COMPUTER GRAPHICS

dda:

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

void init()
{
glClearColor(0,0,0,0);
gluOrtho2D(-500,500,-500,500);
}
void display()
{
int i,xa,ya,xb,yb,x,y,dx,dy,steps,xincr,yincr;
xa=0;
ya=100;
xb=100;
yb=200;
dx=xb-xa;
dy=yb-ya;
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glPointSize(5);
glBegin(GL_POINTS);
if(abs(dx)>=abs(dy))
steps=abs(dx);
else
steps=abs(dy);
xincr=dx/steps;
yincr=dy/steps;
x=xa;
y=ya;
while(x<=steps)
{
glVertex2d(x,y);
x=x+xincr;
y=y+yincr;
}

for(i=-500;i<=500;i++)
{
glVertex2d(i,0);
glVertex2d(0,i);
}
glEnd();
glFlush();
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowSize(1000,1000);
glutInitWindowPosition(10,10);
glutCreateWindow("dda");
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
init();
glutDisplayFunc(display);
glutMainLoop();

}
```

bresenham:

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

void init()
{
glClearColor(0,0,0,0);
gluOrtho2D(-500,500,-500,500);
}
void display()
{
int i,xa,ya,xb,yb,x,y,dx,dy,p,xend,yend;
xa=0;
ya=-100;
xb=-100;
yb=0;
dx=xb-xa;
dy=yb-ya;
p=2*abs(dy)-dx;
glClear(GL_COLOR_BUFFER_BIT);
```

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glPointSize(5);
glBegin(GL_POINTS);
if(xa>xb)
{
x=xb;
y=yb;
xend=xa;
dy=-dy;
}
else
{
x=xa;
y=ya;
xend=xb;
}
if(dx!=0)
{
while(x<=xend)
{
glVertex2d(x,y);
if(p<0)
{
p+=2*abs(dy);
}
else
{
p+=2*abs(dy-dx);
if(dy>0)
y++;
else
y--;
}
x++;
}
}
else
{
while(y<=yend)
{
glVertex2d(x,y);
y++;
}
}
for(i=-500;i<=500;i++)
{
glVertex2d(i,0);
glVertex2d(0,i);
}
glEnd();
glFlush();
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowSize(1000,1000);
glutInitWindowPosition(10,10);
glutCreateWindow("bresenham");
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
init();
glutDisplayFunc(display);
glutMainLoop();

}
```

line attributes:

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

void init()
{
glClearColor(0,0,0,0);
gluOrtho2D(-500,500,-500,500);
}
void display()
{

glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glShadeModel(GL_SMOOTH);
glLineWidth(2);
glEnable(GL_LINE_STIPPLE);
//GLushort pattern=0xAAAA;
glLineStipple(10,0xAAAA);
glBegin(GL_LINES);
glColor3f(1,0,1);
glVertex2d(0,0);
glColor3f(1,0,0);
glVertex2d(0,1000);
glEnd();
glFlush();
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowSize(1000,1000);
glutInitWindowPosition(10,10);
glutCreateWindow("line attributes");
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
init();
glutDisplayFunc(display);
glutMainLoop();

}
```

circle:

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

void init()
{
glClearColor(0,0,0,0);
gluOrtho2D(-500,500,-500,500);
}
void display()
{
int xc,yc,x,y,r,p,i;
xc=0;
yc=0;
r=50;
x=0;
```

```
y=r;
p=1-r;
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glPointSize(5);

glBegin(GL_LINES);
while(x<=y)
{
glVertex2d(xc+x,yc+y);
glVertex2d(xc-x,yc-y);
glVertex2d(xc+x,yc-y);
glVertex2d(xc-x,yc+y);

glVertex2d(xc+y,yc+x);
glVertex2d(xc-y,yc-x);
glVertex2d(xc+y,yc-x);
glVertex2d(xc-y,yc+x);
if(p<0)
{
p+=2*x+3;
}
else
{
p+=2*(x-y)+5;
y--;
}
x++;
}
glEnd();
glBegin(GL_POINTS);
for(i=-500;i<=500;i++)
{
glVertex2d(i,0);
glVertex2d(0,i);
}
glEnd();
glFlush();
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
```

