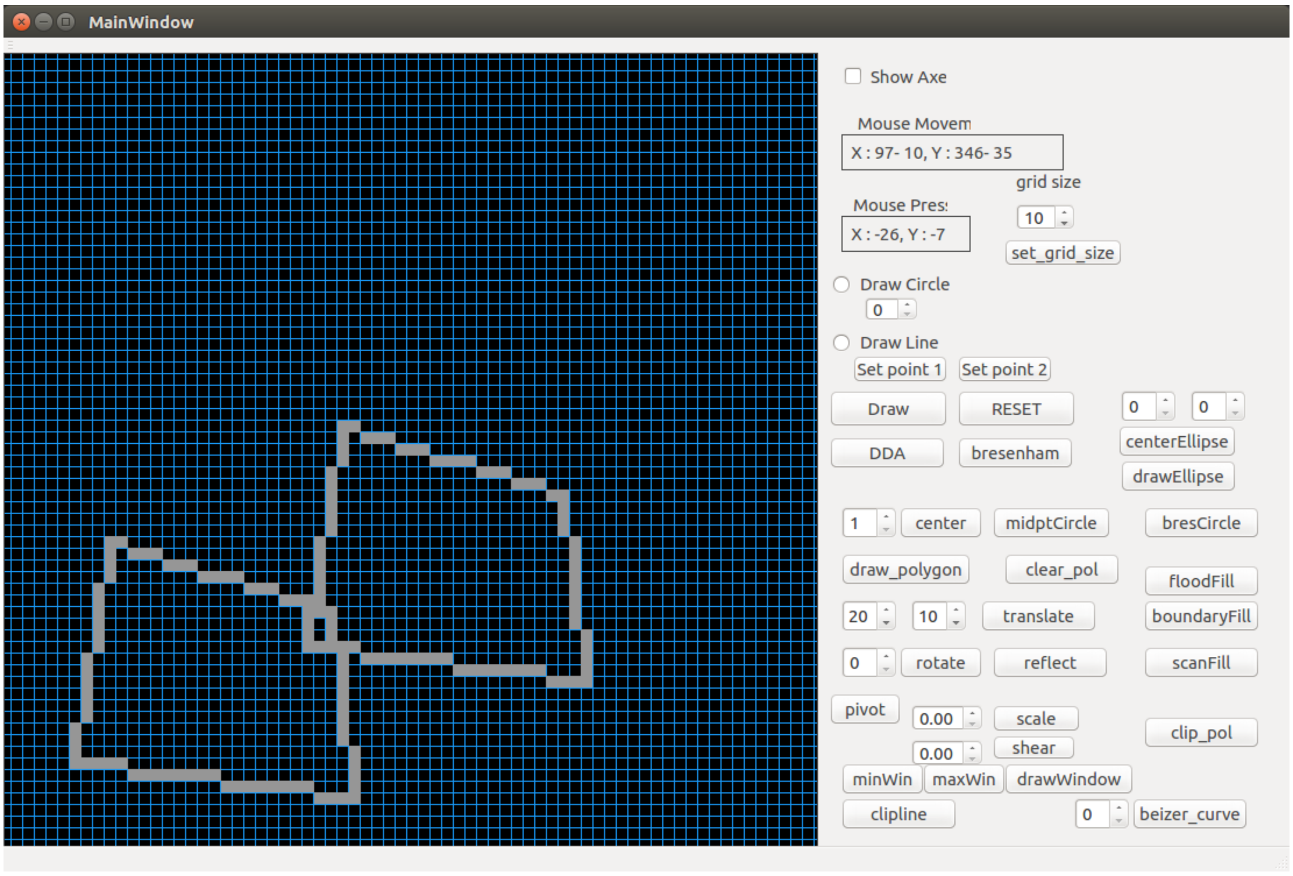
TRANSFORMATION:

To store a polygon:

```
int polygon[100][2];
int polygonindex=0;
void MainWindow::on_polygon_clicked()
{
    int x1=ui->frame->x;
    int y1=ui->frame->y;
    x1=getx(x1);
    y1=gety(y1);
    polygon[polygonindex][0]=x1;
    polygon[polygonindex][1]=y1;
    //qDebug()<<QString::number(x1)+"  "+QString::number(y1)+"  ";
    if(polygonindex>0){
        drawbresenham(x1,y1,polygon[polygonindex-1][0],polygon[polygonindex-
1][1]);
    }
    polygonindex++;
}
void MainWindow::on_clear_polygon_clicked()
{
    polygonindex=0;
}
```

Translation:

```
void MainWindow::on_transform_clicked()
{
    int sx=ui->displace_x->value();
    int sy=ui->displace_y->value();
    int i=0;
    polygon[0][0]+=sx; polygon[0][1]+=sy;
    for(i=1;i<polygonindex;i++){
        polygon[i][0]+=sx;
        polygon[i][1]+=sy;
        drawbresenham(polygon[i][0],polygon[i][1],polygon[i-1][0],
      polygon[i1][1]);
    }
}
```



