ASSIGNMENT-4

NAME-ANUBHAV ANAND

ENROLLMENT NUMBER-2020CSB102

SUBJECT-ASSIGNMENT-4 OF COMPUTER GRAPHICS

G-SUITE ID-2020CSB102.anubhav@students.iiests.ac.in

Q1.Ans-DIAGRAM.JAVA-

Code-

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class Diagram extends Applet implements
ActionListener,MouseWheelListener{
    //It is for generating a rectangle corresponding to a particular
point in cartesian coordinate syatem
        int gap = 2;
        int temp=0;
        int temp1=0;
        int temp2=0;
        int temp3=0;
        int temp5=0;
        int temp6=0;
        int temp7=0;
        int temp8=0;
        Button button1 = new Button("+");
        Button button2 = new Button("-");
        Button button = new Button("Teeth");
        Button button3=new Button("Ear");
        Button button4=new Button("Circles");
        Button button5=new Button("Tail");
        Button button6=new Button("Leg");
        Button button7=new Button("Beek");
        Button button8=new Button("Hair");
        Button button9=new Button("Fingers");
    //It is for initialisation purpose
    public void init(){
        add(button1);
        add(button2);
        add(button);
        add(button3);
        add(button4);
        add(button5);
        add(button6);
        add(button7);
        add(button8);
        add(button9);
        button1.setBackground(Color.white);
        button2.setBackground(Color.white);
        button.setBackground(Color.white);
```

```
        button3.setBackground(Color.white);
        button4.setBackground(Color.white);
        button5.setBackground(Color.white);
        button6.setBackground(Color.white);
        button7.setBackground(Color.white);
        button8.setBackground(Color.white);
        button9.setBackground(Color.white);

        button1.addActionListener(this);
        button2.addActionListener(this);
        button.addActionListener(this);
        button3.addActionListener(this);
        button4.addActionListener(this);
        button5.addActionListener(this);
        button6.addActionListener(this);
        button7.addActionListener(this);
        button8.addActionListener(this);
        button9.addActionListener(this);
        addMouseWheelListener(this);
        setForeground(Color.red);
        setBackground(Color.black);
    }
    //it is for implementing button function
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == button1){
         gap+=gap+gap/10;
         repaint();
        }
        if(e.getSource()==button2)
        {
             gap-=gap/10;
             repaint();
        }
        if(e.getSource()== button)
        {
          if(temp==0)
            temp=1;
          else
            temp=0;

         repaint();
        }
        if(e.getSource()==button3)
        {
            if(temp1==0)
              temp1=1;
            else
```

```
            temp1=0;
            repaint();
        }
        if(e.getSource()==button4)
        {
            if(temp2==0)
              temp2=1;
            else
              temp2=0;
              repaint();
        }
           if(e.getSource()==button5)
        {
            if(temp3==0)
              temp3=1;
            else if(temp3==1)
              temp3=2;
            else
            temp3=0;
              repaint();
        }
         if(e.getSource()==button6)
        {
            if(temp5==0)
              temp5=1;
            else
             temp5=0;
             repaint();
        }
        if(e.getSource()==button7)
        {
            if(temp6==0)
              temp6=1;
            else
             temp6=0;
             repaint();
        }
        if(e.getSource()==button8)
        {
            if(temp7==0)
              temp7=1;
            else
             temp7=0;
             repaint();
        }
       if(e.getSource()==button9)
        {
            if(temp8==0)
```

```
                temp8=1;
            else
                temp8=0;
                repaint();
        }


    }
    public void plotPoint(Graphics g,int x,int y,Color c)
    {
        g.setColor(c);
        g.fillRect(
            (getX()+getWidth())/2+(x*gap)-(gap/2),
            (getY()+getHeight())/2-(y*gap)-(gap/2),
            3*gap,3*gap
        );
    }
    public int slope(int x1,int x2,int y1,int y2)
    {
        int x=x2-x1;
        int y=y2-y1;
        int m=y/x;
        return m;
    }
    //Round function
    public int round(float n) {
    if (n - (int)n < 0.5)
        return (int)n;
    return (int)(n + 1);
}
//Circle Drawing Algorithm
 public void Circles(Graphics g,int radius,int x1,int y1)
    {
        int x=0;
        int y=radius;
        double p=1.25-radius;
        plotPoint(g,x+x1,y+y1,Color.green);
        plotPoint(g,x+x1,-y+y1,Color.green);
        plotPoint(g,-x+x1,y+y1,Color.green);
        plotPoint(g,-x+x1,-y+y1,Color.green);
        while(x<y)
        {
            if(p<0)
                {
                    x=x+1;
                    p=p+2*x+1;

                }
```

```
            else
            {
                x=x+1;
                y=y-1;
                p=p+2*x+1-2*y;
            }
             plotPoint(g,x+x1,y+y1,Color.green);
             plotPoint(g,y+x1,x+y1,Color.green);
             plotPoint(g,x+x1,-y+y1,Color.green);
             plotPoint(g,-x+x1,y+y1,Color.green);
             plotPoint(g,y+x1,-x+y1,Color.green);
             plotPoint(g,-y+x1,x+y1,Color.green);
             plotPoint(g,-x+x1,-y+y1,Color.green);
             plotPoint(g,-y+x1,-x+y1,Color.green);
        }
    }



//Line Drawing Algorithm
 //function to return line
    public void DDALine(Graphics g,int x0, int y0, int x1, int y1) {

      //calculate dx and dy
      int dx = x1 - x0;
    int dy= y1 - y0;

    int step;

      //if dx > dy we will take step as dx
      //else we will take step as dy to draw the complete line
    if (Math.abs(dx) > Math.abs(dy))
        step = Math.abs(dx);
    else
        step = Math.abs(dy);

      //calculate x-increment and y-increment for each step
    float x_incr = (float)dx / step;
    float y_incr = (float)dy / step;

      //take the initial points as x and y
    float x = x0;
    float y = y0;

    for (int i = 0; i < step; i ++) {
        //putpixel(round(x), round(y), WHITE);
       plotPoint(g, round(x), round(y), Color.green);
        x += x_incr;
        y += y_incr;
```

```
        //delay(10);
    }
}


//Ellipse drawing algorithm
public void midptellipse(Graphics g,float rx, float ry,
                         float xc, float yc,Double degree)
{

    float dx, dy, d1, d2, x, y;
    x = 0;
    y = ry;
    double radian=Math.toRadians(degree);
    // Initial decision parameter of region 1
    d1 = (ry * ry) - (rx * rx * ry) +
                     (0.25f * rx * rx);
    dx = 2 * ry * ry * x;
    dy = 2 * rx * rx * y;
    // DecimalFormat df = new DecimalFormat("#,###,##0.00000");

    // For region 1
    while (dx < dy)
    {

     plotPoint(g,((int)((x)*Math.cos(radian)+xc-
(y)*Math.sin(radian))),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(radi
an))),Color.green);
    plotPoint(g,((int)((-x)*Math.cos(radian)+xc-
(y)*Math.sin(radian))),((int)((-
x)*Math.sin(radian)+yc+(y)*Math.cos(radian))),Color.green);
    plotPoint(g,((int)((x)*Math.cos(radian)+xc-(-
y)*Math.sin(radian))),((int)((x)*Math.sin(radian)+yc+(-
y)*Math.cos(radian))),Color.green);
   plotPoint(g,((int)((-x)*Math.cos(radian)+xc-(-
y)*Math.sin(radian))),((int)((-x)*Math.sin(radian)+yc+(-
y)*Math.cos(radian))),Color.green);
    //  plotPoint(g,((int)((-x)*Math.cos(radian)+xc-
(y)*Math.sin(radian))),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(radi
an))),Color.red);
    //  plotPoint(g,((int)((x)*Math.cos(radian)+xc-(-
y)*Math.sin(radian))),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(radia
n))),Color.red);
    //  plotPoint(g,((int)((-x)*Math.cos(radian)+xc-(-
y)*Math.sin(radian))),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(radia
n))),Color.red);

        if (d1 < 0)
        {
```

```
            x++;
            dx = dx + (2 * ry * ry);
            d1 = d1 + dx + (ry * ry);
        }
        else
        {
            x++;
            y--;
            dx = dx + (2 * ry * ry);
            dy = dy - (2 * rx * rx);
            d1 = d1 + dx - dy + (ry * ry);
        }
    }

    // Decision parameter of region 2
    d2 = ((ry * ry) * ((x + 0.5f) * (x + 0.5f)))
        + ((rx * rx) * ((y - 1) * (y - 1)))
        - (rx * rx * ry * ry);

    // Plotting points of region 2
    while (y >= 0) {

        // printing points based on 4-way symmetry
        // plotPoint(g,(int)(x+xc),(int)(y+yc),Color.red);
        // plotPoint(g,(int)(-x+xc),(int)(y+yc),Color.red);
        // plotPoint(g,(int)(x+xc),(int)(-y+yc),Color.red);
        // plotPoint(g,(int)(-x+xc),(int)(-y+yc),Color.red);

    plotPoint(g,((int)((x)*Math.cos(radian)-
(y)*Math.sin(radian)+xc)),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(r
adian))),Color.green);
    plotPoint(g,((int)((-x)*Math.cos(radian)-
(y)*Math.sin(radian)+xc)),((int)((-
x)*Math.sin(radian)+yc+(y)*Math.cos(radian))),Color.green);
    plotPoint(g,((int)((x)*Math.cos(radian)-(-
y)*Math.sin(radian)+xc)),((int)((x)*Math.sin(radian)+yc+(-
y)*Math.cos(radian))),Color.green);
    plotPoint(g,((int)((-x)*Math.cos(radian)-(-
y)*Math.sin(radian)+xc)),((int)((-x)*Math.sin(radian)+yc+(-
y)*Math.cos(radian))),Color.green);

    //  plotPoint(g,((int)((-x)*Math.cos(radian)-
(y)*Math.sin(radian)+xc)),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(r
adian))),Color.red);
    //  plotPoint(g,((int)((x)*Math.cos(radian)-(-
y)*Math.sin(radian)+xc)),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(ra
dian))),Color.red);
```

```
    //  plotPoint(g,((int)((-x)*Math.cos(radian)-(-
y)*Math.sin(radian)+xc)),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(ra
dian))),Color.red);

        if (d2 > 0) {
            y--;
            dy = dy - (2 * rx * rx);
            d2 = d2 + (rx * rx) - dy;
        }
        else {
            y--;
            x++;
            dx = dx + (2 * ry * ry);
            dy = dy - (2 * rx * rx);
            d2 = d2 + dx - dy + (rx * rx);
        }
    }
}
public void paintGrid(Graphics g,int gap,int originx,int originy)
    {
        g.setColor(Color.black);

        for(int i = gap;i<=getWidth();i+=gap)
        {
            g.drawLine(originx+i, originy-getHeight()/2, originx+i,
originy+getHeight()/2);
            g.drawLine(originx-i, originy-getHeight()/2, originx-i,
originy+getHeight()/2);
        }
        for(int i = gap;i<=getHeight();i+=gap)
        {
            g.drawLine(originx-getWidth()/2, originy+i,
originx+getWidth()/2, originy+i);
            g.drawLine(originx-getWidth()/2, originy-i,
originx+getWidth()/2, originy-i);

        }
    }









    //It is for mouse wheel operation
    public void mouseWheelMoved(MouseWheelEvent e)
    {
```

```
        int z=e.getWheelRotation();
        gap+=z;
        repaint();
    }
    //function for making spotted cirecles
    public void makespot(Graphics g,int x)
    {
        if(temp2==0)
        {
          Circles(g,6,44+x,50);
        Circles(g,6,24+x,50);
        Circles(g,6,4+x,50);
        Circles(g,6,-14+x,50);
        Circles(g,6,-34+x,50);
        Circles(g,6,-54+x,50);
        Circles(g,6,-74+x,50);

        Circles(g,6,44+x,80);
        Circles(g,6,24+x,80);
        Circles(g,6,4+x,80);
        Circles(g,6,-14+x,80);
        Circles(g,6,-34+x,80);
        Circles(g,6,-54+x,80);
        Circles(g,6,-74+x,80);
        Circles(g,6,-94+x,80);