```
glutInitWindowSize(500,500);
glutInitWindowPosition(10,10);
glutCreateWindow("dda");
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
init();
glutDisplayFunc(display);
glutMainLoop();

}
```

ellipse:

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

void init()
{
glClearColor(0,0,0,0);
gluOrtho2D(-500,500,-500,500);
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glPointSize(1);

int xc,yc,x,y,rx,ry,p,i,px,py,rxsq,rysq;
xc=0;
yc=0;
rx=70;
ry=170;
x=0;
rxsq=rx*rx;
rysq=ry*ry;
y=ry;
px=0;
py=2*rxsq*y;

glBegin(GL_POINTS);
```

```
glVertex2d(xc+x,yc+y);
glVertex2d(xc+x,yc-y);
glVertex2d(xc-x,yc+y);
glVertex2d(xc-x,yc-y);
p=rysq+0.25*rxsq-rxsq*ry;
while(px<py)
{
x++;
px=px+2*rysq;
if(p<0)
{
p=p+rysq+px;
}
else
{
y--;
py=py-2*rxsq;
p=p+rysq+px-py;
}
glVertex2d(xc+x,yc+y);
glVertex2d(xc+x,yc-y);
glVertex2d(xc-x,yc+y);
glVertex2d(xc-x,yc-y);
}


p=rysq*(x+0.5)*(x+0.5)+rxsq*(y-1)*(y-1)-rxsq*rysq;
while(y>0)
{
y--;
py=py-2*rxsq;
if(p>0)
{
p=p+rxsq-py;
}
else
{
x++;
px=px+2*rysq;
p=p+rxsq+px-py;
}
glVertex2d(xc+x,yc+y);
glVertex2d(xc+x,yc-y);
glVertex2d(xc-x,yc+y);
glVertex2d(xc-x,yc-y);
}

glEnd();
glFlush();
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowSize(1000,1000);
glutInitWindowPosition(10,10);
glutCreateWindow("ellipse");
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
init();
glutDisplayFunc(display);
glutMainLoop();

}
```

translation rotation scaling:

```
#include<iostream>
#include<GL/glut.h>
#include<math.h>
using namespace std;
int tx,ty,deg,sx=1,sy=1;
void init()
{
glClearColor(0,0,0,0);
gluOrtho2D(-500,500,-500,500);
}
void drawline(int xa,int ya,int xb,int yb)
{
int i,x,y,dx,dy,p,xend,yend;
dx=xb-xa;
dy=yb-ya;
p=2*abs(dy)-dx;
glPointSize(1);
glBegin(GL_POINTS);
if(xa>xb)
{
```

```
x=xb;
y=yb;
xend=xa;
dy=-dy;
}
else
{
x=xa;
y=ya;
xend=xb;
}
if(dx!=0)
{
while(x<=xend)
{
glVertex2d(tx+sx*(x*cos(deg*3.14/180)-y*sin(deg*3.14/180)),ty+sy*(x*sin(deg*3.14/180)+y*cos(
deg*3.14/180)));
if(p<0)
{
p+=2*abs(dy);
}
else
{
p+=2*abs(dy-dx);
if(dy>0)
y++;
else
y--;
}
x++;
}
}
else
{
while(y<yb)
{
glVertex2d(tx+sx*(x*cos(deg*3.14/180)-y*sin(deg*3.14/180)),ty+sy*(x*sin(deg*3.14/180)+y*cos(
deg*3.14/180)));
y++;
}
}
glEnd();
glFlush();
```

```
}
void drawcircle(int xc,int yc,int r)
{
int x,y,p,i;
x=0;
y=r;
p=1-r;
glPointSize(1);

glBegin(GL_POINTS);
while(x<=y)
{
glVertex2d(tx+sx*(xc+x),ty+sy*(yc+y));
glVertex2d(tx+sx*(xc-x),ty+sy*(yc-y));
glVertex2d(tx+sx*(xc+x),ty+sy*(yc-y));
glVertex2d(tx+sx*(xc-x),ty+sy*(yc+y));

glVertex2d(tx+sx*(xc+y),ty+sy*(yc+x));
glVertex2d(tx+sx*(xc-y),ty+sy*(yc-x));
glVertex2d(tx+sx*(xc+y),ty+sy*(yc-x));
glVertex2d(tx+sx*(xc-y),ty+sy*(yc+x));
if(p<0)
{
p+=2*x+3;
}
else
{
p+=2*(x-y)+5;
y--;
}
x++;
}
glEnd();
glFlush();
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
drawcircle(0,0,50);
drawline(0,-50,0,50);
drawline(-50,0,50,0);
```

```
tx=200;
ty=200;
deg=30;
sx=2;
sy=2;
drawcircle(0,0,50);
drawline(0,-50,0,50);
drawline(-50,0,50,0);

}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowSize(500,500);
glutInitWindowPosition(10,10);
glutCreateWindow("trans and rot and scaling");
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
init();
glutDisplayFunc(display);
glutMainLoop();
}
```