MainWindow

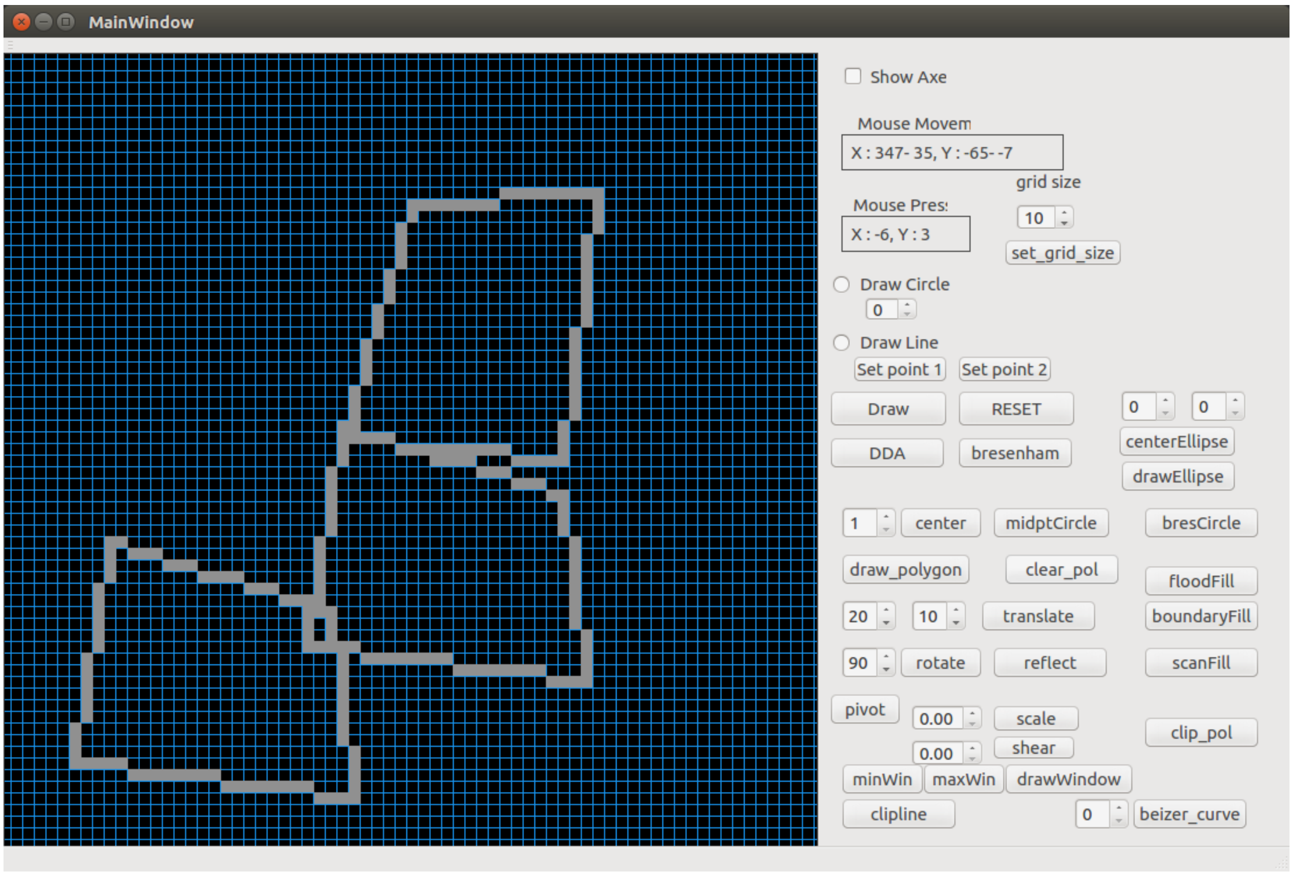
Mouse Movem: X : 97- 10, Y : 346- 35

Mouse Pres: X : -26, Y : -7

grid size: 10

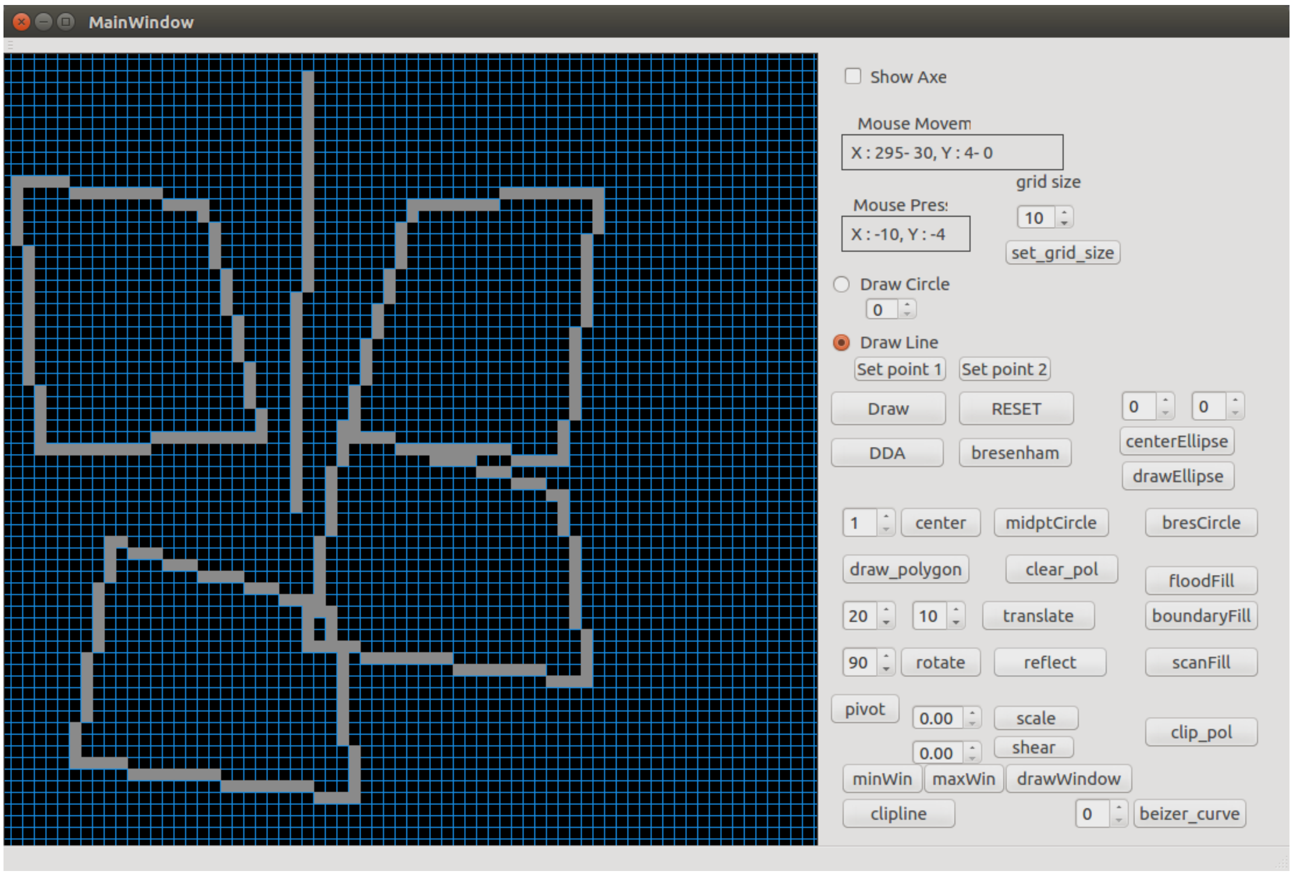
Rotation:

```
void MainWindow::on_set_pivot_clicked()
{
    p3.setX(ui->frame->x);
    p3.setY(ui->frame->y);
}
void MainWindow::on_rotation_clicked()
{
    qreal theta=ui->rotate_angle->value();
    theta=(theta*M_PI/180);
    int i=0;
    int tx,ty;

    int xr=getx(p3.x());
    int yr=gety(p3.y());

    tx=polygon[i][0];
    ty=polygon[i][1];
    polygon[i][0]=tx*cos(theta)-ty*sin(theta)+xr*(1-cos(theta))+yr*sin(theta);
    polygon[i][1]=ty*cos(theta)+tx*sin(theta)+yr*(1-cos(theta))-xr*sin(theta);

    for(i=1;i<polygonindex;i++){
        tx=polygon[i][0];
        ty=polygon[i][1];
        polygon[i][0]=tx*cos(theta)-ty*sin(theta)+xr*(1-
cos(theta))+yr*sin(theta);
        polygon[i][1]=ty*cos(theta)+tx*sin(theta)+yr*(1-cos(theta))-
xr*sin(theta);
        drawbresenham(polygon[i][0],polygon[i][1],polygon[i-1][0],polygon[i-
1][1]);
    }
}
```



