFunc(mydisplay);
glutReshapeFunc(myreshape);
glutMainLoop();
```

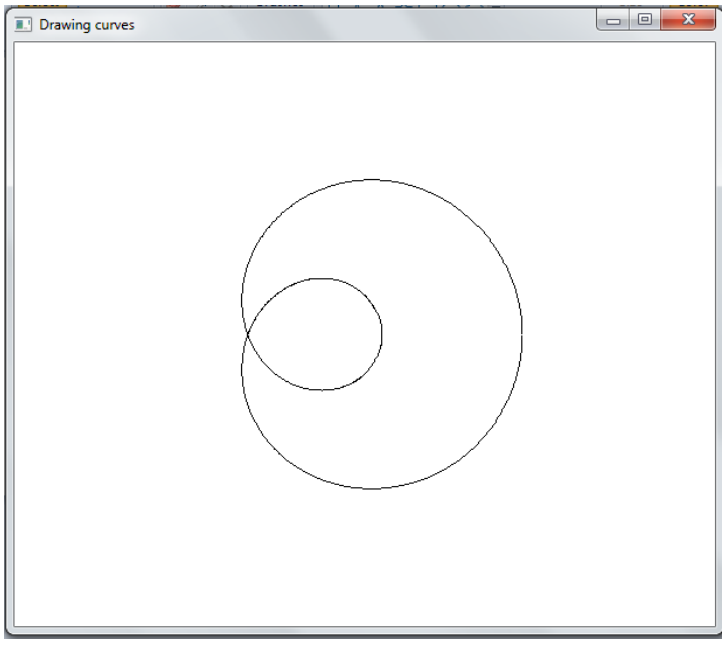
}

Output:

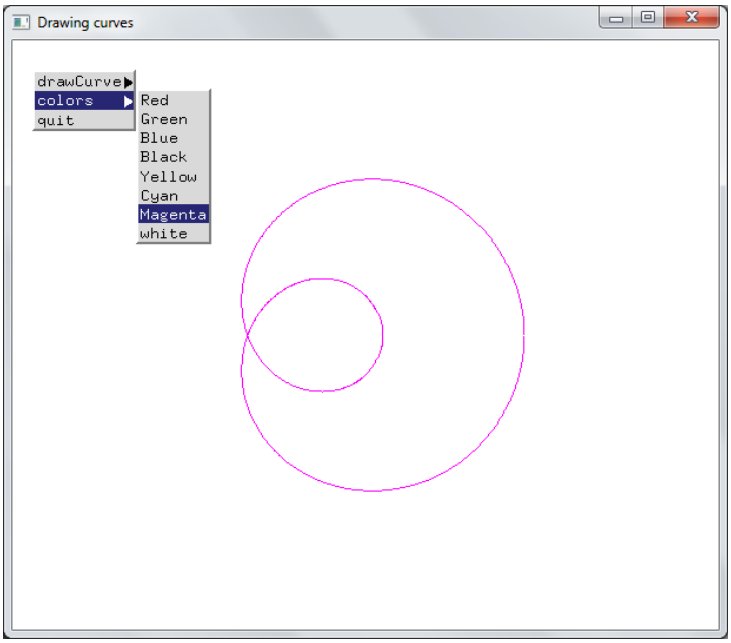
- Click left button show and select the menu.



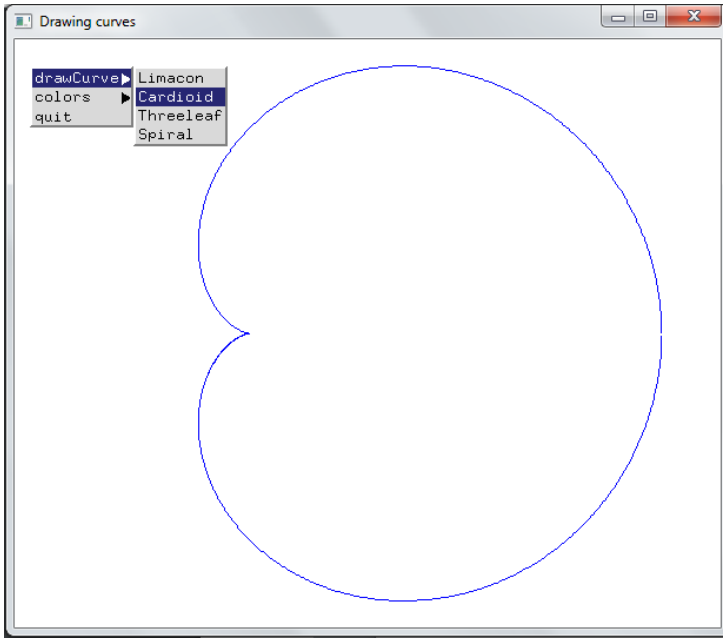
- Choose menu options to draw respective objects.