reflection shearing:

```
#include<iostream>
#include<GL/glut.h>
#include<math.h>
using namespace std;
int tx,ty,refx=1,refy=1,shx=0,shy=0;
void init()
{
glClearColor(0,0,0,0);
gluOrtho2D(-500,500,-500,500);
}
void drawline(int xa,int ya,int xb,int yb)
{
int i,x,y,dx,dy,p,xend,yend;
dx=xb-xa;
dy=yb-ya;
p=2*abs(dy)-dx;
glPointSize(1);
```

```
glBegin(GL_POINTS);
if(xa>xb)
{
x=xb;
y=yb;
xend=xa;
dy=-dy;
}
else
{
x=xa;
y=ya;
xend=xb;
}
if(dx!=0)
{
while(x<=xend)
{
glVertex2d(refy*(tx+x+shx*y),refx*(ty+y+shy*x));
if(p<0)
{
p+=2*abs(dy);
}
else
{
p+=2*abs(dy-dx);
if(dy>0)
y++;
else
y--;
}
x++;
}
}
else
{
while(y<yb)
{
glVertex2d(refy*(tx+x+shx*y),refx*(ty+y+shy*x));
y++;
}
}
glEnd();
```

```
glFlush();
}
void drawcircle(int xc,int yc,int r)
{
int x,y,p,i;
x=0;
y=r;
p=1-r;
glPointSize(1);

glBegin(GL_POINTS);
while(x<=y)
{
glVertex2d(refy*(tx+(xc+x)+shx*(yc+y)),refx*(ty+shy*(xc+x)+(yc+y)));
glVertex2d(refy*(tx+(xc-x)+shx*(yc-y)),refx*(ty+shy*(xc-x)+(yc-y)));
glVertex2d(refy*(tx+(xc+x)+shx*(yc-y)),refx*(ty+shy*(xc+x)+(yc-y)));
glVertex2d(refy*(tx+(xc-x)+shx*(yc+y)),refx*(ty+shy*(xc-x)+(yc+y)));

glVertex2d(refy*(tx+(xc+y)+shx*(yc+x)),refx*(ty+shy*(xc+y)+(yc+x)));
glVertex2d(refy*(tx+(xc-y)+shx*(yc-x)),refx*(ty+shy*(xc-y)+(yc-x)));
glVertex2d(refy*(tx+(xc+y)+shx*(yc-x)),refx*(ty+shy*(xc+y)+(yc-x)));
glVertex2d(refy*(tx+(xc-y)+shx*(yc+x)),refx*(ty+shy*(xc-y)+(yc+x)));
if(p<0)
{
p+=2*x+3;
}
else
{
p+=2*(x-y)+5;
y--;
}
x++;
}
glEnd();
glFlush();
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
drawcircle(0,0,50);
//drawline(0,-50,0,50);
```

```
//drawline(-50,0,50,0);

tx=200;
ty=200;
shx=2;
shy=2;
drawcircle(0,0,50);
refx=-1;
refy=1;
drawcircle(0,0,50);
refx=-1;
refy=-1;
drawcircle(0,0,50);
refx=1;
refy=-1;
drawcircle(0,0,50);

//drawline(0,-50,0,50);
//drawline(-50,0,50,0);

}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowSize(500,500);
glutInitWindowPosition(10,10);
glutCreateWindow("trans and rot and scaling");
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
init();
glutDisplayFunc(display);
glutMainLoop();
}
```