        Circles(g,6,44+x,110);
        Circles(g,6,24+x,110);
        Circles(g,6,4+x,110);
        Circles(g,6,-14+x,110);
        Circles(g,6,-34+x,110);
        Circles(g,6,-54+x,110);
        Circles(g,6,-74+x,110);
        Circles(g,6,-94+x,110);


        Circles(g,6,24+x,140);
        Circles(g,6,4+x,140);
        Circles(g,6,-14+x,140);
        Circles(g,6,-34+x,140);
        Circles(g,6,-54+x,140);
        Circles(g,6,-74+x,140);
        Circles(g,6,-94+x,140);

        Circles(g,6,-14+x,20);
        Circles(g,6,-34+x,20);
        Circles(g,6,-54+x,20);
```

```
        Circles(g,6,-34+x,170);
        Circles(g,6,-14+x,170);
        Circles(g,6,-54+x,170);
        }
    }
    //function to make hairy tail
    public void makehairytail(Graphics g,int x)
    {
        if(temp3==2)
        {
           DDALine(g,50+x,35,120+x,90);
            DDALine(g,50+x,35,120+x,60);
        DDALine(g,50+x,35,130+x,70);
        DDALine(g,50+x,35,120+x,50);
        DDALine(g,50+x,35,130+x,40);
        DDALine(g,50+x,35,120+x,30);
        DDALine(g,50+x,35,130+x,20);
        }
    }
    //function to make ellipse tail
    public void maketailellipse(Graphics g,int x)
    {
        if(temp3==0)

midptellipse(g,(float)45,(float)10,(float)90+x,(float)50,(double)20);
    }
    //function to make hair in body
    public void makehair(Graphics g)
    {
        if(temp7==1)
        {
            for(int y=160;y>=20;y-=25)
            {
                if(y==180 || y==20)
                {
                    for(int i=-50;i<-10;i+=30)
                    {
                        DDALine(g,i,y,i+13,y-25);
                    }
                }
                else
                {
                    for(int i=-100;i<50;i+=30)
                    {
                        DDALine(g,i,y,i+13,y-25);
                    }

                }
```

```
            }
        }
    }
    //function for making legspot
    public void maketailtri(Graphics g,int x)
    {
        if(temp3==1)
        {
           DDALine(g,50+x,35,120+x,70);
        DDALine(g,50+x,35,130+x,40);
        DDALine(g,120+x,70,130+x,40);
        }
    }
    //function for making triangle
    public void maketriangle(Graphics g,int x)
    {
            if(temp1==0)
            {
                DDALine(g,-70+x,240,-60+x,270);
                DDALine(g,-70+x,240,-38+x,235);
                DDALine(g,-60+x,270,-38+x,235);
            }

    }
    //function for making leg
    public void makeleg(Graphics g,int x)
    {
        if(temp5==1)
        {
            //front upper
         Circles(g,4,-50+x,-70);
        Circles(g,4,-30+x,-70);
        Circles(g,4,-15+x,-49);
        Circles(g,4,-30+x,-31);
        Circles(g,4,-35+x,-53);
        Circles(g,4,-5+x,-10);
        Circles(g,4,-8+x,-29);
        //front lower
        Circles(g,4,-85+x,-130);
        Circles(g,4,-70+x,-131);
        Circles(g,4,-75+x,-153);
        Circles(g,4,-73+x,-110);
        Circles(g,4,-55+x,-129);
        // //back upper
        Circles(g,4,70+x,-70);
        Circles(g,4,50+x,-70);
        Circles(g,4,35+x,-49);
        Circles(g,4,50+x,-31);
```

```
        Circles(g,4,55+x,-53);
        Circles(g,4,25+x,-10);
        Circles(g,4,28+x,-29);
        // //back lower
        Circles(g,4,85+x,-130);
        Circles(g,4,70+x,-131);
        Circles(g,4,75+x,-153);
        Circles(g,4,73+x,-110);
        }
    }
    //function for making circle of ear
    public void makeear(Graphics g,int x)
    {
        if(temp1==1)
          Circles(g,20,-51+x,249);
    }
    //function for beek
    public void makebeek(Graphics g,int x)
    {
        if(temp6==0)
        {
            DDALine(g,-140+x,210,-140+x,230);
            DDALine(g,-140+x,210,-200+x,210);
            DDALine(g,-140+x,230,-200+x,210);
            DDALine(g,-140+x,190,-140+x,170);
            DDALine(g,-140+x,190,-200+x,190);
            DDALine(g,-140+x,170,-200+x,190);
        }
        else
        {
            DDALine(g,-140+x,210,-140+x,230);
            DDALine(g,-140+x,210,-230+x,210);
            DDALine(g,-140+x,230,-230+x,210);
            DDALine(g,-140+x,190,-140+x,170);
            DDALine(g,-140+x,190,-230+x,190);
            DDALine(g,-140+x,170,-230+x,190);

        }
    }
    //function for teeth making
    public void maketeeth(Graphics g,int x)
    {
        if(temp==1 && temp6==0){
            DDALine(g,-200+x,190,-200+x,197);
              DDALine(g,-180+x,190,-180+x,197);
              DDALine(g,-160+x,190,-160+x,197);
              DDALine(g,-150+x,190,-150+x,197);
              DDALine(g,-160+x,190,-160+x,197);
```

```
            DDALine(g,-170+x,190,-170+x,197);
            DDALine(g,-180+x,190,-180+x,197);
            DDALine(g,-190+x,190,-190+x,197);

            DDALine(g,-201+x,190,-201+x,197);
            DDALine(g,-181+x,190,-181+x,197);
            DDALine(g,-161+x,190,-161+x,197);
            DDALine(g,-151+x,190,-151+x,197);
            DDALine(g,-161+x,190,-161+x,197);
            DDALine(g,-171+x,190,-171+x,197);
            DDALine(g,-181+x,190,-181+x,197);
            DDALine(g,-191+x,190,-191+x,197);

            DDALine(g,-200+x,210,-200+x,203);
            DDALine(g,-180+x,210,-180+x,203);
            DDALine(g,-160+x,210,-160+x,203);
            DDALine(g,-150+x,210,-150+x,203);
            DDALine(g,-160+x,210,-160+x,203);
            DDALine(g,-170+x,210,-170+x,203);
            DDALine(g,-180+x,210,-180+x,203);
            DDALine(g,-190+x,210,-190+x,203);

            DDALine(g,-201+x,210,-201+x,203);
            DDALine(g,-181+x,210,-181+x,203);
            DDALine(g,-161+x,210,-161+x,203);
            DDALine(g,-151+x,210,-151+x,203);
            DDALine(g,-161+x,210,-160+x,203);
            DDALine(g,-171+x,210,-171+x,203);
            DDALine(g,-181+x,210,-181+x,203);
            DDALine(g,-191+x,210,-191+x,203);
        }
        else if(temp==0 && temp6==1)
        {
            DDALine(g,-200+x,190,-200+x,197);
            DDALine(g,-180+x,190,-180+x,197);
            DDALine(g,-160+x,190,-160+x,197);
            DDALine(g,-150+x,190,-150+x,197);
            DDALine(g,-160+x,190,-160+x,197);
            DDALine(g,-170+x,190,-170+x,197);
            DDALine(g,-180+x,190,-180+x,197);
            DDALine(g,-190+x,190,-190+x,197);
            DDALine(g,-230+x,190,-230+x,197);
            DDALine(g,-210+x,190,-210+x,197);
            DDALine(g,-220+x,190,-220+x,197);

            DDALine(g,-201+x,190,-201+x,197);
            DDALine(g,-181+x,190,-181+x,197);
            DDALine(g,-161+x,190,-161+x,197);
```

```
            DDALine(g,-151+x,190,-151+x,197);
            DDALine(g,-161+x,190,-161+x,197);
            DDALine(g,-171+x,190,-171+x,197);
            DDALine(g,-181+x,190,-181+x,197);
            DDALine(g,-191+x,190,-191+x,197);
            DDALine(g,-231+x,190,-231+x,197);
            DDALine(g,-211+x,190,-211+x,197);
            DDALine(g,-221+x,190,-221+x,197);

            DDALine(g,-200+x,210,-200+x,203);
            DDALine(g,-180+x,210,-180+x,203);
            DDALine(g,-160+x,210,-160+x,203);
            DDALine(g,-150+x,210,-150+x,203);
            DDALine(g,-160+x,210,-160+x,203);
            DDALine(g,-170+x,210,-170+x,203);
            DDALine(g,-180+x,210,-180+x,203);
            DDALine(g,-190+x,210,-190+x,203);
            DDALine(g,-210+x,210,-210+x,203);
            DDALine(g,-220+x,210,-220+x,203);
            DDALine(g,-230+x,210,-230+x,203);

            DDALine(g,-201+x,210,-201+x,203);
            DDALine(g,-181+x,210,-181+x,203);
            DDALine(g,-161+x,210,-161+x,203);
            DDALine(g,-151+x,210,-151+x,203);
            DDALine(g,-161+x,210,-160+x,203);
            DDALine(g,-171+x,210,-171+x,203);
            DDALine(g,-181+x,210,-181+x,203);
            DDALine(g,-191+x,210,-191+x,203);
            DDALine(g,-211+x,210,-211+x,203);
            DDALine(g,-221+x,210,-221+x,203);
            DDALine(g,-231+x,210,-231+x,203);
        }
    }
    //function to make fingers
    public void makefingers(Graphics g)
    {
        if(temp8==1)
        {
        midptellipse(g,(float)13,(float)3,
                        (float)-245,(float)50,(double)50);
            midptellipse(g,(float)13,(float)3,
                        (float)-253,(float)63,(double)40);
            midptellipse(g,(float)13,(float)3,
                        (float)-235,(float)40,(double)50);
            midptellipse(g,(float)13,(float)3,
                        (float)-261,(float)76,(double)30);
            midptellipse(g,(float)10,(float)3,
```

```
                        (float)-251,(float)90,(double)110);



            midptellipse(g,(float)13,(float)3,
                        (float)-245-30,(float)50+65,(double)30);
            midptellipse(g,(float)13,(float)3,
                        (float)-253-30,(float)63+65,(double)30);
            midptellipse(g,(float)13,(float)3,
                        (float)-235-30,(float)40+65,(double)40);
            midptellipse(g,(float)13,(float)3,
                        (float)-261-23,(float)76+65,(double)20);
            midptellipse(g,(float)10,(float)3,
                        (float)-251-23,(float)90+65,(double)100);
        }
    }

    //making clone
    public void clone(Graphics g,int x)
    {
        Circles(g,50,-100+x,200);
            Circles(g,10,-100+x,200);