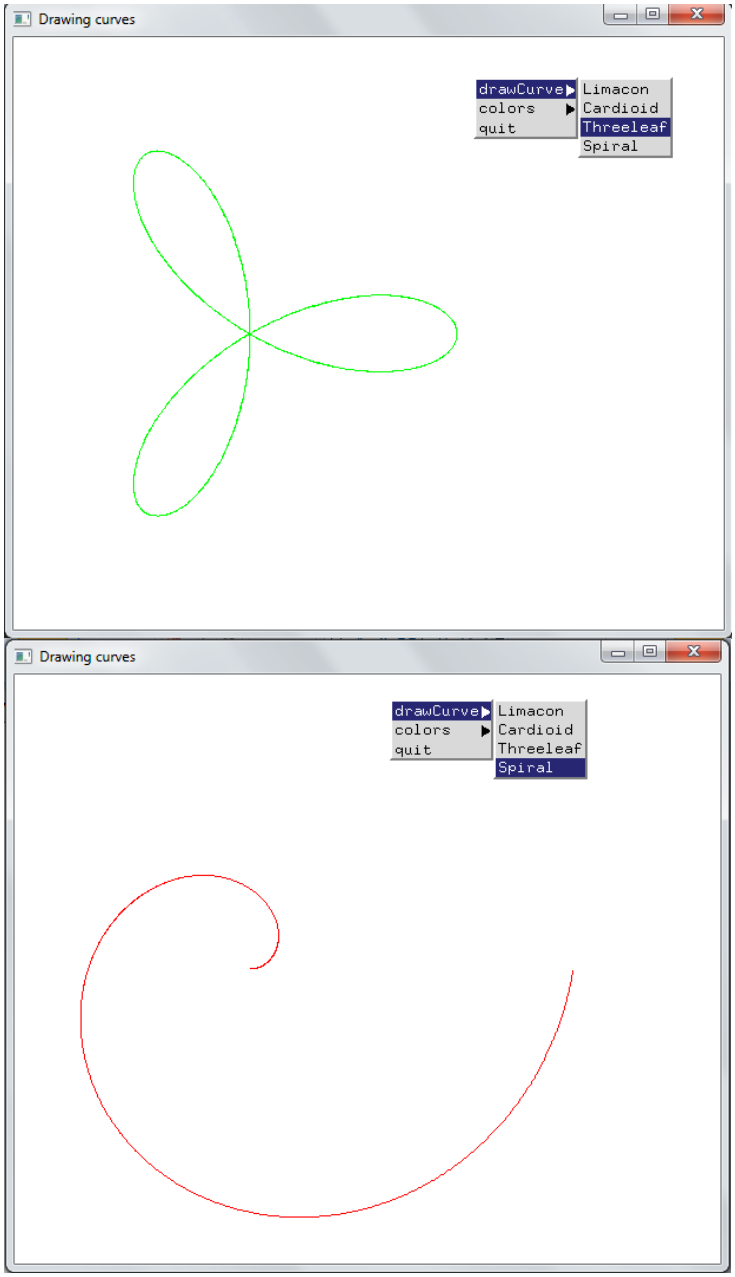


- Choose submenu ‘colors’ to draw respective objects in the selected color(within eight colors).

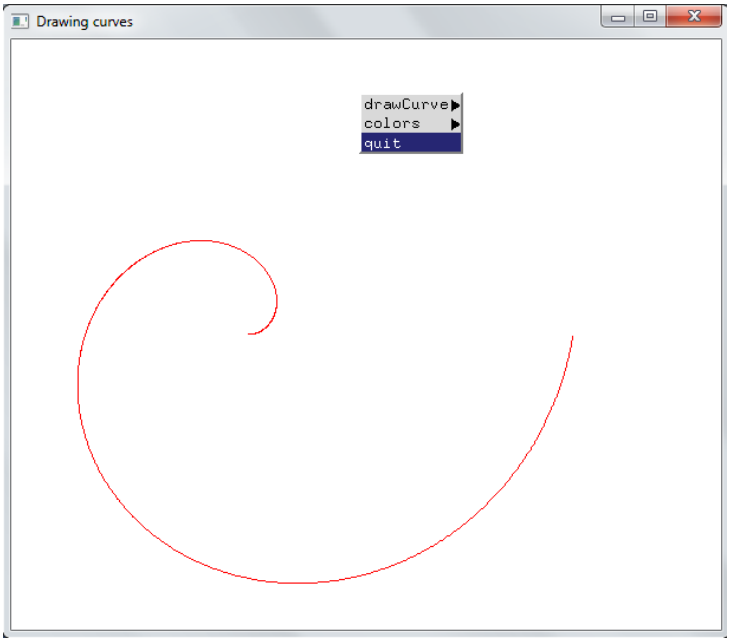


- Select limacon/cardioid/threeleaf/spiral to draw Respective object in selected color.