translation with timer:

```
#include<iostream>
#include<GL/glut.h>
#include<math.h>
using namespace std;
float tx,ty;
void init()
{
```

```
glClearColor(0,0,0,0);
gluOrtho2D(-500,500,-500,500);
}
void drawline(int xa,int ya,int xb,int yb)
{
int i,x,y,dx,dy,p,xend,yend;
dx=xb-xa;
dy=yb-ya;
p=2*abs(dy)-dx;
glPointSize(1);
glBegin(GL_POINTS);
if(xa>xb)
{
x=xb;
y=yb;
xend=xa;
dy=-dy;
}
else
{
x=xa;
y=ya;
xend=xb;
}
if(dx!=0)
{
while(x<=xend)
{
glVertex2d(tx+x,ty+y);
if(p<0)
{
p+=2*abs(dy);
}
else
{
p+=2*abs(dy-dx);
if(dy>0)
y++;
else
y--;
}
x++;
}
}
else
{
while(y<yb)
{
glVertex2d(tx+x,ty+y);
y++;
}
}
glEnd();
glFlush();
}
void drawcircle(int xc,int yc,int r)
{
int x,y,p,i;
x=0;
y=r;
p=1-r;
glPointSize(1);

glBegin(GL_POINTS);
while(x<=y)
{
glVertex2d(tx+(xc+x),ty+(yc+y));
glVertex2d(tx+(xc-x),ty+(yc-y));
glVertex2d(tx+(xc+x),ty+(yc-y));
glVertex2d(tx+(xc-x),ty+(yc+y));
glVertex2d(tx+(xc+y),ty+(yc+x));
glVertex2d(tx+(xc-y),ty+(yc-x));
glVertex2d(tx+(xc+y),ty+(yc-x));
glVertex2d(tx+(xc-y),ty+(yc+x));
if(p<0)
{
p+=2*x+3;
}
else
{
p+=2*(x-y)+5;
y--;
}
x++;
}
glEnd();
```

```
glFlush();
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
drawcircle(0,0,50);
drawline(0,-50,0,50);
drawline(-50,0,50,0);
}
void animate2()
{
while(tx>=0)
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
tx-=10;
ty-=10;
drawcircle(0,0,50);
drawline(0,-50,0,50);
drawline(-50,0,50,0);
}
}
void animate1()
{
while(tx<=200)
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
tx+=0.5;
ty+=0.5;
drawcircle(0,0,50);
drawline(0,-50,0,50);
drawline(-50,0,50,0);
}
}
void animate()
{
if(tx<=200)
{
```

```
animate1();
}
else
{
animate2();
}
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowSize(500,500);
glutInitWindowPosition(10,10);
glutCreateWindow("trans and rot and scaling");
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
init();
glutDisplayFunc(display);
glutIdleFunc(animate);
glutMainLoop();
}
```

