PRIMITIVE.C

```
/*
 * GL02Primitive.cpp: Vertex, Primitive and Color
 * Draw Simple 2D colored Shapes: quad, triangle and polygon.
 */

#include <GL/glut.h>  // GLUT, include glu.h and gl.h

/* Initialize OpenGL Graphics */
void init(void)
{
glClearColor(0.0,0.0,0.0,1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-100,100,-100,100);
}

/* Handler for window-repaint event. Call back when the window first appears and
   whenever the window needs to be re-painted. */
void display() {
   glClear(GL_COLOR_BUFFER_BIT);
glPointSize(4.0f);
 glBegin(GL_LINES);

    glVertex2f(-100.0f, 0.0f);
    glVertex2f(100.0f, 0.0f);

    glVertex2f(0.0f, 100.0f);
    glVertex2f(0.0f, -100.0f);

   glEnd();
glBegin(GL_LINE_LOOP);
         glColor3f(1.0f, 0.0f, 5.0f);
    glVertex2f(10.0f, 10.0f);
    glVertex2f(40.0f, 10.0f);

    glVertex2f(40.0f, 40.0f);
   glVertex2f(10.0f, 40.0f);

   glEnd();

glBegin(GL_LINE_LOOP);
         glColor3f(1.0f, 0.0f, 5.0f);
    glVertex2f(25.0f, 25.0f);
    glVertex2f(55.0f, 25.0f);
      glVertex2f(55.0f, 55.0f);
    glVertex2f(25.0f, 55.0f);

   glEnd();

glBegin(GL_LINES);
         glColor3f(1.0f, 0.0f, 5.0f);
    glVertex2f(10.0f, 10.0f);
glVertex2f(25.0f, 25.0f);
    glVertex2f(40.0f, 10.0f);
      glVertex2f(55.0f, 25.0f);
    glVertex2f(40.0f, 40.0f);
 glVertex2f(55.0f, 55.0f);
   glVertex2f(10.0f, 40.0f);
 glVertex2f(25.0f, 55.0f);
```

```
   glEnd();



   glFlush();  // Render now
}

/* Main function: GLUT runs as a console application starting at main()  */
int main(int argc, char** argv) {
  glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (50, 50);
glutCreateWindow ("Cube");
init();
glutDisplayFunc(display);
glutMainLoop();                // Enter the event-processing loop
   return 0;
}
```

DDA LINE GENERATION

```
#include<GL/glut.h>
#include<stdlib.h>
#include<stdio.h>
int x1,x2,y1,y2;

void display(void)
{
float dy,dx,length,x,y,k,Xin,Yin;

dx=x2-x1;
dy=y2-y1;

if(abs(dx)>= abs(dy))
{
length = abs(dx);
}
else
length = abs(dy);

Xin = dx/length;
Yin = dy/length;

x= x1;
y= y1;
 glBegin(GL_LINES);
     // glColor3f(0.0f, 1.0f, 0.0f);


      glVertex2f(-100.0f, 0.0f);
 glVertex2f(100.0f, 0.0f);

glVertex2f(0.0f, -100.0f);
 glVertex2f(0.0f, 100.0f);
   glEnd();
glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
// glColor3f(0.0f, 1.0f, 0.0f);
for (k=1;k<=length;k++)
{
x= x + Xin;
y= y + Yin;

glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
}


glFlush();
}

void init(void)
{
glClearColor(0.0,0.0,0.0,1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-100,100,-100,100);
}

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

printf("Enter the value of x1 : ");
scanf("%d",&x1);
printf("Enter the value of y1 : ");
scanf("%d",&y1);
printf("Enter the value of x2 : ");
scanf("%d",&x2);
printf("Enter the value of y2 : ");
scanf("%d",&y2);

glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (50, 50);
glutCreateWindow ("DDA Line Algorithm");
init();
glutDisplayFunc(display);
glutMainLoop();

return 0;
}
```