            Circles(g,50,-100+x,200);
            Circles(g,10,-100+x,200);

            midptellipse(g,(float)100,(float)80,(float)-
30+x,(float)90,(double)135);
            midptellipse(g,(float)70,(float)20,(float)-30+x,(float)-
50,(double)65);
            midptellipse(g,(float)79,(float)25,(float)55+x,(float)-
40,(double)125);
            midptellipse(g,(float)50,(float)20,(float)-70+x,(float)-
130,(double)55);
            midptellipse(g,(float)20,(float)50,(float)80+x,(float)-
130,(double)0);
            midptellipse(g,(float)20,(float)10,(float)60+x,(float)-
180,(double)0);
            midptellipse(g,(float)30,(float)10,(float)-100+x,(float)-
170,(double)0);
            midptellipse(g,(float)40,(float)10,(float)-
140+x,(float)80,(double)40);
            midptellipse(g,(float)40,(float)10,(float)-
200+x,(float)60,(double)170);
            midptellipse(g,(float)40,(float)10,(float)-
160+x,(float)100,(double)10);
            midptellipse(g,(float)40,(float)10,(float)-
220+x,(float)110,(double)155);
             Circles(g,20,-260+x,125);
```

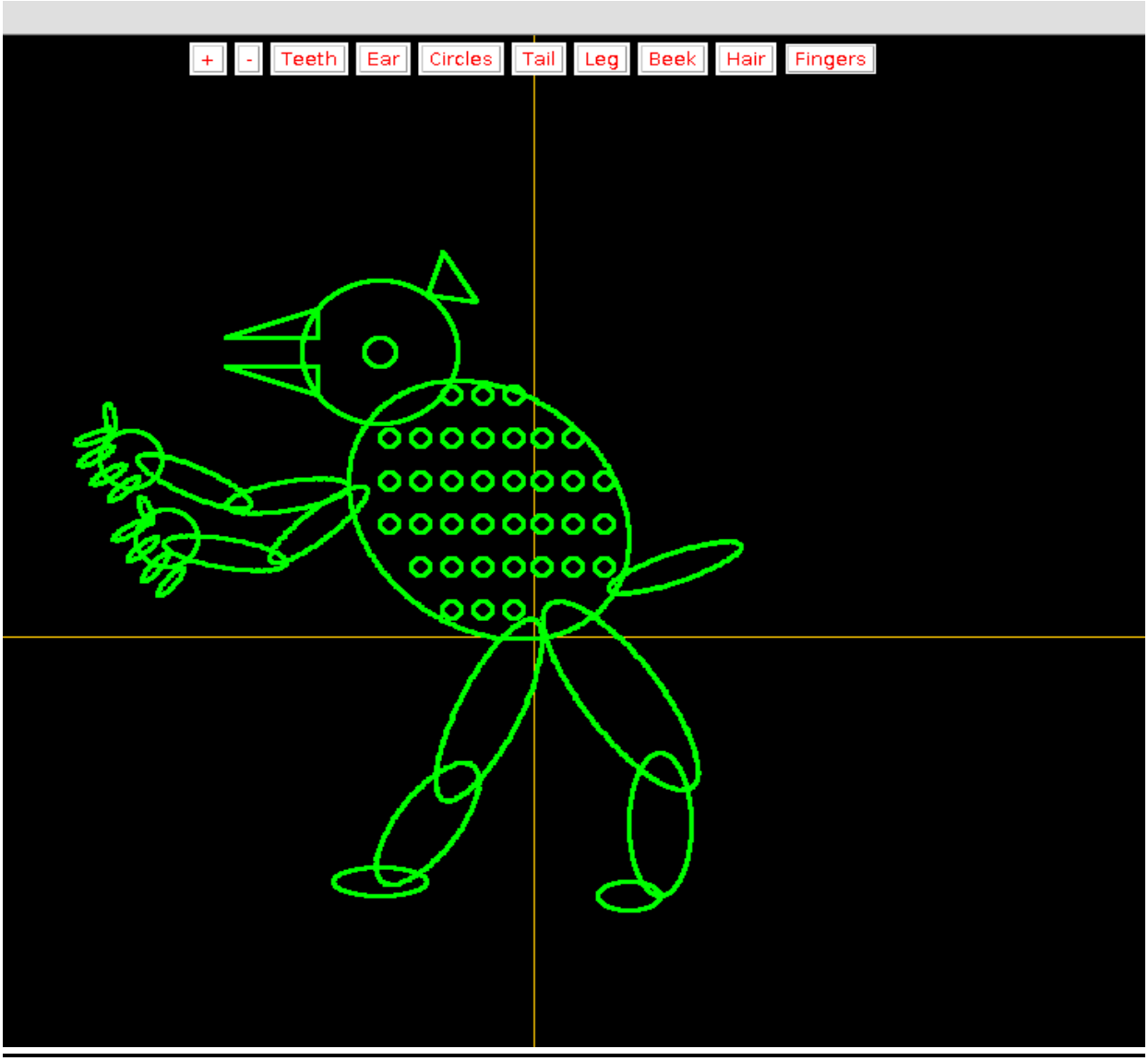
```
              Circles(g,20,-237+x,70);
            //function for making teeth
              maketriangle(g,x);
              maketeeth(g,x);
            makeear(g,x);
            //
midptellipse(g,(float)45,(float)10,(float)90,(float)50,(double)20);
              makespot(g,x);
              maketailtri(g,x);
              maketailellipse(g,x);
              makebeek(g,x);
              //Traingle making

            // midptellipse(g,(float)3,(float)6,(float)1,(float)2);
            // paintGrid(g,gap,originx,originy);

               makehairytail(g,x);
               makeleg(g,x);
               makehair(g);
               makefingers(g);
    }
    public void paint(Graphics g){
            g.setColor(Color.orange);
            int originx=getX()+getWidth()/2;
            int originy=getY()+getHeight()/2;
            g.drawLine(originx-getWidth()/2, originy,
originx+getWidth()/2, originy);
            g.drawLine(originx, originy-getHeight()/2, originx,
originy+getHeight()/2);
            // paintGrid(g,gap,originx,originy);
            Color c=new Color(100,100,100);
            int i=0;
            int x1=200,y1=101;
            clone(g,0);
            // clone(g,-400);
            // clone(g,400);