julia:

```
#include<GL/glut.h>
#include<iostream>
using namespace std;
void init()
{
glClearColor(0,0,0,0);
gluOrtho2D(0,300,0,300);
}

int dwell(double sx,double sy)
{
double tmp,dx=sx,dy=sy,fsq=sx*sx+sy*sy;
int iter,maxiter=100;
for(iter=1;iter<=maxiter&&fsq<=4;iter++)
{
tmp=dx;
dx=dx*dx-dy*dy-0.5;
dy=2.0*tmp*dy+0.5;
fsq=dx*dx+dy*dy;
}
```

```
return iter;
}

void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
int iter,maxiter=100,hyres=300,hxres=300;
double sx;
double sy;
glBegin(GL_POINTS);
for(int hy=0;hy<hyres;hy++)
{
sy=(hy-hyres/2)/(0.5*1.5*hyres);
for(int hx=0;hx<hxres;hx++)
{
sx=(hx-hxres/2)/(0.5*1.5*hxres);
iter=dwell(sx,sy);
glBegin(GL_POINTS);
if(iter<=maxiter)
{

glColor3f(1,0,1);
glVertex2d(hx,hy);
}
else
{

glColor3f(1,0,0);
glVertex2d(hx,hy);
}
}
}
glEnd();
glFlush();
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(300,300);
glutInitWindowPosition(100,100);
```

```
glutCreateWindow("julia");
init();
glutDisplayFunc(display);
glutMainLoop();
}
```

mandelbrot:

```
#include<GL/glut.h>
int imght=1000,imgwt=1000;
void init()
{
glClearColor(0,0,0,0);
gluOrtho2D(0,1000,0,1000);
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
double rmin=-2.0;
double rmax=1.2;
double imin=-1.0;
double imax=imin+(rmax-rmin)*imght/imgwt;
double imfactor=(imax-imin)/(imght-1);
double refactor=(rmax-rmin)/(imgwt-1);
int maxiter=50;
glBegin(GL_POINTS);
for(unsigned y=0;y<imght;y++)
{
double cim=imin+y*imfactor;
for(unsigned x=0;x<imgwt;x++)
{
double cre=rmin+x*refactor;
bool inside=true;
double zim=cim;
double zre=cre;

for(int n=0;n<maxiter;n++)
{
double zre2=zre*zre;
double zim2=zim*zim;
if(zre2+zim2>4)
{
inside=false;
break;
}
zim=cim+2*zre*zim;
zre=cre+zre2-zim2;
}
if(inside)
{
glVertex2d(x,y);
}
}
}
glEnd();
glFlush();
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(1000,1000);
glutInitWindowPosition(100,100);
glutCreateWindow("mandelbrot");
init();
glutDisplayFunc(display);
glutMainLoop();
}
```