Reflection:

```
void MainWindow::on_reflection_clicked()
{
    //if(x2==x1)//
    //float m=(float)
    int px=getx(p1.x());
    int py=gety(p1.y());
    int qx=getx(p2.x());
    int qy=gety(p2.y());

    int i=0;
    int a=polygon[i][0];
    int b=polygon[i][1];

    int A=py-qy;
    int B=qx-px;
    int C= -py*(qx-px)-px*(py-qy);

    float a1=(a*B*B-a*A*A-2*b*A*B-2*A*C)/(float)(A*A+B*B);
    float b1=(b*A*A-b*B*B-2*a*A*B-2*B*C)/(float)(A*A+B*B);


    polygon[i][0]=(a1);
    polygon[i][1]=(b1);
    //qDebug()<<QString::number(a)+" "+QString::number(b)+"
"+QString::number(a1)+" "+QString::number(b1)+" ";
    for(i=1;i<polygonindex;i++){
        a=polygon[i][0];
        b=polygon[i][1];
        A=py-qy;
        B=qx-px;
        C= -py*(qx-px)-px*(py-qy);
        a1=(a*B*B-a*A*A-2*b*A*B-2*A*C)/(float)(A*A+B*B);
        b1=(b*A*A-b*B*B-2*a*A*B-2*B*C)/(float)(A*A+B*B);
        polygon[i][0]=(a1);
        polygon[i][1]=(b1);
        drawbresenham(polygon[i][0],polygon[i][1],polygon[i-1][0], polygon[i-
1][1]);
    }
}
```



Mouse Movem X : 295- 30, Y : 4- 0

Mouse Pres: X : -10, Y : -4

grid size 10

Draw Circle 0

[✓ Draw Line]