    }
};
```

DIAGRAM.HTML-

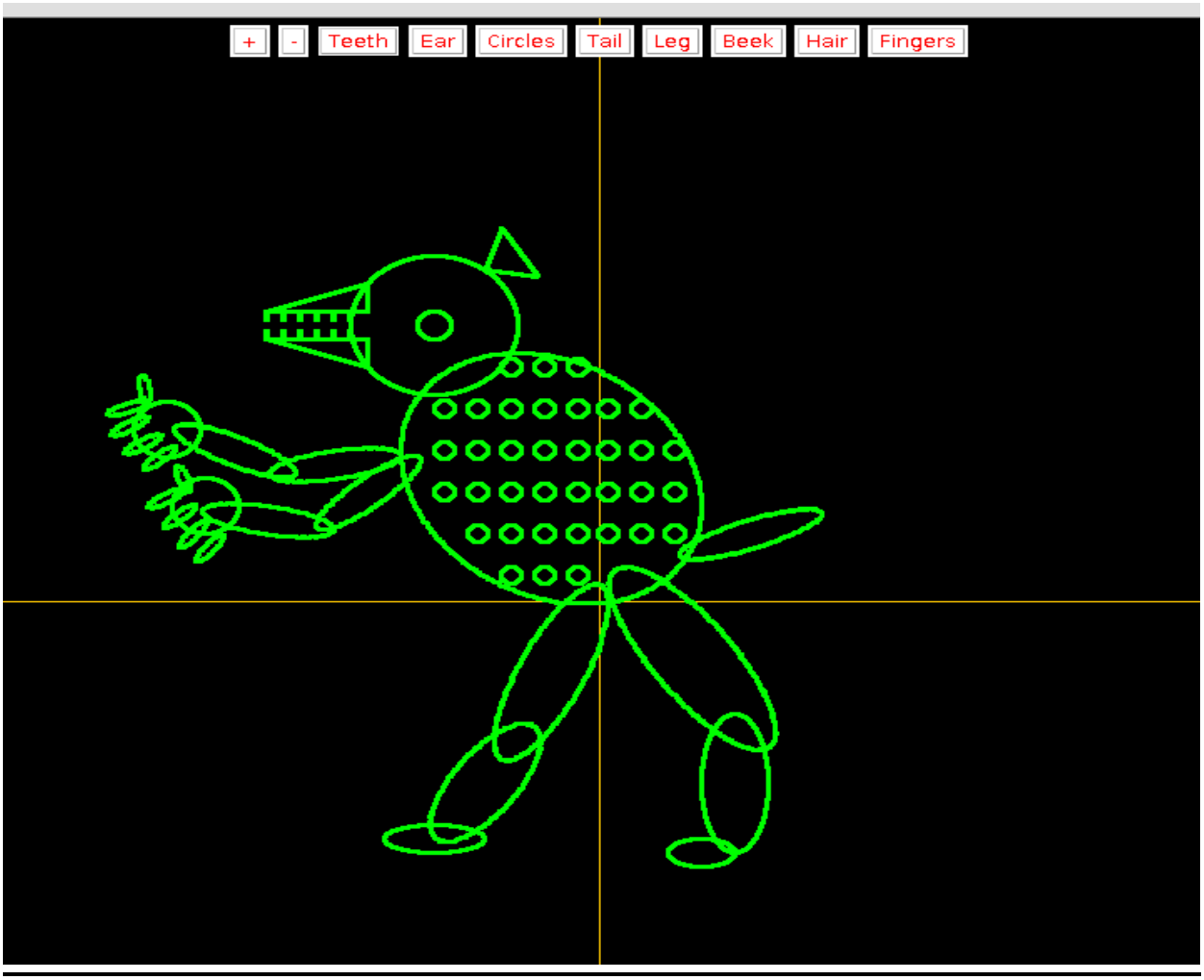
```
<html>
    <head></head>
    <body>
        <applet code ="Diagram.class"  height="1000"
width="1000"></applet>
    </body>
</html>
```

DIFFERENT OUTPUTS-

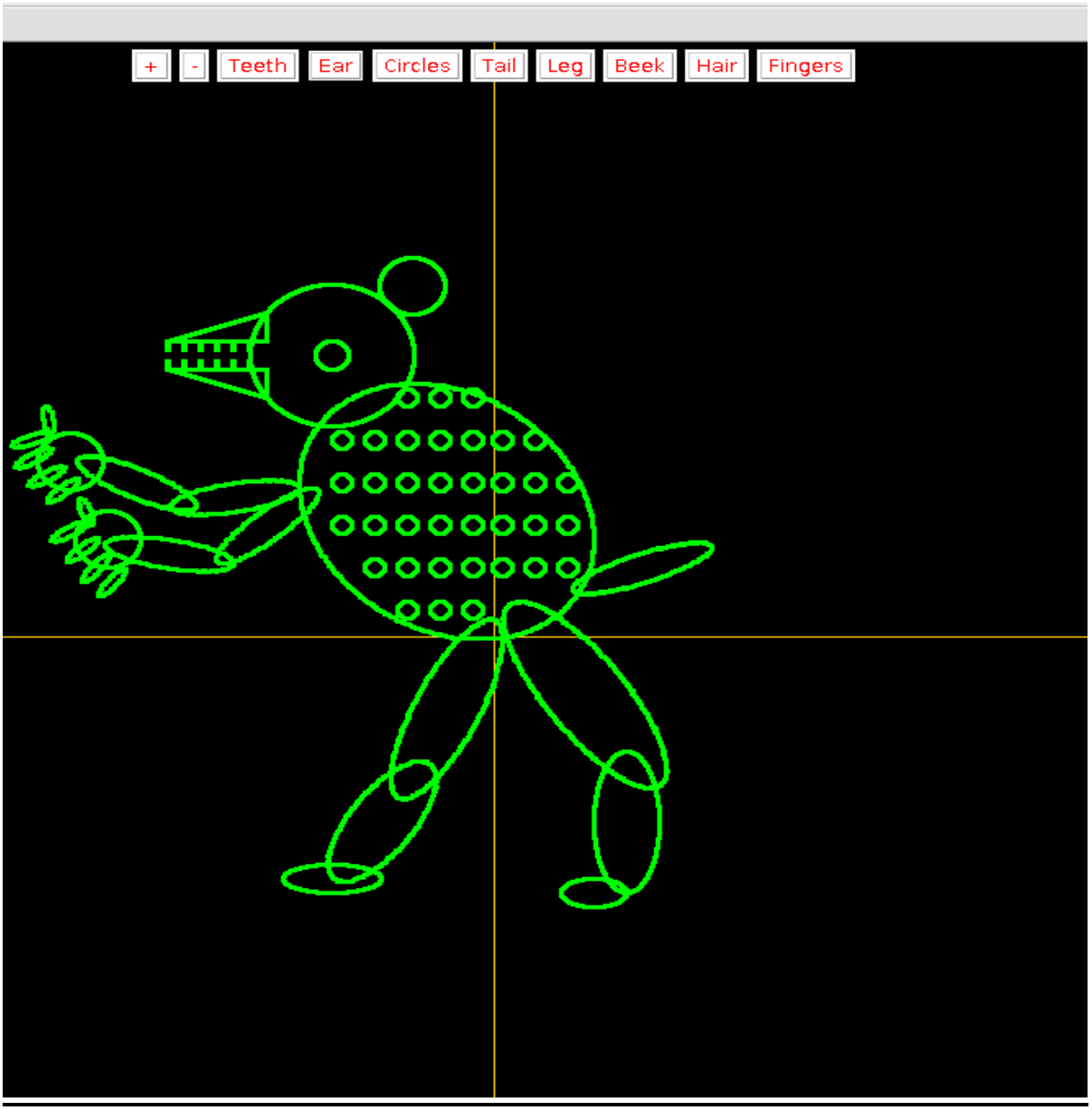
1.Simple



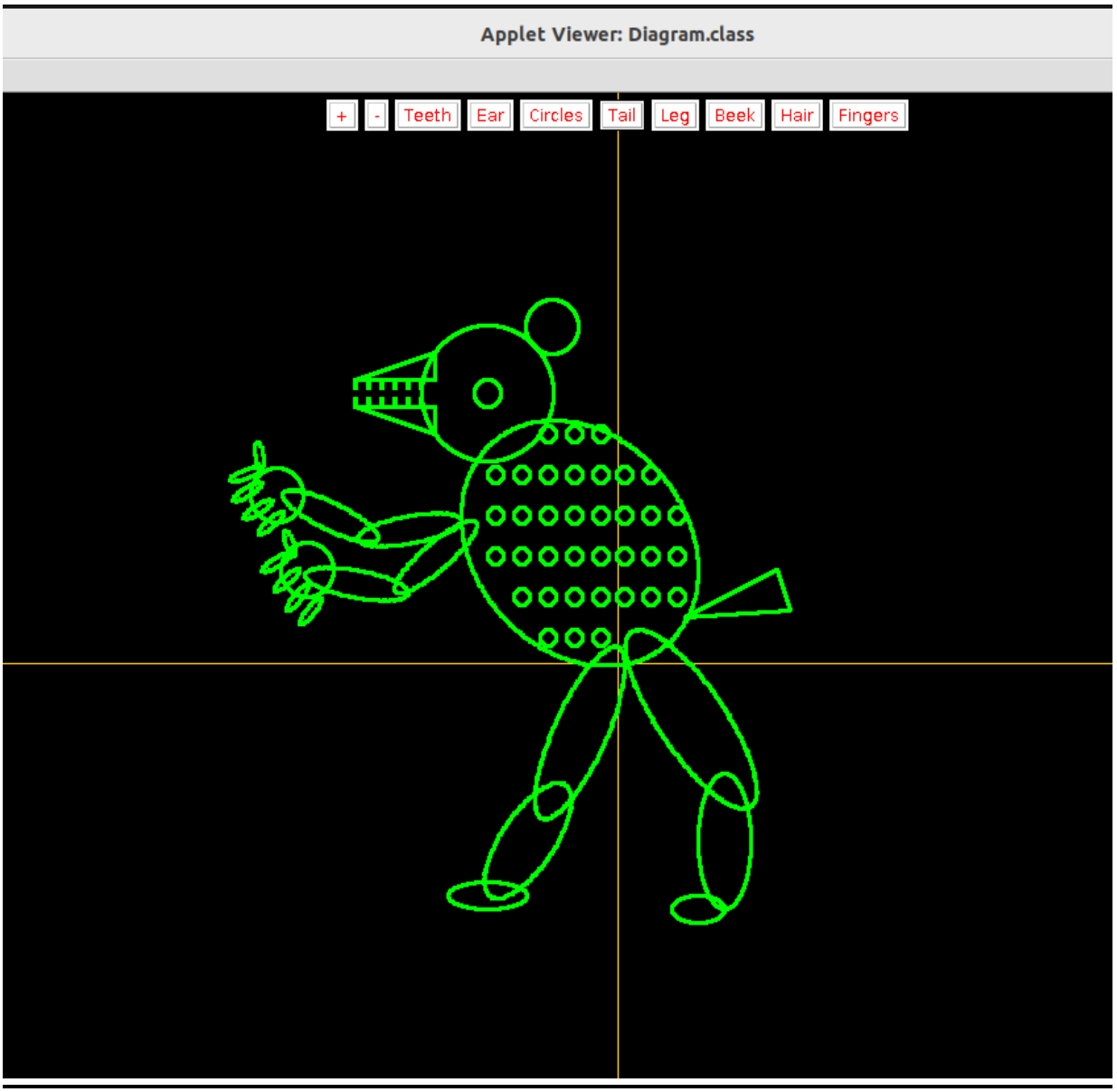
2.After Adding Teeth-



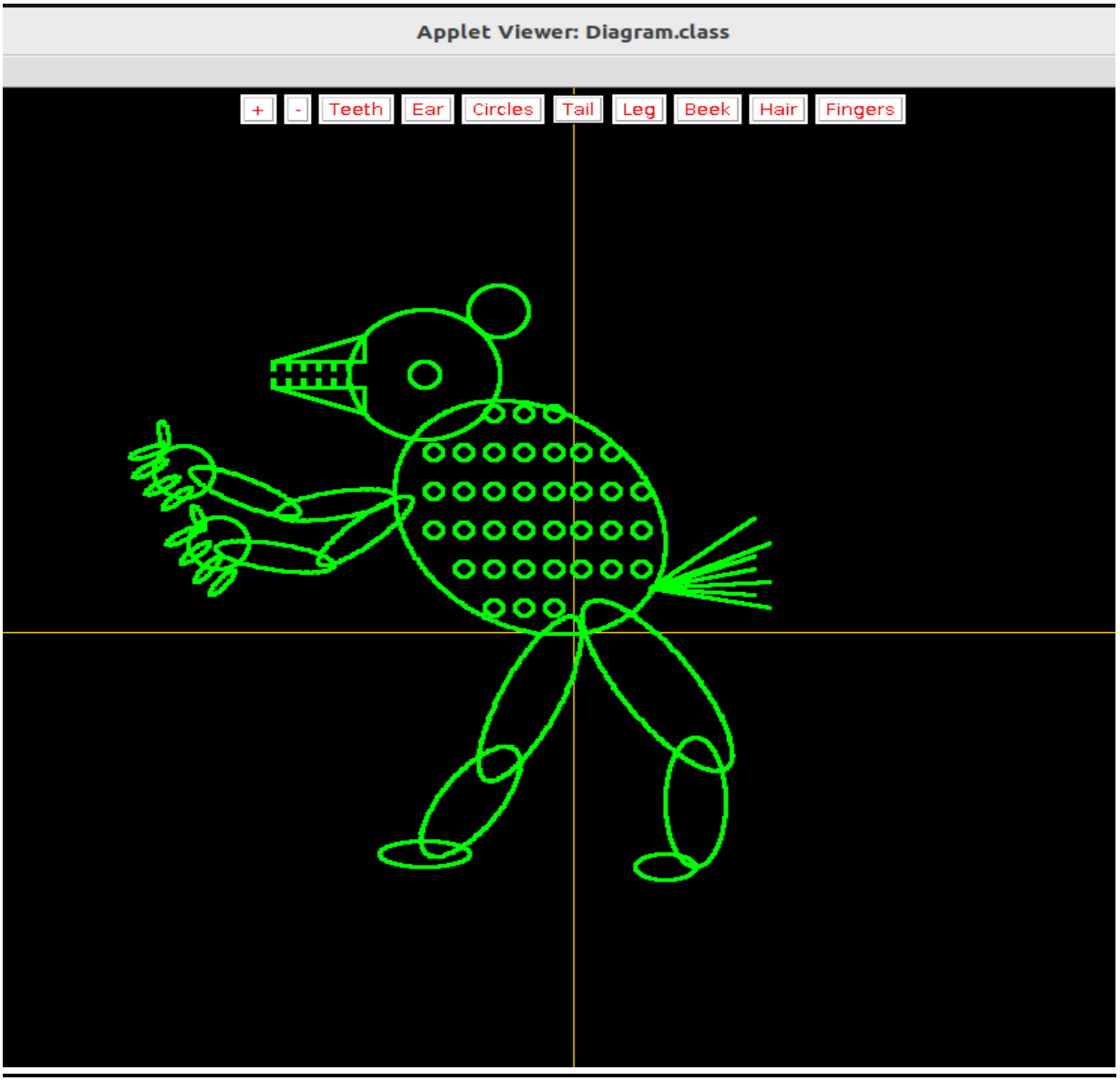
3.After Changing ear-



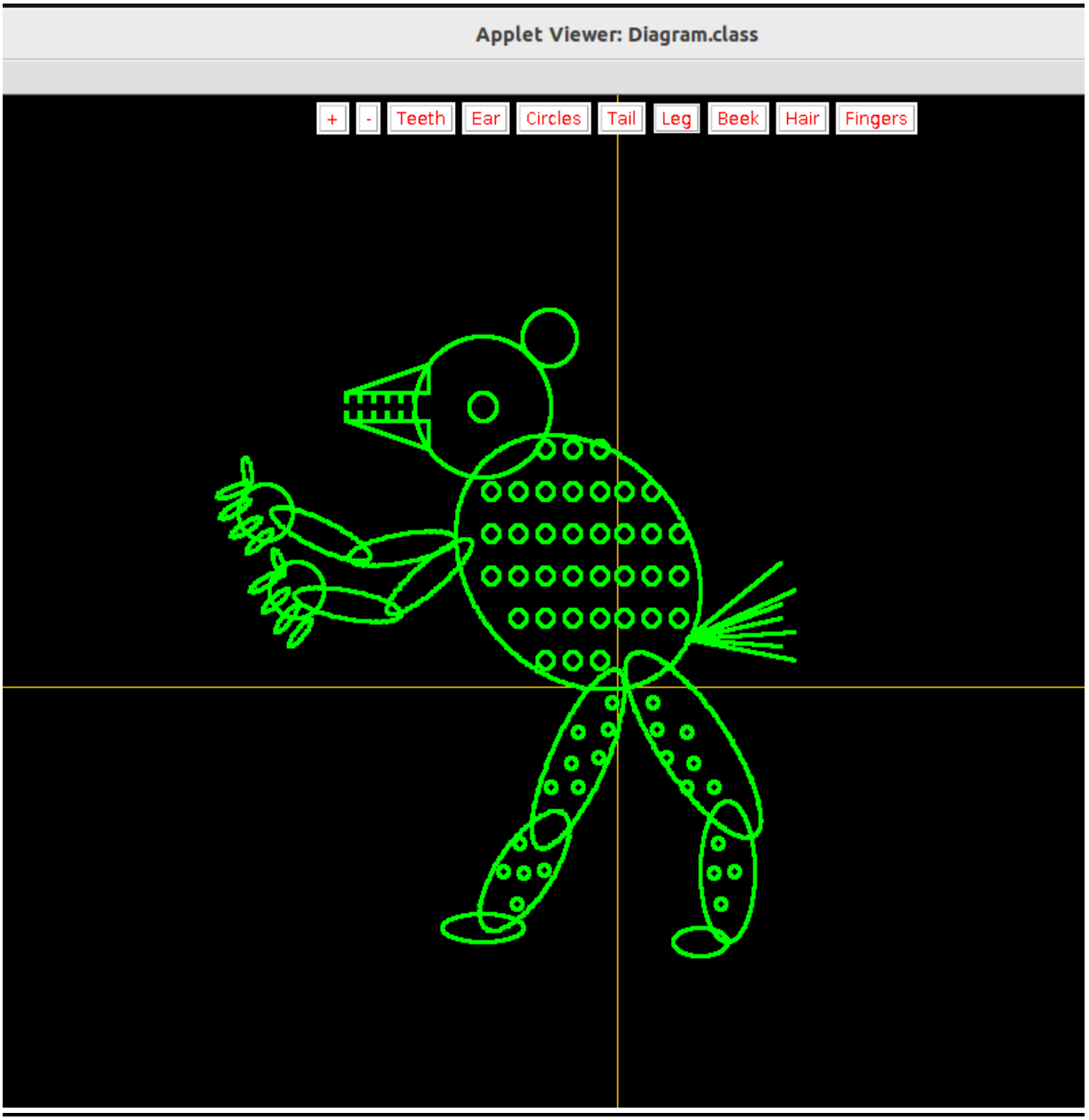
4.After Changing Shape of Tail-



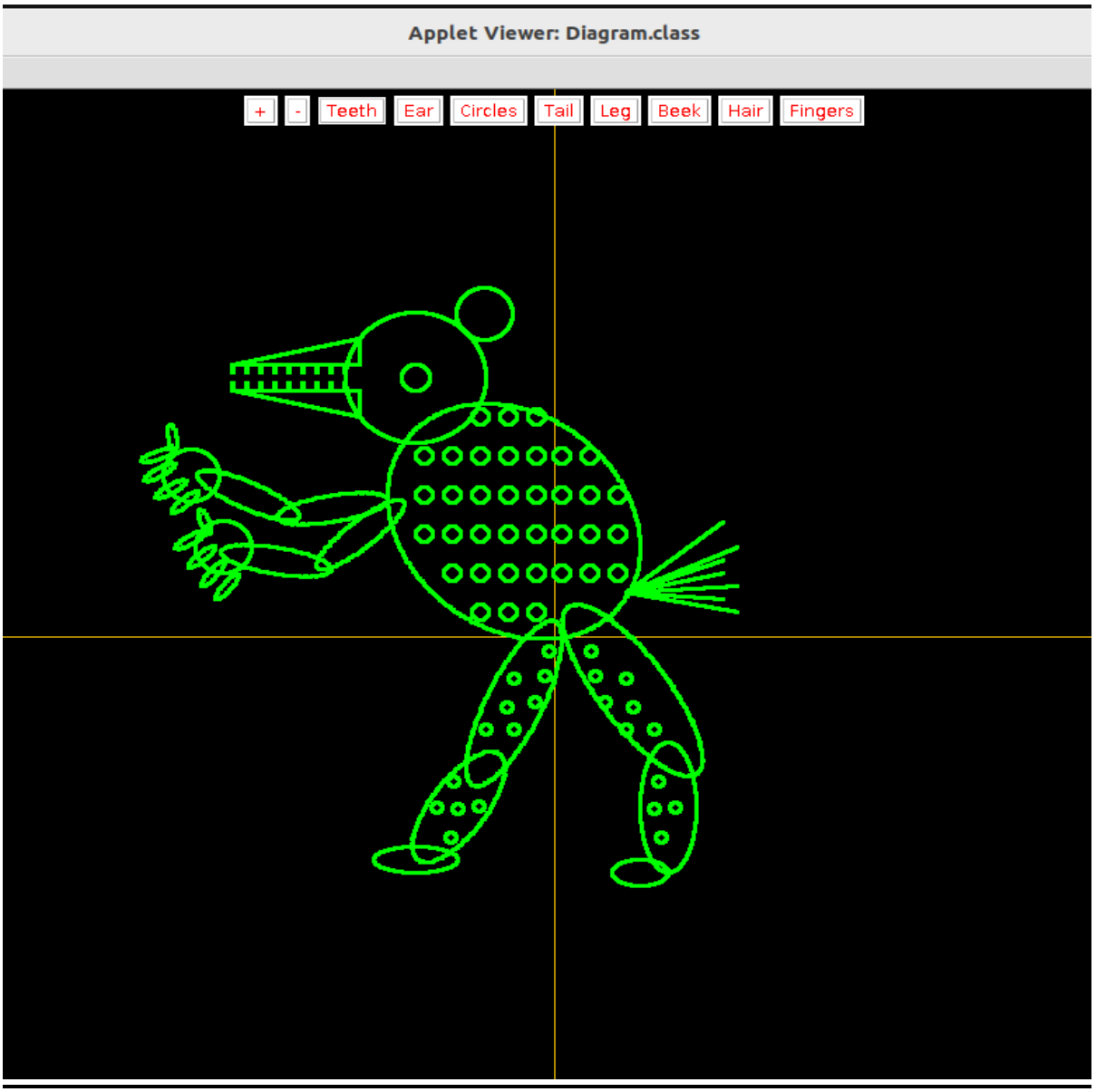
5.After Changing Tail Second time-



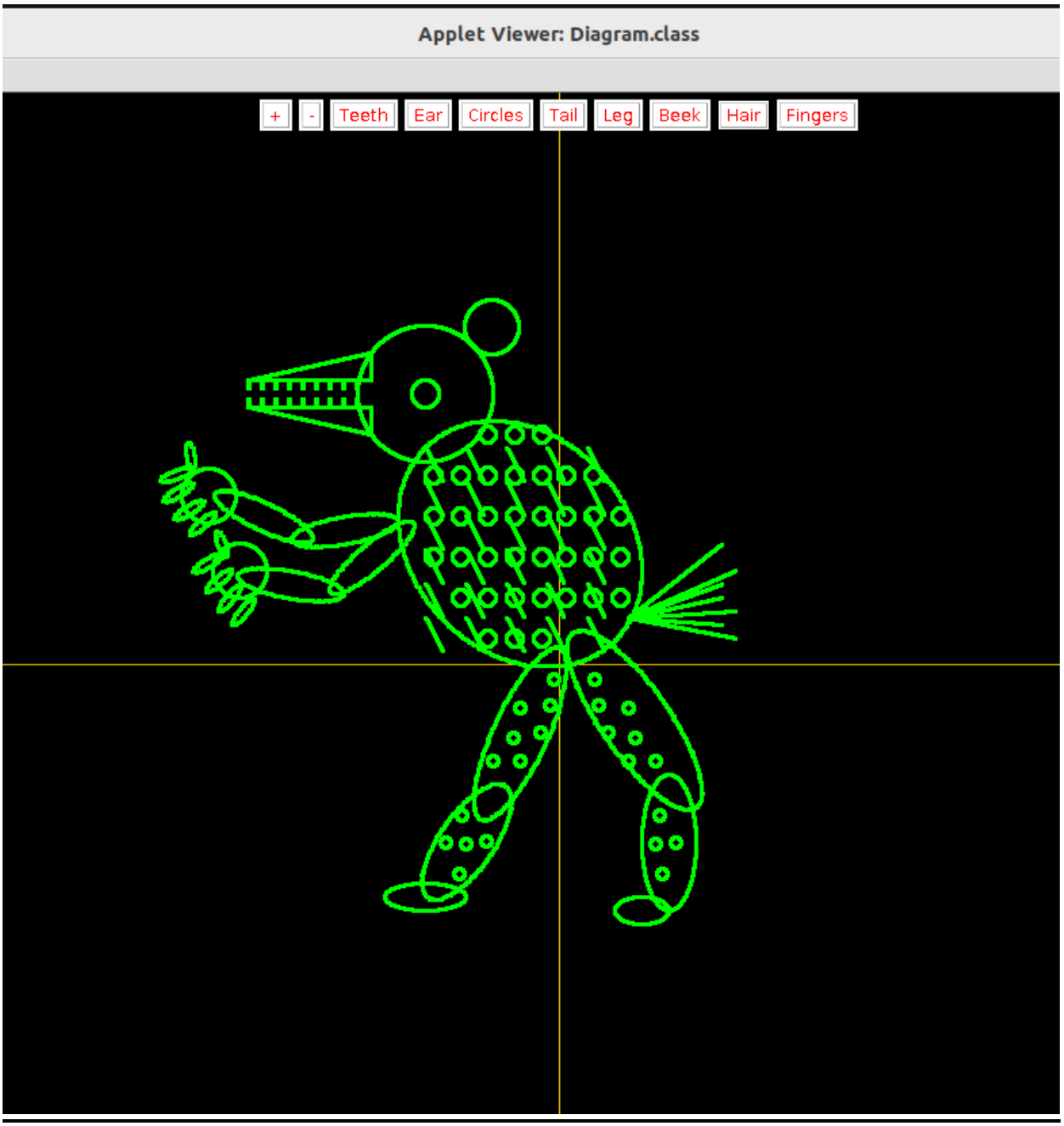
6.After Adding Spots in Leg-



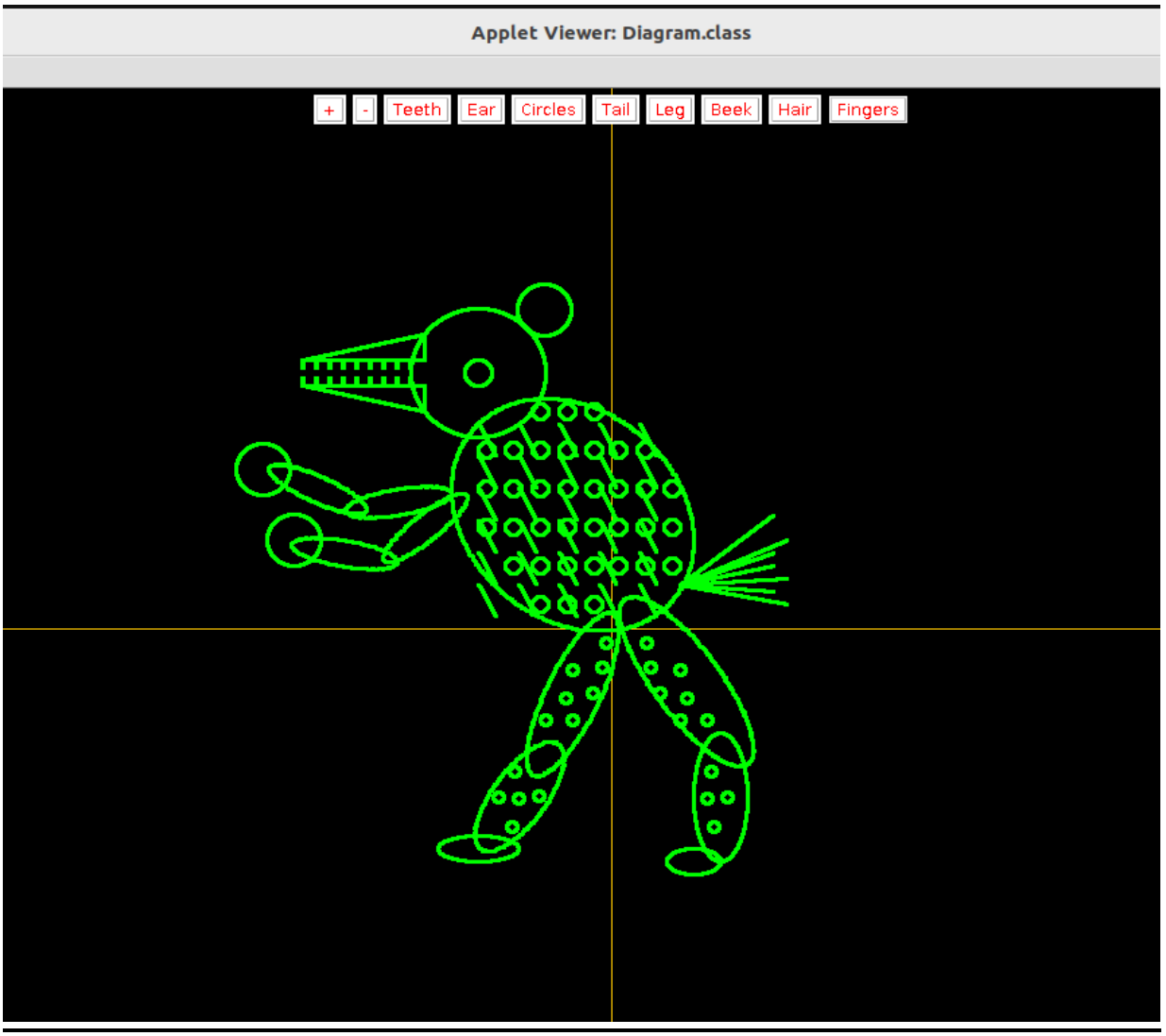
7.After Increasing the size of beak -



8.After Adding hairs on the body-



9.After removing fingers-



2Q. Ans-code-DIAGRAM1.JAVA-

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class Diagram1 extends Applet implements
ActionListener,MouseWheelListener{
    //It is for generating a rectangle corresponding to a particular
point in cartesian coordinate syatem
        int gap = 2;
        int temp=0;
        int temp1=0;
        int temp2=0;
        int temp3=0;
        int temp5=0;
        int temp7=0;
        int temp6=0;
        int temp8=1;
        int animal1=0;
        int animal2=0;
        int animal3=0;
        Button button1 = new Button("+");
        Button button2 = new Button("-");
        Button button = new Button("Animal1");
        Button button3=new Button("Animal2");
        Button button4=new Button("Combined");
    //It is for initialisation purpose
    public void init(){
        add(button1);
        add(button2);
        add(button);
        add(button3);
        add(button4);
        button1.setBackground(Color.white);
        button2.setBackground(Color.white);
        button.setBackground(Color.white);
        button3.setBackground(Color.white);
        button4.setBackground(Color.white);


        button1.addActionListener(this);
        button2.addActionListener(this);
        button.addActionListener(this);
        button3.addActionListener(this);
        button4.addActionListener(this);
```

```
        addMouseWheelListener(this);
        setForeground(Color.red);
        setBackground(Color.black);
    }
    //it is for implementing button function
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == button1){
         gap+=gap+gap/10;
         repaint();
        }
        if(e.getSource()==button2)
        {
             gap-=gap/10;
             repaint();
        }
        if(e.getSource()== button)
        {
          if(animal1==0)
            animal1=1;
          else
            animal1=0;

         repaint();
        }
        if(e.getSource()==button3)
        {
            if(animal2==0)
              animal2=1;
            else
             animal2=0;
             repaint();
        }
        if(e.getSource()==button4)
        {
            if(animal3==0)
              animal3=1;
            else
              animal3=0;
              repaint();
        }


    }
    public void plotPoint(Graphics g,int x,int y,Color c)
    {
        g.setColor(c);
        g.fillRect(
```

```
            (getX()+getWidth())/2+(x*gap)-(gap/2),
            (getY()+getHeight())/2-(y*gap)-(gap/2),
            3*gap,3*gap
        );
    }
    public int slope(int x1,int x2,int y1,int y2)
    {
        int x=x2-x1;
        int y=y2-y1;
        int m=y/x;
        return m;
    }
    //Round function
    public int round(float n) {
    if (n - (int)n < 0.5)
        return (int)n;
    return (int)(n + 1);
}
//Circle Drawing Algorithm
 public void Circles(Graphics g,int radius,int x1,int y1)
    {
        int x=0;
        int y=radius;
        double p=1.25-radius;
        plotPoint(g,x+x1,y+y1,Color.green);
        plotPoint(g,x+x1,-y+y1,Color.green);
        plotPoint(g,-x+x1,y+y1,Color.green);
        plotPoint(g,-x+x1,-y+y1,Color.green);
        while(x<y)
        {
            if(p<0)
                {
                    x=x+1;
                    p=p+2*x+1;

                }
            else
            {
                x=x+1;
                y=y-1;
                p=p+2*x+1-2*y;
            }
             plotPoint(g,x+x1,y+y1,Color.green);
             plotPoint(g,y+x1,x+y1,Color.green);
             plotPoint(g,x+x1,-y+y1,Color.green);
             plotPoint(g,-x+x1,y+y1,Color.green);
             plotPoint(g,y+x1,-x+y1,Color.green);
             plotPoint(g,-y+x1,x+y1,Color.green);
```

```
            plotPoint(g,-x+x1,-y+y1,Color.green);
            plotPoint(g,-y+x1,-x+y1,Color.green);
        }
    }



//Line Drawing Algorithm
 //function to return line
    public void DDALine(Graphics g,int x0, int y0, int x1, int y1) {

      //calculate dx and dy
      int dx = x1 - x0;
    int dy= y1 - y0;

    int step;

      //if dx > dy we will take step as dx
      //else we will take step as dy to draw the complete line
    if (Math.abs(dx) > Math.abs(dy))
        step = Math.abs(dx);
    else
        step = Math.abs(dy);

      //calculate x-increment and y-increment for each step
    float x_incr = (float)dx / step;
    float y_incr = (float)dy / step;

      //take the initial points as x and y
    float x = x0;
    float y = y0;

    for (int i = 0; i < step; i ++) {
        //putpixel(round(x), round(y), WHITE);
       plotPoint(g, round(x), round(y), Color.green);
        x += x_incr;
        y += y_incr;
        //delay(10);
    }
}


//Ellipse drawing algorithm
public void midptellipse(Graphics g,float rx, float ry,
                         float xc, float yc,Double degree)
{

    float dx, dy, d1, d2, x, y;
    x = 0;
```

```
    y = ry;
    double radian=Math.toRadians(degree);
    // Initial decision parameter of region 1
    d1 = (ry * ry) - (rx * rx * ry) +
                    (0.25f * rx * rx);
    dx = 2 * ry * ry * x;
    dy = 2 * rx * rx * y;
    // DecimalFormat df = new DecimalFormat("#,###,##0.00000");

    // For region 1
    while (dx < dy)
    {

     plotPoint(g,((int)((x)*Math.cos(radian)+xc-
(y)*Math.sin(radian))),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(radi
an))),Color.green);
    plotPoint(g,((int)((-x)*Math.cos(radian)+xc-
(y)*Math.sin(radian))),((int)((-
x)*Math.sin(radian)+yc+(y)*Math.cos(radian))),Color.green);
    plotPoint(g,((int)((x)*Math.cos(radian)+xc-(-
y)*Math.sin(radian))),((int)((x)*Math.sin(radian)+yc+(-
y)*Math.cos(radian))),Color.green);
   plotPoint(g,((int)((-x)*Math.cos(radian)+xc-(-
y)*Math.sin(radian))),((int)((-x)*Math.sin(radian)+yc+(-
y)*Math.cos(radian))),Color.green);
    //  plotPoint(g,((int)((-x)*Math.cos(radian)+xc-
(y)*Math.sin(radian))),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(radi
an))),Color.red);
    //  plotPoint(g,((int)((x)*Math.cos(radian)+xc-(-
y)*Math.sin(radian))),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(radia
n))),Color.red);
    //  plotPoint(g,((int)((-x)*Math.cos(radian)+xc-(-
y)*Math.sin(radian))),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(radia
n))),Color.red);

        if (d1 < 0)
        {
            x++;
            dx = dx + (2 * ry * ry);
            d1 = d1 + dx + (ry * ry);
        }
        else
        {
            x++;
            y--;