line clipping:

```
#include<iostream>
#include<GL/glut.h>
using namespace std;
struct Point
{
float x,y;
}w[4],over[20];
int nout=0;
void init()
{
glClearColor(0,0,0,0);
```

```
gluOrtho2D(-1000,1000,-1000,1000);
w[0].x=-100;
w[0].y=100;
w[1].x=100;
w[1].y=100;
w[2].x=100;
w[2].y=-100;
w[3].x=-100;
w[3].y=-100;
}
void drawpoly(Point t[],int n,int x)
{
glBegin(GL_LINE_LOOP);
for(int i=0;i<n;i++)
{
glVertex2d(t[i].x+x,t[i].y);
}
glEnd();
glFlush();
}

bool inside(Point t,int i)
{
if(i==0 && t.x>w[i].x)
return true;
if(i==1 && t.y<w[i].y)
return true;
if(i==2 && t.x<w[i].x)
return true;
if(i==3 && t.y>w[i].y)
return true;

return false;
}

void add(Point t)
{
over[nout]=t;
nout++;
}

Point findinter(Point p,Point q,int i)
{
```

```
Point inter;
if(i==0||i==2)
{
inter.x=w[i].x;
inter.y=q.y+(p.y-q.y)*(inter.x-q.x)/(p.x-q.x);
}
else
{
inter.y=w[i].y;
inter.x=q.x+(p.x-q.x)*(inter.y-q.y)/(p.y-q.y);
}
return inter;
}

void clipanddraw(Point iver[],int nin)
{
cout<<"hello";
Point p,q;
for(int i=0;i<4;i++)
{
nout=0;
p=iver[nin-1];
for(int j=0;j<nin;j++)
{
q=iver[j];
if(inside(p,i)==true && inside(q,i)==true)
add(q);
else if(inside(p,i)==true && inside(q,i)==false)
{
Point inter=findinter(p,q,i);
add(inter);
}
else if(inside(p,i)==false && inside(q,i)==true)
{
Point inter=findinter(p,q,i);
add(inter);
add(q);
}
else
{

}
p=q;
}

nin=nout;
for(int k=0;k<nin;k++)
iver[k]=over[k];
if(i==0)
drawpoly(iver,nin,-500);
if(i==1)
drawpoly(iver,nin,-200);
if(i==2)
drawpoly(iver,nin,100);
if(i==3)
drawpoly(iver,nin,400);
}
}

void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
Point iver[20];
int nin;
drawpoly(w,4,-800);
iver[0].x=-120;
iver[0].y=0;
iver[1].x=120;
iver[1].y=0;
nin=2;
drawpoly(w,4,-500);
drawpoly(w,4,-200);
drawpoly(w,4,100);
drawpoly(w,4,400);
drawpoly(iver,nin,-800);
clipanddraw(iver,nin);
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowSize(2000,2000);
glutInitWindowPosition(10,10);
glutCreateWindow("line");
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
```

```
init();
glutDisplayFunc(display);
glutMainLoop();

}
```

polygon clipping:

```
#include<iostream>
#include<GL/glut.h>
using namespace std;
struct Point
{
float x,y;
}w[4],over[20];
int nout=0;
void init()
{
glClearColor(0,0,0,0);
gluOrtho2D(-1000,1000,-1000,1000);
w[0].x=-100;
w[0].y=100;
w[1].x=100;
w[1].y=100;
w[2].x=100;
w[2].y=-100;
w[3].x=-100;
w[3].y=-100;
}
void drawpoly(Point t[],int n,int x)
{
glBegin(GL_LINE_LOOP);
for(int i=0;i<n;i++)
{
glVertex2d(t[i].x+x,t[i].y);
}
glEnd();
glFlush();
}

bool inside(Point t,int i)
{
if(i==0 && t.x>w[i].x)
```

```
return true;
if(i==1 && t.y<w[i].y)
return true;
if(i==2 && t.x<w[i].x)
return true;
if(i==3 && t.y>w[i].y)
return true;

return false;
}

void add(Point t)
{
over[nout]=t;
nout++;
}

Point findinter(Point p,Point q,int i)
{
Point inter;
if(i==0||i==2)
{
inter.x=w[i].x;
inter.y=q.y+(p.y-q.y)*(inter.x-q.x)/(p.x-q.x);
}
else
{
inter.y=w[i].y;
inter.x=q.x+(p.x-q.x)*(inter.y-q.y)/(p.y-q.y);
}
return inter;
}

void clipanddraw(Point iver[],int nin)
{
cout<<"hello";
Point p,q;
for(int i=0;i<4;i++)
{
nout=0;
p=iver[nin-1];
for(int j=0;j<nin;j++)
{
```

```
q=iver[j];
if(inside(p,i)==true && inside(q,i)==true)
add(q);
else if(inside(p,i)==true && inside(q,i)==false)
{
Point inter=findinter(p,q,i);
add(inter);
}
else if(inside(p,i)==false && inside(q,i)==true)
{
Point inter=findinter(p,q,i);
add(inter);
add(q);
}
else
{

}
p=q;
}

nin=nout;
for(int k=0;k<nin;k++)
iver[k]=over[k];
if(i==0)
drawpoly(iver,nin,-500);
if(i==1)
drawpoly(iver,nin,-200);
if(i==2)
drawpoly(iver,nin,100);
if(i==3)
drawpoly(iver,nin,400);
}
}

void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
Point iver[20];
int nin;
drawpoly(w,4,-800);
```

```
iver[0].x=-150;
iver[0].y=0;
iver[1].x=0;
iver[1].y=150;
iver[2].x=150;
iver[2].y=0;
iver[3].x=0;
iver[3].y=-150;
nin=4;
drawpoly(w,4,-500);
drawpoly(w,4,-200);
drawpoly(w,4,100);
drawpoly(w,4,400);
drawpoly(iver,nin,-800);
clipanddraw(iver,nin);
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowSize(2000,2000);
glutInitWindowPosition(10,10);
glutCreateWindow("poly");
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
init();
glutDisplayFunc(display);
glutMainLoop();

}
```