0 0

1

20 10

90

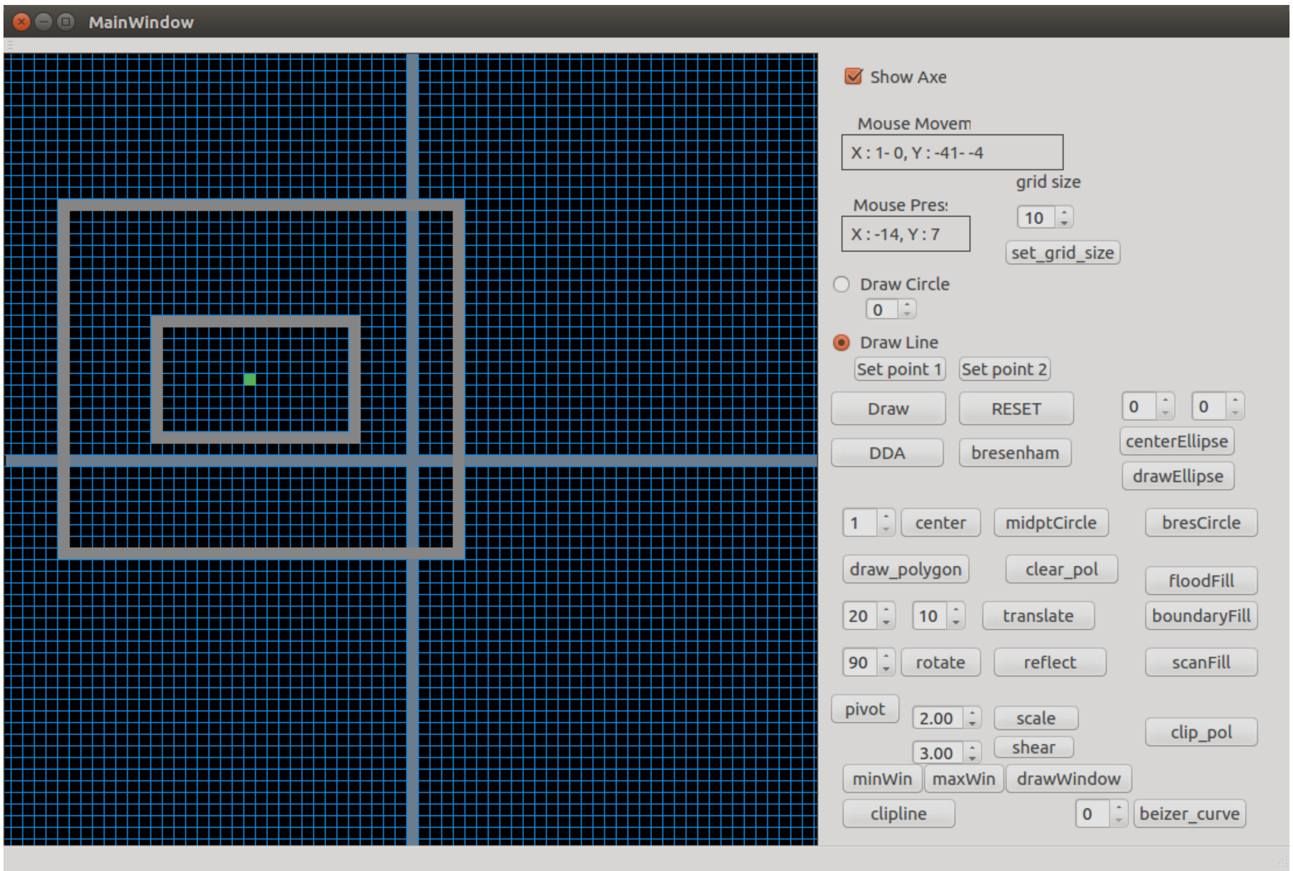
0.00

0.00

0

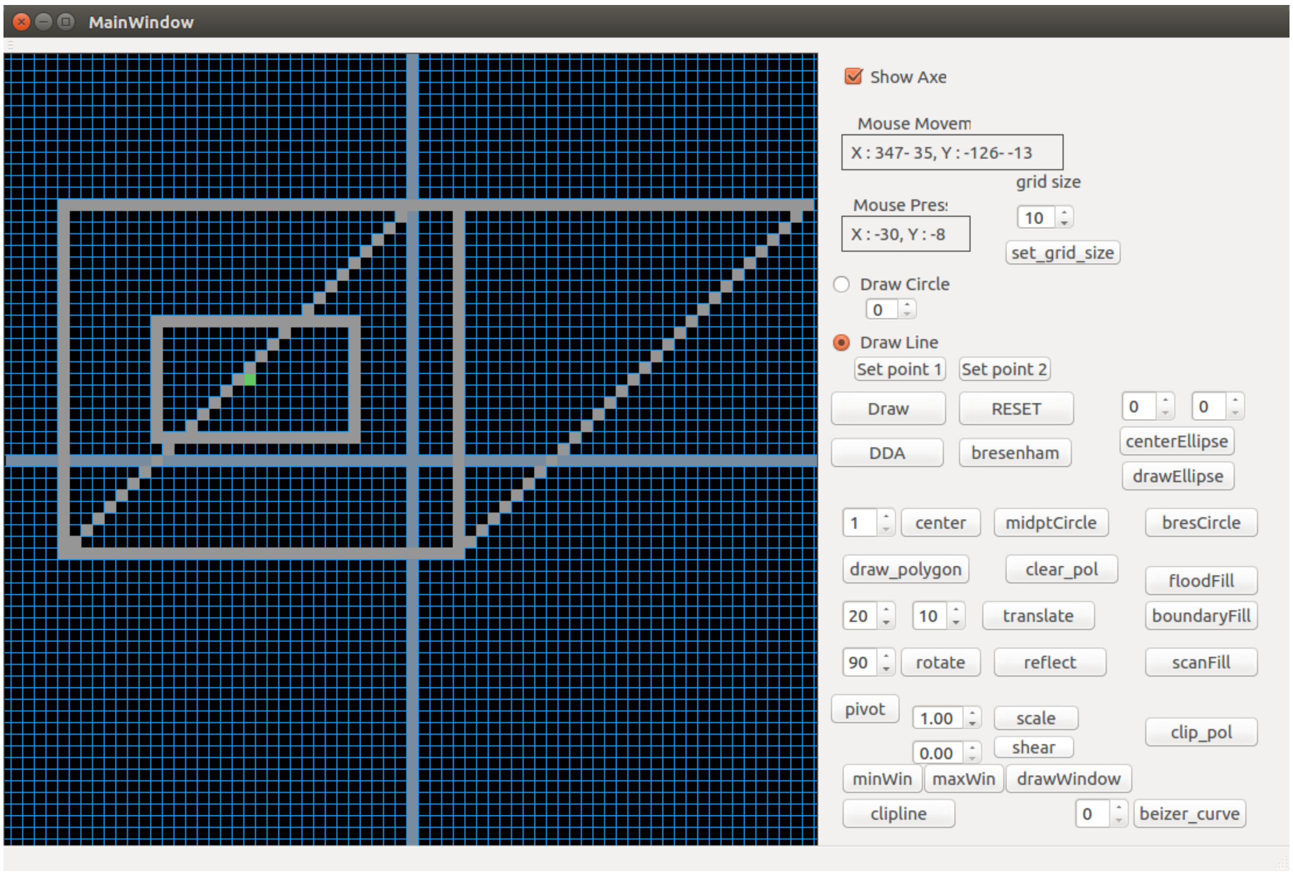
Scaling:

```
void MainWindow::on_scale_clicked()
{
    int xr=getx(p3.x());
    int yr=gety(p3.y());

    float rx=ui->scalex->value();
    float ry=ui->scaley->value();
    int i=0,a,b;
    a=polygon[i][0];
    b=polygon[i][1];
    a-=xr;
    b-=yr;
    a*=rx;
    b*=ry;
    a+=xr;
    b+=yr;
    qDebug()<<QString::number(polygon[i][0])+"
"+QString::number(polygon[i][1])+" "+QString::number(a)+" "+QString::number(b)+"
";
    polygon[i][0]=a;
    polygon[i][1]=b;
    for(i=1;i<polygonindex;i++){
        a=polygon[i][0];
        b=polygon[i][1];
        a-=xr;
        b-=yr;
        a*=rx;
        b*=ry;
        a+=xr;
        b+=yr;
        qDebug()<<QString::number(polygon[i][0])+"
"+QString::number(polygon[i][1])+" "+QString::number(a)+" "+QString::number(b)+"
";
        polygon[i][0]=a;
        polygon[i][1]=b;
        drawbresenham(polygon[i][0],polygon[i][1],polygon[i-1][0],polygon[i-
1][1]);
    }
}
```



