

### LAB-3

**Implement Bresenham circle drawing algorithm to draw any object. The object should be displayed in all the quadrants with respect to center and radius.**

#### Source Code :

```
#include<GL/glut.h>
#include<iostream>
using namespace std;

int r;

void E_way(int x, int y){

glBegin(GL_POINTS);
    glVertex2i(x+320,y+240);
    glVertex2i(y+320,x+240);
    glVertex2i(y+320, -x+240);
    glVertex2i(x+320, -y+240);
    glVertex2i(-x+320,-y+240);
    glVertex2i(-y+320,-x+240);
    glVertex2i(-y+320,x+240);
    glVertex2i(-x+320,y+240);
glEnd();
glFlush();

}

void B_circle(){

    float d;
    d = 3 - 2*r;

    int x,y;
    x = 0 ;
    y = r ;

    do{
        E_way(x,y);

        if(d<0){

            d=d+4*x+6;

        }
        else{

            d= d+4*(x-y)+10;
            y=y-1;

        }

    }
```

```

        x=x+1;

    }while(x<y);

}

void init(){

glClearColor(1,1,1,0);
    glColor3f(1,0,0);
    gluOrtho2D(0,640,0,480);
glClear(GL_COLOR_BUFFER_BIT);
}

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

cout<<"\n Enter Radius \t ";
cin>>r;

glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

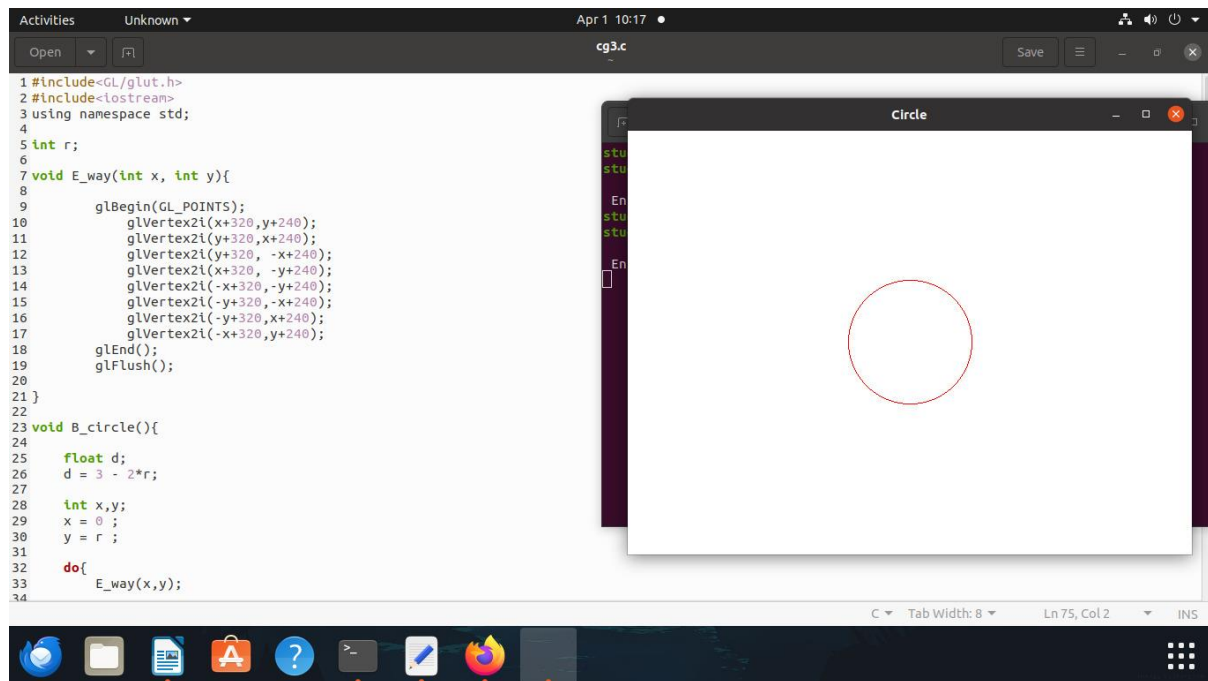
glutInitWindowPosition(100,100);
glutInitWindowSize(640,480);
glutCreateWindow("Circle");
init();
glutDisplayFunc(B_circle);

glutMainLoop();

    return 0;
}

```

## Output :



```
1 #include<GL/glut.h>
2 #include<iostream>
3 using namespace std;
4
5 int r;
6
7 void E_way(int x, int y){
8
9     glBegin(GL_POINTS);
10     glVertex2i(x+320,y+240);
11     glVertex2i(y+320,x+240);
12     glVertex2i(y+320, -x+240);
13     glVertex2i(x+320, -y+240);
14     glVertex2i(-x+320, -y+240);
15     glVertex2i(-y+320,-x+240);
16     glVertex2i(-y+320,x+240);
17     glVertex2i(-x+320,y+240);
18     glEnd();
19     glFlush();
20 }
21
22
23 void B_circle(){
24
25     float d;
26     d = 3 - 2*r;
27
28     int x,y;
29     x = 0 ;
30     y = r ;
31
32     do{
33         E_way(x,y);
34     }
```