window to viewport:

```
#include<iostream>
#include<GL/glut.h>
using namespace std;
struct Point
{
float x,y;
};
void init()
{
glClearColor(0,0,0,0);
gluOrtho2D(-500,500,-500,500);
}
```

```
void drawpoly(Point p[],int n)
{
glBegin(GL_LINE_LOOP);
for(int i=0;i<n;i++)
glVertex2d(p[i].x,p[i].y);
glEnd();
glFlush();
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glPointSize(5);
float sx,sy;
Point w[4],v[4],trw[3],trv[3];
int xwmin=w[0].x=-300;
int ywmax=w[0].y=0;
int xwmax=w[1].x=0;
w[1].y=0;
w[2].x=0;
int ywmin=w[2].y=-300;
w[3].x=-300;
w[3].y=-300;
drawpoly(w,4);
int xvmin=v[0].x=100;
int yvmin=v[0].y=400;
int xvmax=v[1].x=400;
v[1].y=400;
v[2].x=400;
int yvmax=v[2].y=-400;
v[3].x=100;
v[3].y=-400;
drawpoly(v,4);
trw[0].x=-250;
trw[0].y=-100;
trw[1].x=-250;
trw[1].y=-200;
trw[2].x=-150;
trw[2].y=-200;
drawpoly(trw,3);
sx=(xvmax-xvmin)/(xwmax-xwmin);
sy=(yvmax-yvmin)/(ywmax-ywmin);
trv[0].x=xvmin+sx*(trw[0].x-xwmin);
trv[0].y=yvmin+sy*(trw[0].y-ywmin);
trv[1].x=xvmin+sx*(trw[1].x-xwmin);
trv[1].y=yvmin+sy*(trw[1].y-ywmin);
trv[2].x=xvmin+sx*(trw[2].x-xwmin);
trv[2].y=yvmin+sy*(trw[2].y-ywmin);
drawpoly(trv,3);
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowSize(500,500);
glutInitWindowPosition(10,10);
glutCreateWindow("dda");
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
init();
glutDisplayFunc(display);
glutMainLoop();

}
```