Shear:

```
void MainWindow::on_shear_clicked()
{
    int xr=getx(p3.x());
    int yr=gety(p3.y());

    float shx=ui->scalex->value();
    float shy=ui->scaley->value();
    int i=0;
    int a,b,a1,b1;
    a=polygon[i][0];
    b=polygon[i][1];
    a1=a+shx*(b-yr);
    b1=b+shy*(a-xr);
    //qDebug()<<QString::number(polygon[i][0])+"
"+QString::number(polygon[i][1])+" "+QString::number(a)+" "+QString::number(b)+"
";
    polygon[i][0]=a1;
    polygon[i][1]=b1;
    for(i=1;i<polygonindex;i++){
        a=polygon[i][0];
        b=polygon[i][1];
        a1=a+shx*(b-yr);
        b1=b+shy*(a-xr);
        //qDebug()<<QString::number(polygon[i][0])+"
"+QString::number(polygon[i][1])+" "+QString::number(a)+" "+QString::number(b)+"
";
        polygon[i][0]=a1;
        polygon[i][1]=b1;
        drawbresenham(polygon[i][0],polygon[i][1],polygon[i-1][0],polygon[i-
1][1]);
    }

}
```



MainWindow

[✓] Show Axe

Mouse Movem: X : 347- 35, Y : -126- -13

grid size: 10

Mouse Pres: X : -30, Y : -8

Draw Circle: 0

[✓ Draw Line]

0 0

1

20 10

90

1.00

0.00

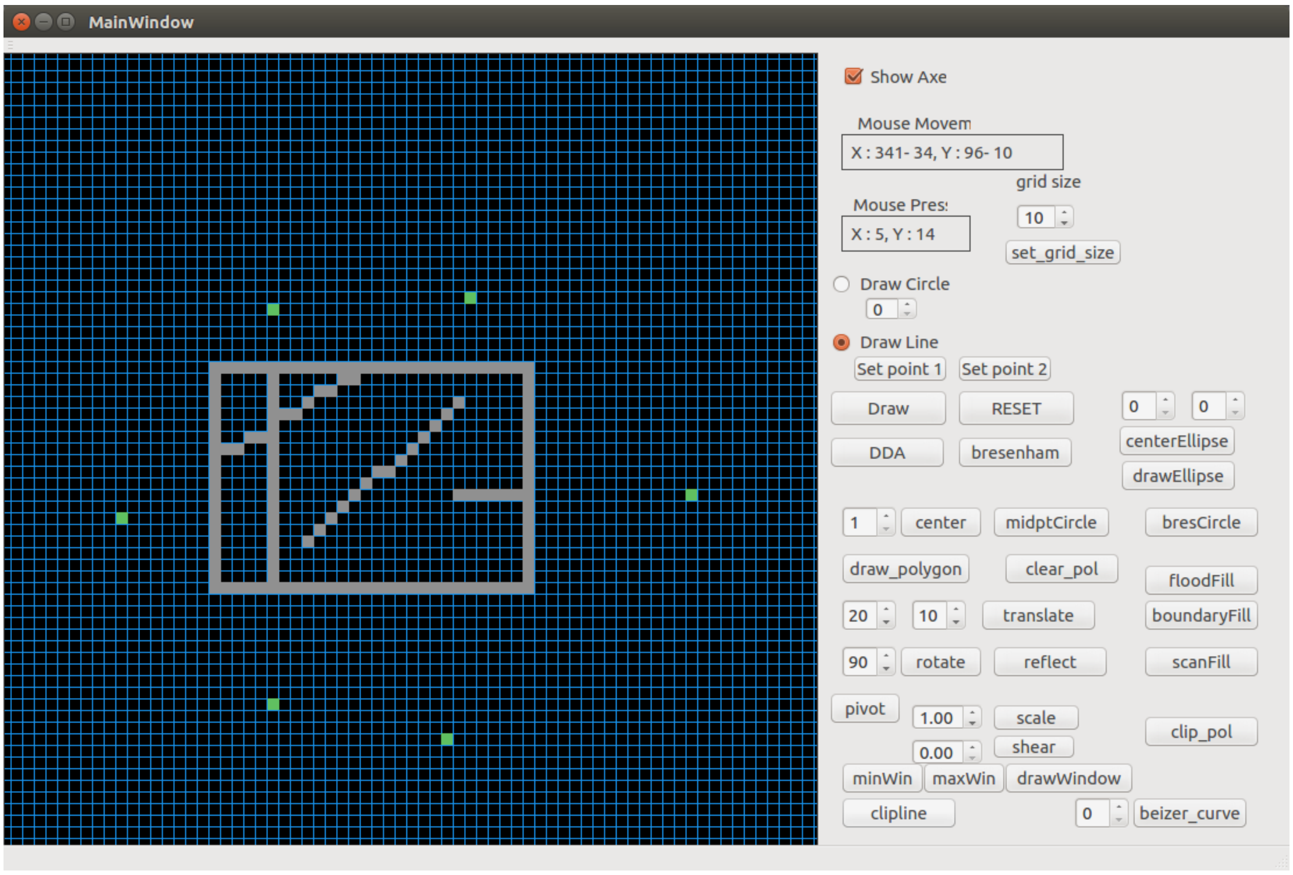
0

CLIPPING:

```
void MainWindow::on_minwin_clicked()
{
    wmin.setX(ui->frame->x);
    wmin.setY(ui->frame->y);
}
void MainWindow::on_maxwin_clicked()
{
    wmax.setX(ui->frame->x);
    wmax.setY(ui->frame->y);
}

void MainWindow::on_draw_window_clicked()
{
    int a=0,b=0,c=0,d=0;
    a=getx(wmax.x());
    b=gety(wmax.y());
    c=getx(wmin.x());
    d=b;
    drawbresenham(a,b,c,d);

    c=a;
    d=gety(wmin.y());
    drawbresenham(a,b,c,d);

    a=getx(wmin.x());
    b=gety(wmin.y());
    drawbresenham(a,b,c,d);

    c=a;
    b=gety(wmax.y());
    drawbresenham(a,b,c,d);

}
```