            dx = dx + (2 * ry * ry);
            dy = dy - (2 * rx * rx);
            d1 = d1 + dx - dy + (ry * ry);
```

```
        }
    }

    // Decision parameter of region 2
    d2 = ((ry * ry) * ((x + 0.5f) * (x + 0.5f)))
        + ((rx * rx) * ((y - 1) * (y - 1)))
        - (rx * rx * ry * ry);

    // Plotting points of region 2
    while (y >= 0) {

        // printing points based on 4-way symmetry
        // plotPoint(g,(int)(x+xc),(int)(y+yc),Color.red);
        // plotPoint(g,(int)(-x+xc),(int)(y+yc),Color.red);
        // plotPoint(g,(int)(x+xc),(int)(-y+yc),Color.red);
        // plotPoint(g,(int)(-x+xc),(int)(-y+yc),Color.red);

    plotPoint(g,((int)((x)*Math.cos(radian)-
(y)*Math.sin(radian)+xc)),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(r
adian))),Color.green);
    plotPoint(g,((int)((-x)*Math.cos(radian)-
(y)*Math.sin(radian)+xc)),((int)((-
x)*Math.sin(radian)+yc+(y)*Math.cos(radian))),Color.green);
    plotPoint(g,((int)((x)*Math.cos(radian)-(-
y)*Math.sin(radian)+xc)),((int)((x)*Math.sin(radian)+yc+(-
y)*Math.cos(radian))),Color.green);
    plotPoint(g,((int)((-x)*Math.cos(radian)-(-
y)*Math.sin(radian)+xc)),((int)((-x)*Math.sin(radian)+yc+(-
y)*Math.cos(radian))),Color.green);

    //  plotPoint(g,((int)((-x)*Math.cos(radian)-
(y)*Math.sin(radian)+xc)),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(r
adian))),Color.red);
    //  plotPoint(g,((int)((x)*Math.cos(radian)-(-
y)*Math.sin(radian)+xc)),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(ra
dian))),Color.red);
    //  plotPoint(g,((int)((-x)*Math.cos(radian)-(-
y)*Math.sin(radian)+xc)),((int)((x)*Math.sin(radian)+yc+(y)*Math.cos(ra
dian))),Color.red);

        if (d2 > 0) {
            y--;
            dy = dy - (2 * rx * rx);
            d2 = d2 + (rx * rx) - dy;
        }
        else {
            y--;
            x++;
```

```
            dx = dx + (2 * ry * ry);
            dy = dy - (2 * rx * rx);
            d2 = d2 + dx - dy + (rx * rx);
        }
    }
}
public void paintGrid(Graphics g,int gap,int originx,int originy)
    {
        g.setColor(Color.black);

        for(int i = gap;i<=getWidth();i+=gap)
        {
            g.drawLine(originx+i, originy-getHeight()/2, originx+i,
originy+getHeight()/2);
            g.drawLine(originx-i, originy-getHeight()/2, originx-i,
originy+getHeight()/2);
        }
        for(int i = gap;i<=getHeight();i+=gap)
        {
            g.drawLine(originx-getWidth()/2, originy+i,
originx+getWidth()/2, originy+i);
            g.drawLine(originx-getWidth()/2, originy-i,
originx+getWidth()/2, originy-i);

        }
    }









    //It is for mouse wheel operation
    public void mouseWheelMoved(MouseWheelEvent e)
    {
        int z=e.getWheelRotation();
        gap+=z;
        repaint();
    }
    //function for making spotted cirecles
    public void makespot(Graphics g,int x,int temp2)
    {
        if(temp2==0)
        {
          Circles(g,6,44+x,50);
        Circles(g,6,24+x,50);
        Circles(g,6,4+x,50);
```

```
        Circles(g,6,-14+x,50);
        Circles(g,6,-34+x,50);
        Circles(g,6,-54+x,50);
        Circles(g,6,-74+x,50);

        Circles(g,6,44+x,80);
        Circles(g,6,24+x,80);
        Circles(g,6,4+x,80);
        Circles(g,6,-14+x,80);
        Circles(g,6,-34+x,80);
        Circles(g,6,-54+x,80);
        Circles(g,6,-74+x,80);
        Circles(g,6,-94+x,80);

        Circles(g,6,44+x,110);
        Circles(g,6,24+x,110);
        Circles(g,6,4+x,110);
        Circles(g,6,-14+x,110);
        Circles(g,6,-34+x,110);
        Circles(g,6,-54+x,110);
        Circles(g,6,-74+x,110);
        Circles(g,6,-94+x,110);


        Circles(g,6,24+x,140);
        Circles(g,6,4+x,140);
        Circles(g,6,-14+x,140);
        Circles(g,6,-34+x,140);
        Circles(g,6,-54+x,140);
        Circles(g,6,-74+x,140);
        Circles(g,6,-94+x,140);

        Circles(g,6,-14+x,20);
        Circles(g,6,-34+x,20);
        Circles(g,6,-54+x,20);

        Circles(g,6,-34+x,170);
        Circles(g,6,-14+x,170);
        Circles(g,6,-54+x,170);
        }
    }
    //function to make hairy tail
    public void makehairytail(Graphics g,int x,int temp3)
    {
        if(temp3==2)
        {
           DDALine(g,50+x,35,120+x,90);
            DDALine(g,50+x,35,120+x,60);
```

```
        DDALine(g,50+x,35,130+x,70);
        DDALine(g,50+x,35,120+x,50);
        DDALine(g,50+x,35,130+x,40);
        DDALine(g,50+x,35,120+x,30);
        DDALine(g,50+x,35,130+x,20);
        }
    }
    //function to make ellipse tail
    public void maketailellipse(Graphics g,int x,int temp3)
    {
        if(temp3==0)

midptellipse(g,(float)45,(float)10,(float)90+x,(float)50,(double)20);
    }
    //function for making legspot
    public void maketailtri(Graphics g,int x,int temp3)
    {
        if(temp3==1)
        {
           DDALine(g,50+x,35,120+x,70);
        DDALine(g,50+x,35,130+x,40);
        DDALine(g,120+x,70,130+x,40);
        }
    }
    //function for making triangle
    public void maketriangle(Graphics g,int x,int temp1)
    {
            if(temp1==0)
            {
                DDALine(g,-70+x,240,-60+x,270);
                DDALine(g,-70+x,240,-38+x,235);
                DDALine(g,-60+x,270,-38+x,235);
            }
    }
    //function for making leg
    public void makeleg(Graphics g,int x,int temp5)
    {
        if(temp5==1)
        {
            //front upper
         Circles(g,4,-50+x,-70);
        Circles(g,4,-30+x,-70);
        Circles(g,4,-15+x,-49);
        Circles(g,4,-30+x,-31);
        Circles(g,4,-35+x,-53);
        Circles(g,4,-5+x,-10);
        Circles(g,4,-8+x,-29);
        //front lower
```

```
        Circles(g,4,-85+x,-130);
        Circles(g,4,-70+x,-131);
        Circles(g,4,-75+x,-153);
        Circles(g,4,-73+x,-110);
        Circles(g,4,-55+x,-129);
        // //back upper
        Circles(g,4,70+x,-70);
        Circles(g,4,50+x,-70);
        Circles(g,4,35+x,-49);
        Circles(g,4,50+x,-31);
        Circles(g,4,55+x,-53);
        Circles(g,4,25+x,-10);
        Circles(g,4,28+x,-29);
        // //back lower
        Circles(g,4,85+x,-130);
        Circles(g,4,70+x,-131);
        Circles(g,4,75+x,-153);
        Circles(g,4,73+x,-110);
        }
    }
    //function for making circle of ear
    public void makeear(Graphics g,int x,int temp1)
    {
        if(temp1==1)
          Circles(g,20,-51+x,249);
    }
    //function for teeth making
    public void maketeeth(Graphics g,int x,int temp,int temp6)
    {
        if(temp==1 && temp6==0){
            DDALine(g,-200+x,190,-200+x,197);
              DDALine(g,-180+x,190,-180+x,197);
              DDALine(g,-160+x,190,-160+x,197);
              DDALine(g,-150+x,190,-150+x,197);
              DDALine(g,-160+x,190,-160+x,197);
              DDALine(g,-170+x,190,-170+x,197);
              DDALine(g,-180+x,190,-180+x,197);
              DDALine(g,-190+x,190,-190+x,197);

              DDALine(g,-201+x,190,-201+x,197);
              DDALine(g,-181+x,190,-181+x,197);
              DDALine(g,-161+x,190,-161+x,197);
              DDALine(g,-151+x,190,-151+x,197);
              DDALine(g,-161+x,190,-161+x,197);
              DDALine(g,-171+x,190,-171+x,197);
              DDALine(g,-181+x,190,-181+x,197);