3.BRESENHAMS LINE GENERATION

```
#include <GL/glut.h>
#include <stdio.h>
int x1,y1,x2,y2;
```

```c
void myInit()
{
  glClearColor(0.0,0.0,0.0,1.0);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D (-250,250,-250,250);
}

void draw_pixel(int x,int y)
{
  glBegin(GL_POINTS);
  glVertex2i(x,y);
  glEnd();
}
void draw_line(int x1, int x2, int y1, int y2)
{
  int dx, dy, i, e;
  int incx,incy,e1,e2;
  int x,y;

  dx=x2-x1;
  dy=y2-y1;

  if(dx < 0)
  dx=-dx;

  if(dy < 0)
  dy=-dy;

  incx = 1;

  if(x2 <x1)
  incx =-1;

  incy=1;

  if (y2 < y1)
  incy = -1;

  x=x1; y=y1;

  if (dx > dy)
{
  draw_pixel (x,y);
  e = 2* dy-dx;
  e1 = 2*(dy-dx);
  e2 = 2*dy;


  for ( i=0; i<dx; i++)
  {
   if (e>=0)
   {
    y+= incy;
    e = e+e1;
   }
   else
    e = e+e2;
    x += incx;
   draw_pixel(x,y);
  }
}
else
{
draw_pixel(x,y);
e = 2* dx-dy;
e1 =2* (dx-dy);
e2 =2*dx;

for(i=0; i<dy; i++)
{
if(e>=0)
{
x=x+incx;
e=e+e1;
}
else
e=e+e2;
y=y+incy;
draw_pixel(x,y);
}
}
}

void myDisplay()
{
  draw_line(x1, x2, y1, y2);
  glFlush();
}


void main(int argc,char **argv)
{
  printf("Enter (x1,x2,y1,y2)\n");
  scanf("%d %d %d %d",&x1,&y1,&x2,&y2);

  glutInit(&argc,argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition(0,0);
  glutCreateWindow("Bresenham's Line Drawing");
  glutInitWindowSize(500,500);
  myInit();
  glutDisplayFunc(myDisplay);
  glutMainLoop();
}
```

4.BRESENHAMS CIRCLE GENERATION

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

void plot_point(int x,int y)
{
   glBegin(GL_LINES);
   glColor3f(0.0f,1.0f,0.0f);
   glVertex2i(-320,0);
   glVertex2i(320,0);
   glVertex2i(0,-240);
   glVertex2i(0,240);
   glEnd();

   glBegin(GL_POINTS);
   glColor3f(1.0f,1.0f,1.0f);
   glVertex2i(x,y);
   glVertex2i(y,x);
   glVertex2i(x,-y);
   glVertex2i(y,-x);
   glVertex2i(-x,-y);
   glVertex2i(-y,-x);
   glVertex2i(-x,y);
   glVertex2i(-y,x);
   glEnd();
}

void bresenham_circle(int r)
{
   int x=0, y=r;
   float pk=3-2*r;
   plot_point(x,y);
   int k;
   while(x<y)
   {
     x=x+1;
     if(pk<0)
         pk=pk+4*x+6;
     else
     {
          y=y-1;
          pk=pk+4*(x-y)+10;
     }
    plot_point(x,y);
}
glFlush();
}
void concentric_circles(void)
{
  glClear(GL_COLOR_BUFFER_BIT);
  int radius1=50;
  bresenham_circle(radius1);
}
void Init()
{
glClearColor(0.0,0.0,0.0,0.0);
glColor3f(1.0,0.0,0.0);
gluOrtho2D(-320 ,320,-240,240);
}
void main(int argc, char ** argv)
{
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize(640,480);
   glutInitWindowPosition(0,0);
   glutCreateWindow("Bresenham Circle");
   Init();
   glutDisplayFunc(concentric_circles);
   glutMainLoop();
}
```

5.BOUNDARY FILL

```
#include <iostream>
#include <math.h>
//#include <time.h>
#include <GL/glut.h>

using namespace std;

void init()
       {
              glClearColor(0.0,0.0,0.0,1.0);
              glMatrixMode(GL_PROJECTION);
              gluOrtho2D(0,640,0,480);
       }

void flood_it(int x, int y, float* fillColor, float* bc)
       {
              float color[3];

              //to read the current pixel information
              glReadPixels(x,y,1.0,1.0,GL_RGB,GL_FLOAT,color);

              //checking current pixel color is not equal to boundary color or new
filling color
              if((color[0]!=bc[0] || color[1]!=bc[1] || color[2]!=bc[2])
              &&(color[0]!=fillColor[0] || color[1]!=fillColor[1] ||
color[2]!=fillColor[2]))
              {

              //to fill the pixel by new color
               glColor3f(fillColor[0],fillColor[1],fillColor[2]);

               glBegin(GL_POINTS);
              glVertex2i(x,y);
               glEnd();
               glFlush();

              //recursive call to the function

               flood_it(x+1,y,fillColor,bc);
               flood_it(x-2,y,fillColor,bc);
              flood_it(x,y+1,fillColor,bc);
               flood_it(x,y-2,fillColor,bc);


              }
       }
//mouse callback function
void mouse(int btn, int state, int x, int y)
       {
              y = 480-y;
              if(btn==GLUT_LEFT_BUTTON)