Line Clipping:

```
#define LEFTEDGE 0x1
#define RIGHTEDGE 0x2
#define BOTTOMEDGE 0x4
#define TOPEDGE 0x8
#define INSIDE(a) (!a)
#define REJECT(a,b) (a&b)
#define ACCEPT(a,b) (!(a|b))
void swapcodes(unsigned char *c1,unsigned char *c2){
    unsigned char temp;
    temp=*c1;
    *c1=*c2;
    *c2=temp;
}

void swappts(QPoint *p1,QPoint *p2){
    QPoint tmp;
    tmp=*p1;
    *p1=*p2;
    *p2=tmp;
}

unsigned char encode(QPoint p,int minx,int miny,int maxx,int maxy){
    unsigned char code=0x0;
    if(p.x()<minx)
        code=code|LEFTEDGE;
    if(p.x()>maxx)
        code=code|RIGHTEDGE;
    if(p.y()<miny)
        code=code|BOTTOMEDGE;
    if(p.y()>maxy)
        code=code|TOPEDGE;
    return code;
}
void MainWindow::on_clipline_clicked()
{
    QPoint tmp;

    int a,b,c,d;
    a=getx(p1.x());
    b=gety(p1.y());
    c=getx(p2.x());
    d=gety(p2.y());

    p3.setX(a);
    p3.setY(b);
    p4.setX(c);
    p4.setY(d);
    int minx=getx(wmin.x());
    int miny=gety(wmin.y());
    int maxx=getx(wmax.x());
    int maxy=gety(wmax.y());
    //-----------------------------------
    unsigned char code1,code2;
    int done=0,draw=0;
    float m;
    qDebug()<<QString::number(minx)+" "+QString::number(miny);
    qDebug()<<QString::number(maxx)+" "+QString::number(maxy);

    qDebug()<<QString::number(p3.x())+" "+QString::number(p3.y());
    while(!done){
        code1=encode(p3,minx,miny,maxx,maxy);
        code2=encode(p4,minx,miny,maxx,maxy);
        qDebug()<<code1<<" "<<code2<<" ";
        if(ACCEPT(code1,code2)){
            done=1;
            draw=1;
            qDebug()<<"accept";
        }
        else if(REJECT(code1,code2)){
            done=1;
            qDebug()<<"reject";
        }
        else{
            if(INSIDE(code1)){
                swappts(&p3,&p4);
                swapcodes(&code1,&code2);
            }
            if(p3.x()!=p4.x())
                m=(p4.y()-p3.y())/(float)(p4.x()-p3.x());
            if(code1&LEFTEDGE){
                int temp=p3.y()+(minx-p3.x())*m;
                p3.setY(temp);
                p3.setX(minx);
            }
            else if(code1&BOTTOMEDGE){
                qDebug()<<"bottom";
                if(p3.x()!=p4.x()){
                    int temp=p3.x()+(maxx-p3.x())/m;
                    p3.setX(temp);
                }
                p3.setY(miny);
            }
            else if(code1&TOPEDGE){
                qDebug()<<"top";
                if(p3.x()!=p4.x()){
                    int temp=p3.x()+(maxy-p3.y())/m;
                    p3.setX(temp);
                }
                p3.setY(maxy);
            }
            else if(code1&RIGHTEDGE){
                qDebug()<<"right";
                int temp=p3.y()+(maxx-p3.x())*m;
                p3.setY(temp);
                p3.setX(maxx);
            }
        }
    }
    if(draw){
        drawbresenham(p3.x(),p3.y(),p4.x(),p4.y());
    }
}
```