              DDALine(g,-191+x,190,-191+x,197);
```

```
            DDALine(g,-200+x,210,-200+x,203);
            DDALine(g,-180+x,210,-180+x,203);
            DDALine(g,-160+x,210,-160+x,203);
            DDALine(g,-150+x,210,-150+x,203);
            DDALine(g,-160+x,210,-160+x,203);
            DDALine(g,-170+x,210,-170+x,203);
            DDALine(g,-180+x,210,-180+x,203);
            DDALine(g,-190+x,210,-190+x,203);

            DDALine(g,-201+x,210,-201+x,203);
            DDALine(g,-181+x,210,-181+x,203);
            DDALine(g,-161+x,210,-161+x,203);
            DDALine(g,-151+x,210,-151+x,203);
            DDALine(g,-161+x,210,-160+x,203);
            DDALine(g,-171+x,210,-171+x,203);
            DDALine(g,-181+x,210,-181+x,203);
            DDALine(g,-191+x,210,-191+x,203);
        }
        else if(temp==0 && temp6==1)
        {
            DDALine(g,-200+x,190,-200+x,197);
              DDALine(g,-180+x,190,-180+x,197);
              DDALine(g,-160+x,190,-160+x,197);
              DDALine(g,-150+x,190,-150+x,197);
              DDALine(g,-160+x,190,-160+x,197);
              DDALine(g,-170+x,190,-170+x,197);
              DDALine(g,-180+x,190,-180+x,197);
              DDALine(g,-190+x,190,-190+x,197);
              DDALine(g,-230+x,190,-230+x,197);
              DDALine(g,-210+x,190,-210+x,197);
              DDALine(g,-220+x,190,-220+x,197);

              DDALine(g,-201+x,190,-201+x,197);
              DDALine(g,-181+x,190,-181+x,197);
              DDALine(g,-161+x,190,-161+x,197);
              DDALine(g,-151+x,190,-151+x,197);
              DDALine(g,-161+x,190,-161+x,197);
              DDALine(g,-171+x,190,-171+x,197);
              DDALine(g,-181+x,190,-181+x,197);
              DDALine(g,-191+x,190,-191+x,197);
              DDALine(g,-231+x,190,-231+x,197);
              DDALine(g,-211+x,190,-211+x,197);
              DDALine(g,-221+x,190,-221+x,197);

              DDALine(g,-200+x,210,-200+x,203);
              DDALine(g,-180+x,210,-180+x,203);
              DDALine(g,-160+x,210,-160+x,203);
              DDALine(g,-150+x,210,-150+x,203);
```

```
            DDALine(g,-160+x,210,-160+x,203);
            DDALine(g,-170+x,210,-170+x,203);
            DDALine(g,-180+x,210,-180+x,203);
            DDALine(g,-190+x,210,-190+x,203);
            DDALine(g,-210+x,210,-210+x,203);
            DDALine(g,-220+x,210,-220+x,203);
            DDALine(g,-230+x,210,-230+x,203);

            DDALine(g,-201+x,210,-201+x,203);
            DDALine(g,-181+x,210,-181+x,203);
            DDALine(g,-161+x,210,-161+x,203);
            DDALine(g,-151+x,210,-151+x,203);
            DDALine(g,-161+x,210,-160+x,203);
            DDALine(g,-171+x,210,-171+x,203);
            DDALine(g,-181+x,210,-181+x,203);
            DDALine(g,-191+x,210,-191+x,203);
            DDALine(g,-211+x,210,-211+x,203);
            DDALine(g,-221+x,210,-221+x,203);
            DDALine(g,-231+x,210,-231+x,203);
        }
    }
    //function for making hair
     public void makehair(Graphics g,int xshift,int temp7)
    {
        if(temp7==1)
        {
            for(int y=160;y>=20;y-=25)
            {
                if(y==180 || y==20)
                {
                    for(int i=-50+xshift;i<-10+xshift;i+=30)
                    {
                        DDALine(g,i,y,i+13,y-25);
                    }
                }
                else
                {
                    for(int i=-100+xshift;i<50+xshift;i+=30)
                    {
                        DDALine(g,i,y,i+13,y-25);
                    }

                }
            }
        }
    }
    //function to make fingers
    //function to make fingers
```

```
    public void makefingers(Graphics g,int shift)
    {
        if(temp8==1)
        {
        midptellipse(g,(float)13,(float)3,
                        (float)-245+(float)shift,(float)50,(double)50);
            midptellipse(g,(float)13,(float)3,
                        (float)-253+(float)shift,(float)63,(double)40);
            midptellipse(g,(float)13,(float)3,
                        (float)-235+(float)shift,(float)40,(double)50);
            midptellipse(g,(float)13,(float)3,
                        (float)-261+(float)shift,(float)76,(double)30);
            midptellipse(g,(float)10,(float)3,
                        (float)-
251+(float)shift,(float)90,(double)110);



            midptellipse(g,(float)13,(float)3,
                        (float)-245-
30+(float)shift,(float)50+65,(double)30);
            midptellipse(g,(float)13,(float)3,
                        (float)-253-
30+(float)shift,(float)63+65,(double)30);
            midptellipse(g,(float)13,(float)3,
                        (float)-235-
30+(float)shift,(float)40+65,(double)40);
            midptellipse(g,(float)13,(float)3,
                        (float)-261-
23+(float)shift,(float)76+65,(double)20);
            midptellipse(g,(float)10,(float)3,
                        (float)-251-
23+(float)shift,(float)90+65,(double)100);
        }
    }
    //function for beek
    public void makebeek(Graphics g,int x,int temp6)
    {
        if(temp6==0)
        {
            DDALine(g,-140+x,210,-140+x,230);
            DDALine(g,-140+x,210,-200+x,210);
            DDALine(g,-140+x,230,-200+x,210);
            DDALine(g,-140+x,190,-140+x,170);
            DDALine(g,-140+x,190,-200+x,190);
            DDALine(g,-140+x,170,-200+x,190);
        }
        else
        {
```

```
            DDALine(g,-140+x,210,-140+x,230);
            DDALine(g,-140+x,210,-230+x,210);
            DDALine(g,-140+x,230,-230+x,210);
            DDALine(g,-140+x,190,-140+x,170);
            DDALine(g,-140+x,190,-230+x,190);
            DDALine(g,-140+x,170,-230+x,190);

        }
    }
    //making clone
    public void clone(Graphics g,int x,int temp,int temp1,int temp2,int
temp3,int temp4,int temp5,int temp6,int temp7)
    {
        Circles(g,50,-100+x,200);
            Circles(g,10,-100+x,200);

            Circles(g,50,-100+x,200);
            Circles(g,10,-100+x,200);

            midptellipse(g,(float)100,(float)80,(float)-
30+x,(float)90,(double)135);
            midptellipse(g,(float)70,(float)20,(float)-30+x,(float)-
50,(double)65);
            midptellipse(g,(float)79,(float)25,(float)55+x,(float)-
40,(double)125);
            midptellipse(g,(float)50,(float)20,(float)-70+x,(float)-
130,(double)55);
            midptellipse(g,(float)20,(float)50,(float)80+x,(float)-
130,(double)0);
            midptellipse(g,(float)20,(float)10,(float)60+x,(float)-
180,(double)0);
            midptellipse(g,(float)30,(float)10,(float)-100+x,(float)-
170,(double)0);
            midptellipse(g,(float)40,(float)10,(float)-
140+x,(float)80,(double)40);
            midptellipse(g,(float)40,(float)10,(float)-
200+x,(float)60,(double)170);
            midptellipse(g,(float)40,(float)10,(float)-
160+x,(float)100,(double)10);
            midptellipse(g,(float)40,(float)10,(float)-
220+x,(float)110,(double)155);
             Circles(g,20,-260+x,125);
              Circles(g,20,-237+x,70);
            //function for making teeth
              maketeeth(g,x,temp,temp6);
              maketriangle(g,x,temp1);
            makeear(g,x,temp1);
```