              {
                     if(state==GLUT_DOWN)
       {
                            float bCol[] = {1,0,0};
                            float color[] = {0,1,0};
                             flood_it(x,y,color,bCol);
                                   }
              }
       }
//to draw the object
void world()
       {
        glLineWidth(2);
        glPointSize(2);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1,0,0);
        glBegin(GL_LINE_LOOP);
         glVertex2i(150,100);
         glVertex2i(300,300);
         glVertex2i(450,100);
        glEnd();
        glFlush();
       }

int main(int argc, char** argv)
       {
       glutInit(&argc, argv);                          //initialization of the GLUT
       glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);   //to intialize the display mode
       glutInitWindowSize(640,480);                //to set the size of the window
       glutInitWindowPosition(50,50);     //to set the position of the window
       glutCreateWindow("Polygon Fill");  //to give user defined name to the window
       init();
       glutDisplayFunc(world);       //to creat the object
       glutMouseFunc(mouse);       //glutMouseFunc sets the mouse callback for the
current window.


       glutMainLoop();
       return 0;
       }
```

6.FLOOD FILL

```
//#include <iostream>
#include<stdio.h>
#include <math.h>
//#include <time.h>
#include <GL/glut.h>

//using namespace std;

void init()
       {
              glClearColor(0.0,0.0,0.0,1.0);
              glMatrixMode(GL_PROJECTION);
              gluOrtho2D(0,640,0,480);
       }

void flood_it(int x, int y, float* bc)
       {
              float color[3];

              //to read the current pixel information
              glReadPixels(x,y,1.0,1.0,GL_RGB,GL_FLOAT,color);

              //checking current pixel color is not equal to boundary color
              if(color[0]!=bc[0] || color[1]!=bc[1] || color[2]!=bc[2])
              {

              //to fill the pixel by new color
               glColor3f(bc[0],bc[1],bc[2]);

               glBegin(GL_POINTS);
              glVertex2i(x,y);
               glEnd();
               glFlush();

              //recursive call to the function

               flood_it(x+1,y,bc);
               flood_it(x-2,y,bc);
              flood_it(x,y+1,bc);
               flood_it(x,y-2,bc);


              }
       }
//mouse callback function
void mouse(int btn, int state, int x, int y)
       {
              y = 480-y;
              if(btn==GLUT_LEFT_BUTTON)

              {
                     if(state==GLUT_DOWN)
       {
                            float bCol[] = {1,1,0};

                             flood_it(x,y,bCol);
                                   }
              }
       }
//to draw the object
void world()
       {
        glLineWidth(2);
        glPointSize(2);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1,1,0);
        glBegin(GL_LINE_LOOP);
         glVertex2i(150,100);
         glVertex2i(300,300);
         glVertex2i(450,100);
        glEnd();
        glFlush();
       }

int main(int argc, char** argv)
       {
       glutInit(&argc, argv);                          //initialization of the GLUT
       glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);   //to intialize the display mode
       glutInitWindowSize(640,480);              //to set the size of the window
       glutInitWindowPosition(50,50);     //to set the position of the window
       glutCreateWindow("Polygon Fill");  //to give user defined name to the window
       init();
       glutDisplayFunc(world);       //to creat the object
       glutMouseFunc(mouse);       //glutMouseFunc sets the mouse callback for the
current window.
```

```
       glutMainLoop();
       return 0;
       }
```

7.2D TRANSLATION

```
#include <GL/glut.h>  // GLUT, include glu.h and gl.h

/* Initialize OpenGL Graphics */
void init(void)
       {
       glClearColor(0.0,0.0,0.0,1.0);
       glMatrixMode(GL_PROJECTION);
       glLoadIdentity();
       gluOrtho2D(-100,100,-100,100);
       }

/* Handler for window-repaint event. Call back when the window first appears and
   whenever the window needs to be re-painted. */
void display()
 {

        int tx=50, ty=10;
        glClear(GL_COLOR_BUFFER_BIT);
        glPointSize(4.0f);
        glBegin(GL_LINES);

        glVertex2i(-100.0f, 0.0f);
       glVertex2i(100.0f, 0.0f);

       glVertex2i(0.0f, 100.0f);
       glVertex2i(0.0f, -100.0f);

        glEnd();

       glBegin(GL_LINES);
       glColor3f(0.0f, 1.0f, 0.0f);
       glVertex2i(10, 10);
       glVertex2i(40, 40);


       glEnd();



          glBegin(GL_LINES);
         glColor3f(1.0f, 0.0f, 1.0f);
         glVertex2i(10+tx, 10+ty);


       glVertex2i(40+tx, 40+ty);


       glEnd();


       glFlush();  // Render now
}

/* Main function: GLUT runs as a console application starting at main()  */
```

```
int main(int argc, char** argv)
       {
        glutInit(&argc, argv);
       glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
       glutInitWindowSize (500, 500);
       glutInitWindowPosition (50, 50);
       glutCreateWindow ("Translation");
       init();
       glutDisplayFunc(display);
       glutMainLoop();                // Enter the event-processing loop
        return 0;
       }
```

8.2D SCALLING