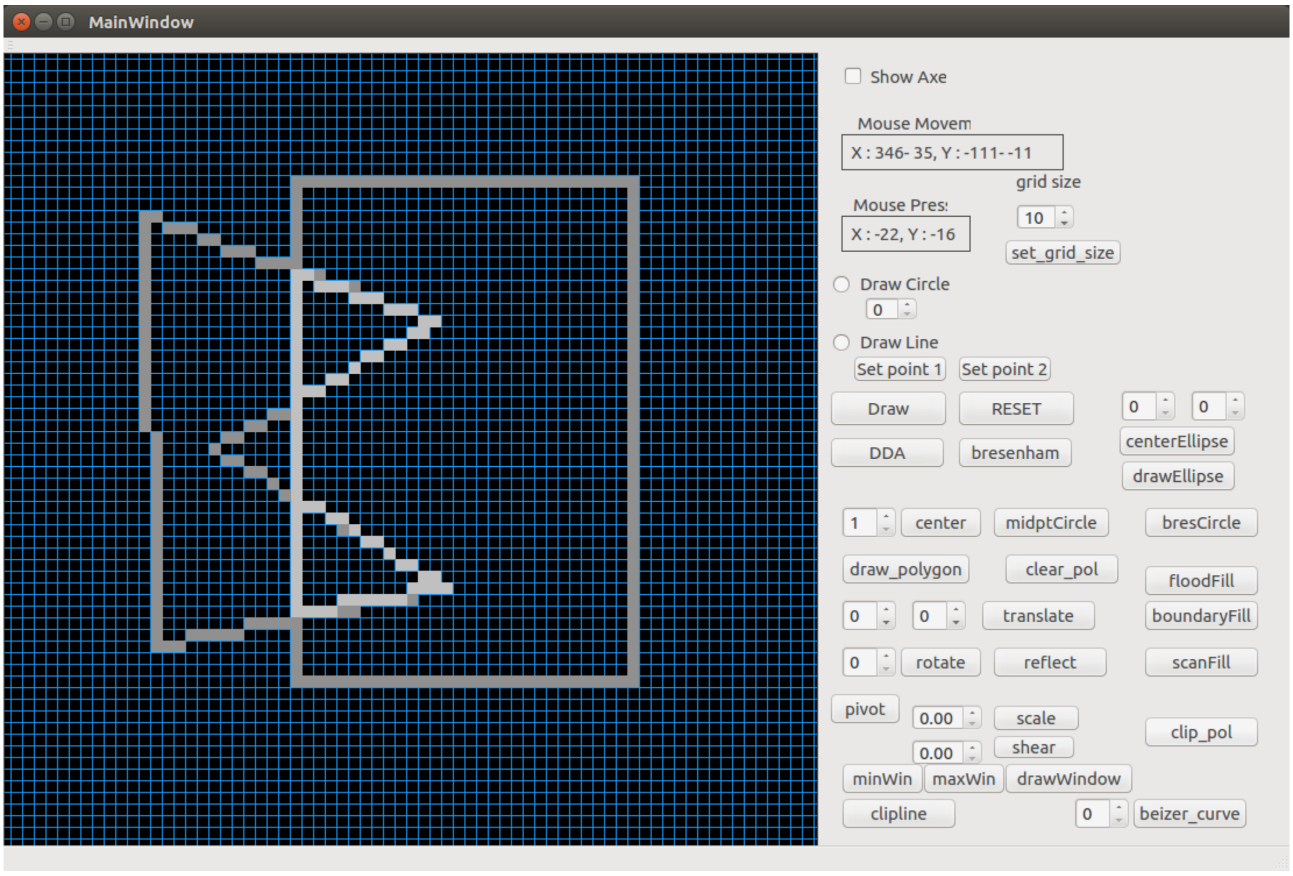


X : 341- 34, Y : 96- 10

X : 5, Y : 14

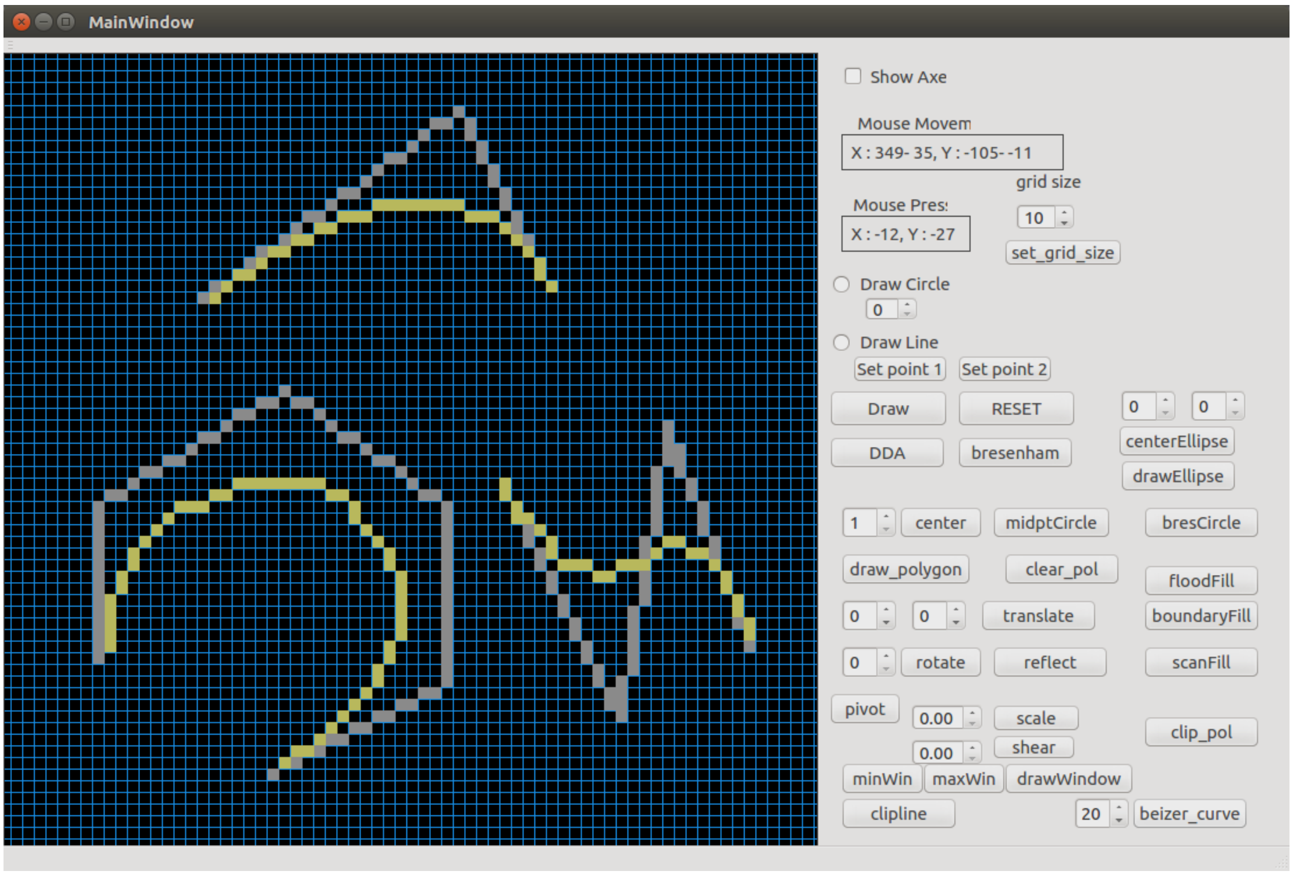
Polygon Clipping:

```
#define MAX_POINTS 20
int x_intersect(int x1, int y1, int x2, int y2,
                int x3, int y3, int x4, int y4)
{
    int num = (x1*y2 - y1*x2) * (x3-x4) -
              (x1-x2) * (x3*y4 - y3*x4);
    int den = (x1-x2) * (y3-y4) - (y1-y2) * (x3-x4);
    return num/den;
}

// Returns y-value of point of intersectipn of
// two lines
int y_intersect(int x1, int y1, int x2, int y2,
                int x3, int y3, int x4, int y4)
{
    int num = (x1*y2 - y1*x2) * (y3-y4) -
              (y1-y2) * (x3*y4 - y3*x4);
    int den = (x1-x2) * (y3-y4) - (y1-y2) * (x3-x4);
    return num/den;
}

// This functions clips all the edges w.r.t one clip
// edge of clipping area
void clip(int poly_points[][2], int &poly_size,
          int x1, int y1, int x2, int y2)
{
    int new_points[MAX_POINTS][2], new_poly_size = 0;

    // (ix,iy),(kx,ky) are the co-ordinate values of
    // the points
    for (int i = 0; i < poly_size; i++)
    {
        // i and k form a line in polygon
        int k = (i+1) % poly_size;
        int ix = poly_points[i][0], iy = poly_points[i][1];
        int kx = poly_points[k][0], ky = poly_points[k][1];

        // Calculating position of first point
        // w.r.t. clipper line
        int i_pos = (x2-x1) * (iy-y1) - (y2-y1) * (ix-x1);

        // Calculating position of second point
        // w.r.t. clipper line
        int k_pos = (x2-x1) * (ky-y1) - (y2-y1) * (kx-x1);

        // Case 1 : When both points are inside
        if (i_pos < 0  && k_pos < 0)
        {
            //Only second point is added
            new_points[new_poly_size][0] = kx;
            new_points[new_poly_size][1] = ky;
            new_poly_size++;
        }

        // Case 2: When only first point is outside
        else if (i_pos >= 0  && k_pos < 0)
        {
            // Point of intersection with edge
            // and the second point is added
            new_points[new_poly_size][0] = x_intersect(x1,
                                  y1, x2, y2, ix, iy, kx, ky);
            new_points[new_poly_size][1] = y_intersect(x1,
                                  y1, x2, y2, ix, iy, kx, ky);
            new_poly_size++;

            new_points[new_poly_size][0] = kx;
            new_points[new_poly_size][1] = ky;
            new_poly_size++;
        }

        // Case 3: When only second point is outside
        else if (i_pos < 0  && k_pos >= 0)
        {
            new_points[new_poly_size][0] = x_intersect(x1,
                                  y1, x2, y2, ix, iy, kx, ky);
            new_points[new_poly_size][1] = y_intersect(x1,
                                  y1, x2, y2, ix, iy, kx, ky);
            new_poly_size++;
        }
        // Case 4: When both points are outside
        else{
            //No points are added
        }
    }
    poly_size = new_poly_size;
    for (int i = 0; i < poly_size; i++)
    {
        poly_points[i][0] = new_points[i][0];
        poly_points[i][1] = new_points[i][1];
    }
}

void suthHodgClip(int poly_points[][2], int &poly_size,
                  int clipper_points[][2], int clipper_size)
{
    for (int i=0; i<clipper_size; i++)
    {
        int k = (i+1) % clipper_size;
        clip(poly_points, poly_size, clipper_points[i][0],
             clipper_points[i][1], clipper_points[k][0],
             clipper_points[k][1]);
    }
    for (int i=0; i < poly_size; i++){
    }
}

void MainWindow::on_clip_pol_clicked()
{
    int i;
    int clipper_size=4;
    int clipper_points[4][2];
    clipper_points[0][0]= getx(wmin.x()); clipper_points[0][1]=gety(wmin.y()) ;
    clipper_points[1][0]= getx(wmin.x()); clipper_points[1][1]=gety(wmax.y());
    clipper_points[2][0]= getx(wmax.x()); clipper_points[2][1]=gety(wmax.y());
    clipper_points[3][0]= getx(wmax.x()); clipper_points[3][1]=gety(wmin.y());

    int poly_size=polygonindex;
    int pol_pts[poly_size][2];
    for(i=0;i<polygonindex;i++){
        pol_pts[i][0]= polygon[i][0];
        pol_pts[i][1]= polygon[i][1];
    }
    suthHodgClip(pol_pts,poly_size,clipper_points,clipper_size);
    for(i=1;i<poly_size;i++){
        drawbresenham(pol_pts[i-1][0],pol_pts[i-1][1],pol_pts[i][0],
pol_pts[i][1],200,200,200);
    }
    drawbresenham(pol_pts[i-1][0],pol_pts[i-1][1],pol_pts[0][0],pol_pts[0][1],
200,200,200);
}
```