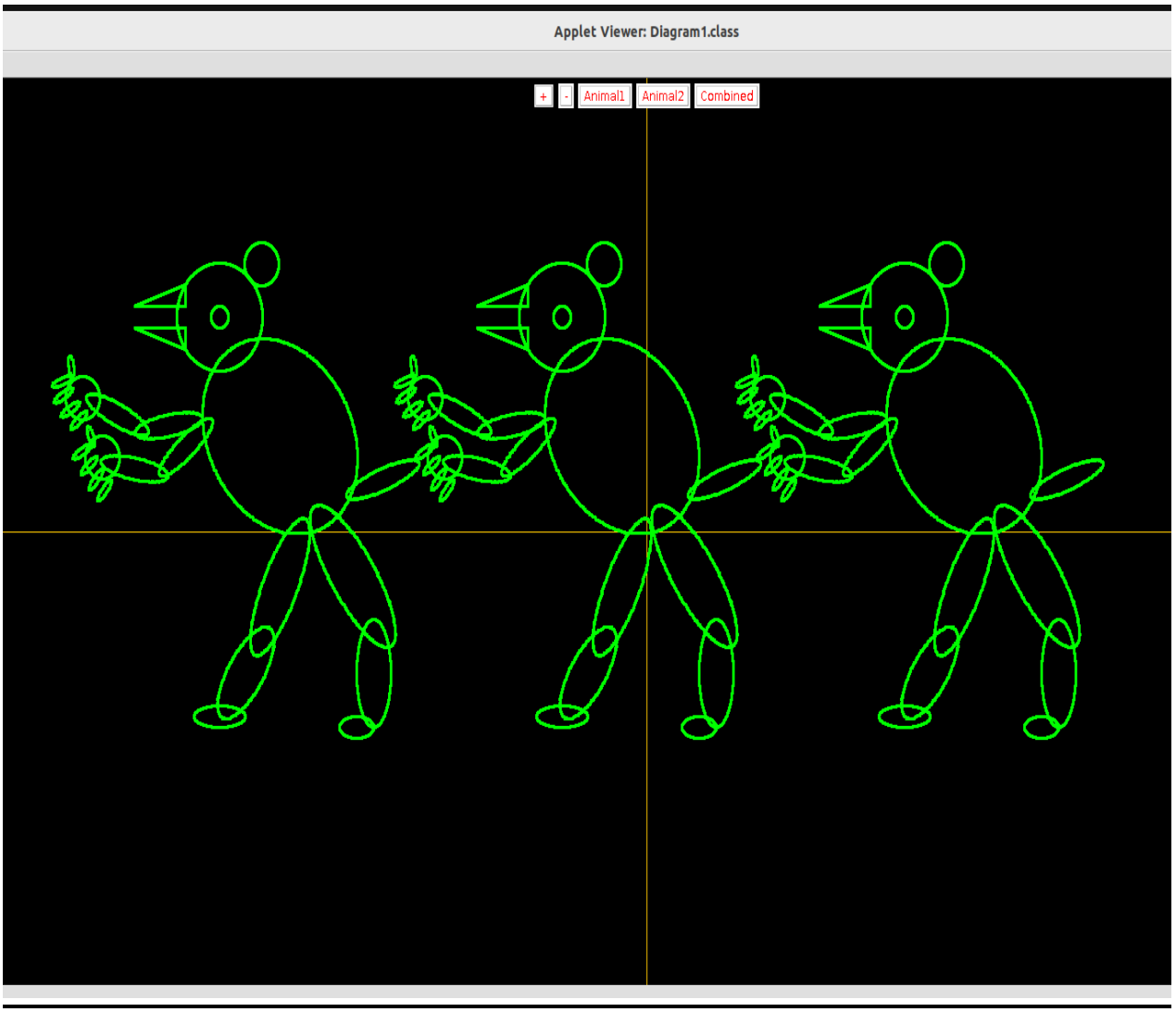
```
            //
midptellipse(g,(float)45,(float)10,(float)90,(float)50,(double)20);
              makespot(g,x,temp2);
              maketailtri(g,x,temp3);
              maketailellipse(g,x,temp3);
              makehair(g,x,temp7);
              makefingers(g,x);
              makebeek(g,x,temp6);
              //Traingle making

            // midptellipse(g,(float)3,(float)6,(float)1,(float)2);
            // paintGrid(g,gap,originx,originy);

          makehairytail(g,x,temp3);
          makeleg(g,x,temp5);
    }
    public void paint(Graphics g){
            g.setColor(Color.orange);
            int originx=getX()+getWidth()/2;
            int originy=getY()+getHeight()/2;
            g.drawLine(originx-getWidth()/2, originy,
originx+getWidth()/2, originy);
            g.drawLine(originx, originy-getHeight()/2, originx,
originy+getHeight()/2);
            // paintGrid(g,gap,originx,originy);
            Color c=new Color(100,100,100);
            int i=0;
            int x1=200,y1=101;
            if(animal1==0)
            clone(g,-400,0,1,1,0,0,0,0,0);
            else
            clone(g,-400,1,0,1,0,1,0,0,1);
            if(animal2==0)
            clone(g,0,0,1,1,0,0,0,0,0);
            else
            clone(g,0,0,1,0,1,0,1,1,0);
            if(animal3==0)
            clone(g,400,0,1,1,0,0,0,0,0);
            else
            clone(g,400,1,1,1,1,1,1,0,1);
            // clone(g,0,0,1,1,0,1);
            // clone(g,400,0,0,0,1,1);

    }
};
```