```
#include <GL/glut.h>  // GLUT, include glu.h and gl.h

/* Initialize OpenGL Graphics */
void init(void)
{
glClearColor(0.0,0.0,0.0,1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-100,100,-100,100);
}

/* Handler for window-repaint event. Call back when the window first appears and
   whenever the window needs to be re-painted. */
void display()
 {

   float sx=1, sy=1;
   glClear(GL_COLOR_BUFFER_BIT);
glPointSize(4.0f);
 glBegin(GL_LINE);

    glVertex2i(-100.0f, 0.0f);
    glVertex2i(100.0f, 0.0f);

    glVertex2i(0.0f, 100.0f);
    glVertex2i(0.0f, -100.0f);

   glEnd();

glBegin(GL_LINE_LOOP);
         glColor3f(0.0f, 1.0f, 0.0f);
        glVertex2i(10, 10);


       glVertex2i(40, 10);
       glVertex2i(20, 40);


   glEnd();



glBegin(GL_LINE_LOOP);
         glColor3f(1.0f, 0.0f, 1.0f);
        glVertex2i(10*sx, 10*sy);
       glVertex2i(40*sx, 10*sy);
       glVertex2i(20*sx, 40*sy);


   glEnd();

   glFlush();  // Render now
}

/* Main function: GLUT runs as a console application starting at main()  */
int main(int argc, char** argv) {
  glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (50, 50);
glutCreateWindow ("Cube");
init();
glutDisplayFunc(display);
glutMainLoop();                // Enter the event-processing loop
   return 0;
}
```

9 COHEN SUTHERLAND LINE CLIPPING.

```
#include<GL/glut.h>
#include<math.h>
#include<stdio.h>
//#include<iostream>

void display();
//using namespace std;
float xmin=-100;
float ymin=-100;
float xmax=100;
float ymax=100;
float xd1,yd1,xd2,yd2;


void init(void)
{

    glClearColor(0.0,0,0,0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(-300,300,-300,300);

}

//to assign the outcode to the end points of the line
int code(float x,float y)
       {
        int c=0;
        if(y>ymax)
              c=8;
        if(y<ymin)
               c=4;
        if(x>xmax)
              c=2;
         if(x<xmin)
              c=1;
         return c;
       }
```

```
//to find the intersection of the line with the boundary of the cliping window
void cohen_Line(float x1,float y1,float x2,float y2)
{
    int c1=code(x1,y1);
    int c2=code(x2,y2);
    float m=(y2-y1)/(x2-x1);
    while((c1|c2)>0)
    {
        if((c1 & c2)>0)
        {
           exit(0);
        }

    float xi=x1;float yi=y1;
    int c=c1;
    if(c==0)
    {
         c=c2;
         xi=x2;
         yi=y2;
    }
    float x,y;
    if((c & 8)>0)
    {
       y=ymax;
       x=xi+ 1.0/m*(ymax-yi);
    }
    else
      if((c & 4)>0)
      {
          y=ymin;
          x=xi+1.0/m*(ymin-yi);
      }
      else
       if((c & 2)>0)
       {
           x=xmax;
           y=yi+m*(xmax-xi);
       }
       else
       if((c & 1)>0)
       {
           x=xmin;
           y=yi+m*(xmin-xi);
       }

       if(c==c1)
       {
           xd1=x;
           yd1=y;
           c1=code(xd1,yd1);
       }

       if(c==c2)
       {
           xd2=x;
           yd2=y;
           c2=code(xd2,yd2);
       }
}

 display();
}

//to clip the line through the keyboard key 'c'
void mykey(unsigned char key,int x,int y)
       {
        if(key=='c')
       {
              cohen_Line(xd1,yd1,xd2,yd2);
                      glFlush();
       }
       }

//to display the cliping window and line before and after cliping
void display()
       {

       glClear(GL_COLOR_BUFFER_BIT);
       glColor3f(0.0,1.0,0.0);

       glBegin(GL_LINE_LOOP);
       glVertex2i(xmin,ymin);
       glVertex2i(xmin,ymax);
       glVertex2i(xmax,ymax);
       glVertex2i(xmax,ymin);
       glEnd();

       glColor3f(1.0,0.0,0.0);
       glBegin(GL_LINES);
       glVertex2i(xd1,yd1);
       glVertex2i(xd2,yd2);
       glEnd();
       glFlush();


       }


int main(int argc,char** argv)
{
    printf("Enter line co-ordinates:");
    scanf("%d" ,xd1 , yd1,xd2,yd2) ;
   // cin>>xd1>>yd1>>xd2>>yd2;
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(600,600);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Cohen Sutherland Line Clipping");
    glutDisplayFunc(display);             //call to the display function
    glutKeyboardFunc(mykey);              //call to the keyboard function
    init();
    glutMainLoop();
    return 0;
}
```