Bezier Curve:

```
int bincoeff(int n, int k)
{
    int res = 1;
    if ( k > n - k )
        k = n - k;
    for (int i = 0; i < k; ++i){
        res *= (n - i);
        res /= (i + 1);
    }
    return res;
}
void computecoeffs(int n,int *c){
    int k,i;
    for(k=0;k<=n;k++){
        c[k]=1;
        c[k]=bincoeff(n,k);
    }
}
void computepoint(float u,int points[][2],int npts,int *c,int outpts[][2],int
index){
    int k,n=npts-1;
    float blend,a,b;
    a=0.0;b=0.0;
    for(k=0;k<npts;k++){
        blend=c[k]*pow(u,k)*pow(1-u,n-k);
        a+=points[k][0]*blend;
        b+=points[k][1]*blend;
    }
    outpts[index][0]=ROUND(a);
    outpts[index][1]=ROUND(b);
}


void MainWindow::on_beizer_clicked()
{
    int m= ui->curve_pts->value();
    int outpts[m+1][2];
    int npts=polygonindex;
    int points[npts][2];
    int i;
    for(i=0;i<npts;i++){
        points[i][0]=polygon[i][0];
        points[i][1]=polygon[i][1];
    }
    int *c=(int*)malloc(npts*sizeof(int));
    computecoeffs(npts-1,c);
    for(i=0;i<=m;i++){
       computepoint(i/(float)m,points,npts,c,outpts,i);
        //qDebug()<<QString::number(outpts[i][0])<<"
"<<QString::number(outpts[i][1]);
    }
    free(c);
    for(i=1;i<=m;i++){
        drawbresenham(outpts[i-1][0],outpts[i-
1][1],outpts[i][0],outpts[i][1],200,200,100);
    }
}
```


MainWindow
Show Axe
Mouse Movem
X : 349- 35, Y : -105- -11
grid size
Mouse Pres:
X : -12, Y : -27
10
set_grid_size
Draw Circle
0
Draw Line
Set point 1
Set point 2
Draw
RESET
0
0
DDA
bresenham
centerEllipse
drawEllipse
1
center
midptCircle
bresCircle
draw_polygon
clear_pol
floodFill
0
0
translate
boundaryFill
0
rotate
reflect
scanFill
pivot
0.00
scale
clip_pol
0.00
shear
minWin
maxWin
drawWindow
clipline
20
beizer_curve