DIAGRAM1.HTML-

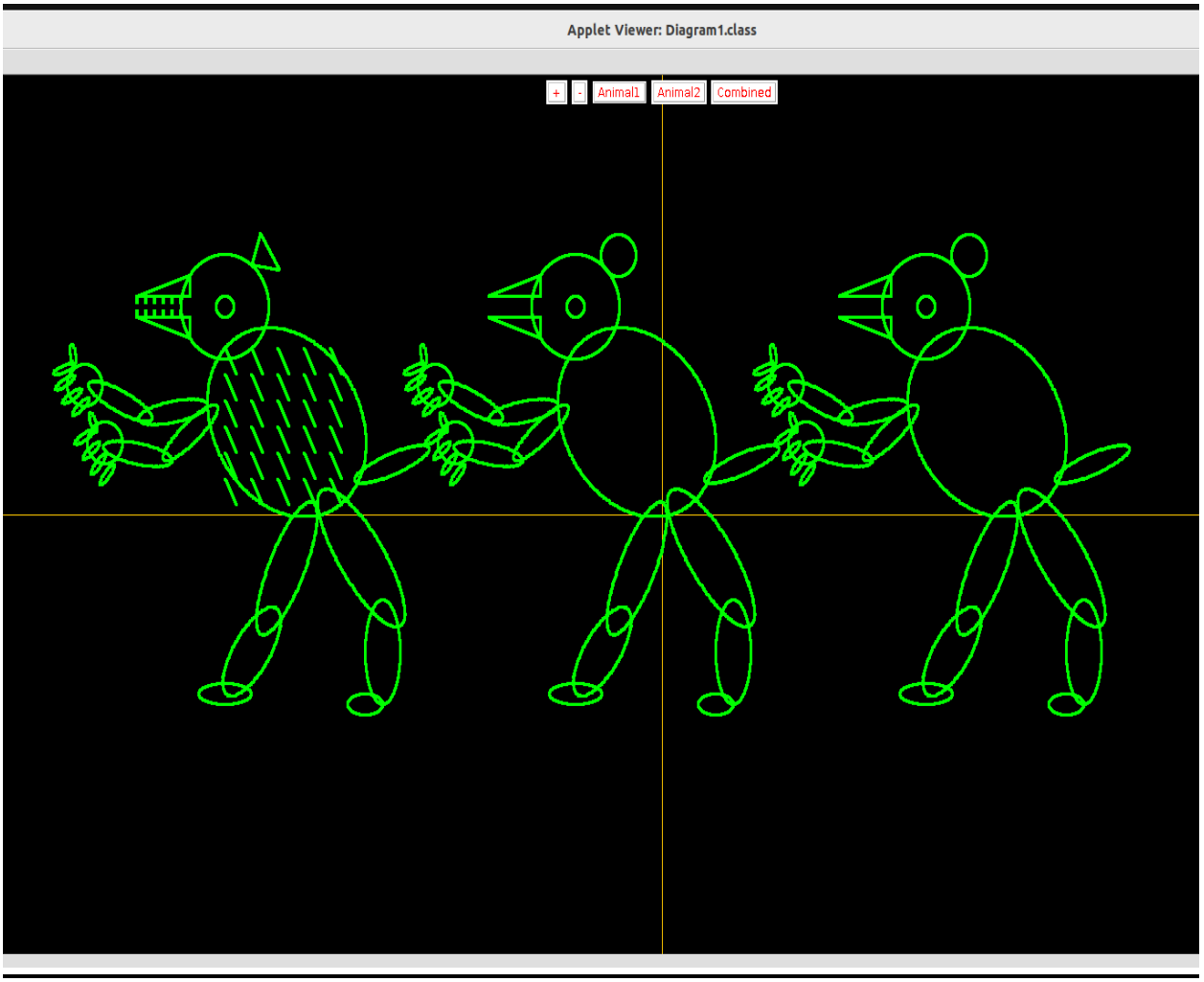
```
<html>
    <head></head>
    <body>
        <applet code ="Diagram1.class"  height="1000"
width="1000"></applet>
    </body>
</html>
```

APPLETVIEWER-

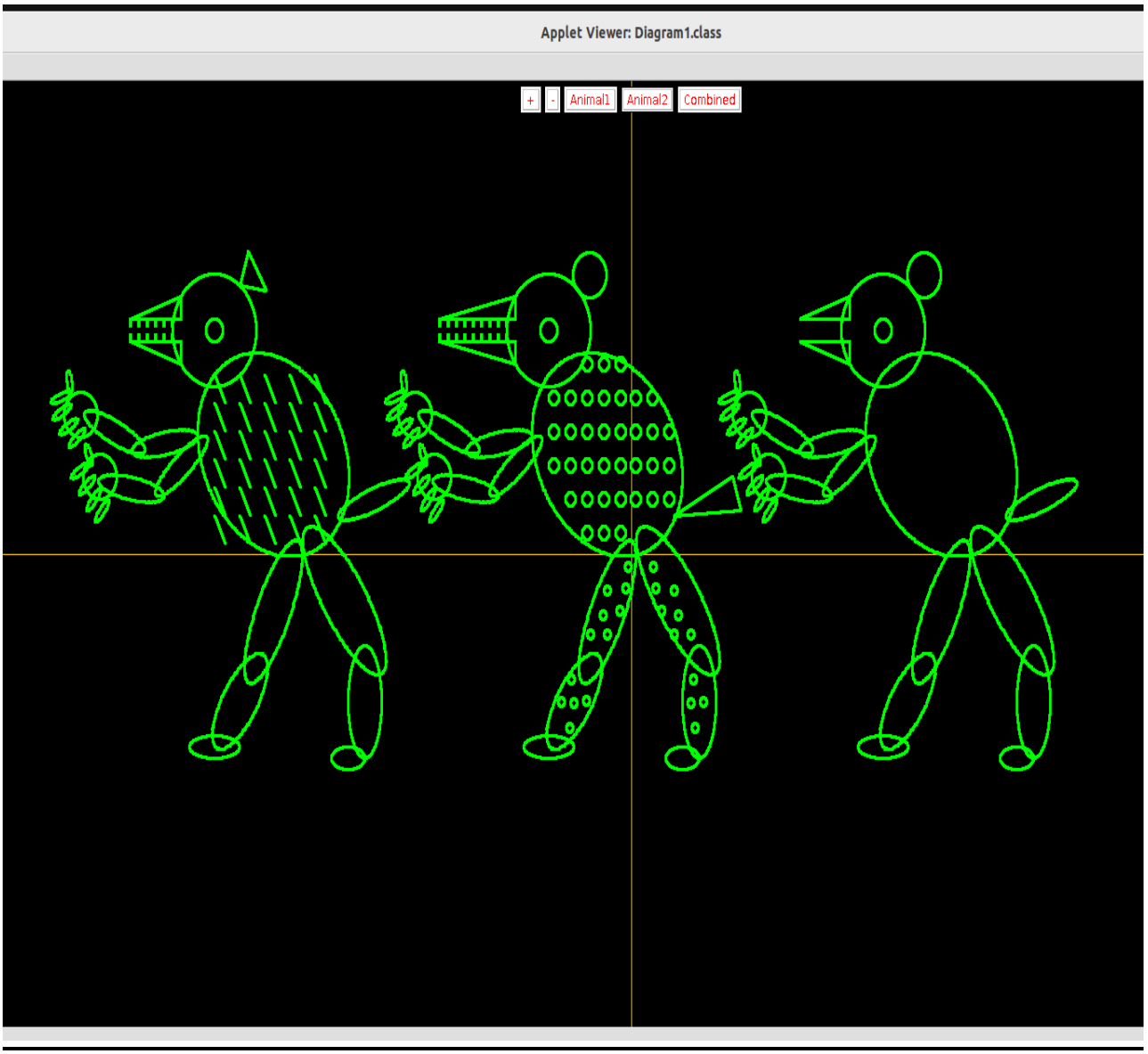
1.Simple Starting View-



2.When applying some property to first animal-



3.When some property applied to second animal-



4.When we combined property of both animal and inherited some propery from animal 1 and some property from animal2 –