Drawing curves
drawCurve
colors
quit
Limacon
Cardioid
Threeleaf
Spiral
Drawing curves
drawCurve
colors
quit
Limacon
Cardioid
Threeleaf
Spiral

- select quit to exit.



Program 12

12. Write a program to construct Bezier curve. Control points are supplied through keyboard/ mouse.

```
#include<iostream>
#include<math.h>
#include<gl/glut.h>
using namespace std;
float f, g, r, x1[4], yc[4];
int flag = 0;
void myInit() {
glClearColor(1, 1, 1, 1);
glColor3f(1, 1, 1);
glPointSize(5);
gluOrtho2D(0, 500, 0, 500);
}
void drawPixel(float x, float y) {
glBegin(GL_POINTS);
glVertex2f(x, y);
glEnd();
}
void display() {
glClear(GL_COLOR_BUFFER_BIT);
int i;
double t;
glColor3f(0, 0, 0);
glBegin(GL_POINTS);
```

```
for (t = 0; t < 1; t = t + 0.005) {
double xt = pow(1 - t, 3) * x1[0] + 3 * t * pow(1 - t, 2) * x1[1] + 3 * pow(t, 2) * (1 - t) *
x1[2] + pow(t, 3) * x1[3];
double yt = pow(1 - t, 3) * yc[0] + 3 * t * pow(1 - t, 2) * yc[1] + 3 * pow(t, 2) * (1 - t) *
yc[2] + pow(t, 3) * yc[3];
glVertex2f(xt, yt);
}
glColor3f(1, 1, 0);
for (i = 0; i < 4; i++) {
glVertex2f(x1[i], yc[i]);
glEnd();
glFlush();
}
}
void mymouse(int btn, int state, int x, int y)
{
if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN && flag < 4)
{
x1[flag] = x;
yc[flag] = 500 - y;
cout << " X: " << x << " Y" << 500 - y;
glPointSize(3);
glColor3f(1, 1, 0);
glBegin(GL_POINTS);
glVertex2i(x, 500 - y);
```

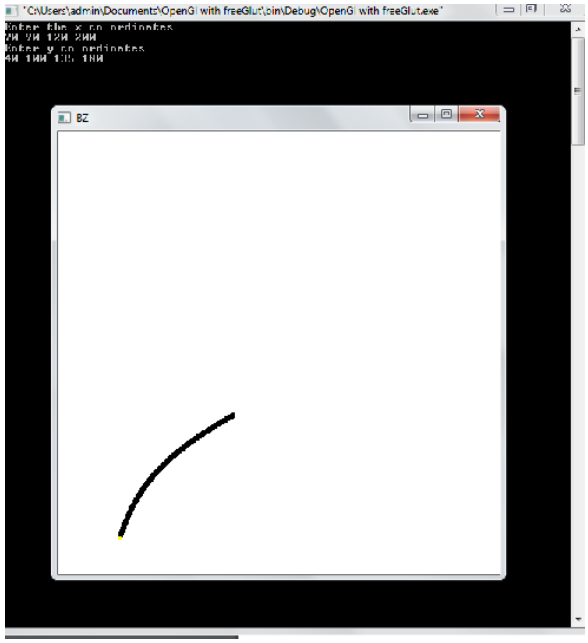
```
glEnd();
glFlush();
flag++;
}
if (flag >= 4 && btn == GLUT_LEFT_BUTTON)
{
glColor3f(0, 0, 1);
display();
flag = 0;
})
int main(int argc, char* argv[]) {
glutInit(&argc, argv);
//USE KEYBOARD
cout << "Enter the x co-ordinates";
cin >> x1[0] >> x1[1] >> x1[2] >> x1[3];
cout << "Enter y co-ordinates";
cin >> yc[0] >> yc[1] >> yc[2] >> yc[3];
//END KEYBOARD
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(500, 500);
glutInitWindowPosition(0, 0);
glutCreateWindow("BZ");
glutDisplayFunc(display);
//glutMouseFunc(mymouse); //INCLUDE FOR MOUSE, REMOVE FOR KEYBOARd
myInit();
```

```
glutMainLoop();

}
```

Output:

- Drawing BZ curve by inputting control points through keyboard.



- Drawing BZ curve through Mouse