dice:

```
#include <GL/glut.h>
 GLfloat xRotated, yRotated, zRotated;
void init(void)
{
glClearColor(0,0,0,0);
}

void DrawCube()
{

        glMatrixMode(GL_MODELVIEW);
        glClear(GL_COLOR_BUFFER_BIT);
  glLoadIdentity();
        glTranslatef(0.0,0.0,-10.5);
        glRotatef(xRotated,1.0,0.0,0.0);
        // rotation about Y axis
        glRotatef(yRotated,0.0,1.0,0.0);
        // rotation about Z axis
        glRotatef(zRotated,0.0,0.0,1.0);
glPointSize(5);
  glBegin(GL_QUADS);          // Draw The Cube Using quads
        glColor3f(0.0f,1.0f,0.0f);        // Color Blue
        glVertex3f( 1.0f, 1.0f,-1.0f);    // Top Right Of The Quad (Top)
        glVertex3f(-1.0f, 1.0f,-1.0f);    // Top Left Of The Quad (Top)
        glVertex3f(-1.0f, 1.0f, 1.0f);    // Bottom Left Of The Quad (Top)
        glVertex3f( 1.0f, 1.0f, 1.0f);    // Bottom Right Of The Quad (Top)
        glColor3f(1.0f,0.5f,0.0f);        // Color Orange
        glVertex3f( 1.0f,-1.0f, 1.0f);    // Top Right Of The Quad (Bottom)
        glVertex3f(-1.0f,-1.0f, 1.0f);    // Top Left Of The Quad (Bottom)
        glVertex3f(-1.0f,-1.0f,-1.0f);    // Bottom Left Of The Quad (Bottom)
        glVertex3f( 1.0f,-1.0f,-1.0f);    // Bottom Right Of The Quad (Bottom)
        glColor3f(1.0f,1.0f,0.0f);        // Color Yellow
        glVertex3f( 1.0f,-1.0f,-1.0f);    // Top Right Of The Quad (Back)
        glVertex3f(-1.0f,-1.0f,-1.0f);    // Top Left Of The Quad (Back)
        glVertex3f(-1.0f, 1.0f,-1.0f);    // Bottom Left Of The Quad (Back)
        glVertex3f( 1.0f, 1.0f,-1.0f);    // Bottom Right Of The Quad (Back)
        glColor3f(0.0f,0.0f,1.0f);        // Color Blue
        glVertex3f(-1.0f, 1.0f, 1.0f);    // Top Right Of The Quad (Left)
        glVertex3f(-1.0f, 1.0f,-1.0f);    // Top Left Of The Quad (Left)
        glVertex3f(-1.0f,-1.0f,-1.0f);    // Bottom Left Of The Quad (Left)
        glVertex3f(-1.0f,-1.0f, 1.0f);    // Bottom Right Of The Quad (Left)
        glColor3f(1.0f,0.0f,1.0f);        // Color Violet
        glVertex3f( 1.0f, 1.0f,-1.0f);    // Top Right Of The Quad (Right)
        glVertex3f( 1.0f, 1.0f, 1.0f);    // Top Left Of The Quad (Right)
        glVertex3f( 1.0f,-1.0f, 1.0f);    // Bottom Left Of The Quad (Right)
        glVertex3f( 1.0f,-1.0f,-1.0f);    // Bottom Right Of The Quad (Right)
glColor3f(1.0f,0.0f,0.0f);        // Color Red
        glVertex3f( 1.0f, 1.0f, 1.0f);    // Top Right Of The Quad (Front)
        glVertex3f(-1.0f, 1.0f, 1.0f);    // Top Left Of The Quad (Front)
        glVertex3f(-1.0f,-1.0f, 1.0f);    // Bottom Left Of The Quad (Front)
        glVertex3f( 1.0f,-1.0f, 1.0f);    // Bottom Right Of The Quad (Front)

glEnd();          // End Drawing The Cube

glBegin(GL_POINTS);
glColor3f(1.0f,1.0f,1.0f);
glVertex3f(0.0f,0.0f,1.0f);
glEnd();
glFlush();
}
```

```
void animation()
{

	yRotated += 0.01;
	xRotated += 0.02;
	DrawCube();
}


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();
	//Angle of view:40 degrees
	//Near clipping plane distance: 0.5
	//Far clipping plane distance: 20.0

	gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.0);
	glMatrixMode(GL_MODELVIEW);
	glViewport(0,0,x,y);  //Use the whole window for rendering
}

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

glutInit(&argc, argv);
//we initizlilze the glut. functions
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowPosition(100, 100);
glutInitWindowSize(1000,1000);
glutCreateWindow("cube");
init();
glutDisplayFunc(DrawCube);
glutReshapeFunc(reshape);
//Set the function for the animation.
glutIdleFunc(animation);
glutMainLoop();
return 0;
}
```