Assignment – 4

Name : Bijay kumar sah

Roll : 2020CSB062

G-Suite: 2020csb062.bijay@students.iiests.ac.in

Subject :Computer Graphics

PART 1 :

Code :

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

public class Diagram extends Applet implements ActionListener,
MouseWheelListener {
    int gap = 2;
    int tempteeth = 0;
    int tempear = 0;
    int tspot = 0;
    int ttail = 0;
    int bigb = 0;
    int spotonLegs = 0;
    int hair = 0;
    Button plusbutton = new Button("+");
    Button minusbutton = new Button("-");
    Button butteeth = new Button("Teeth");
    Button butear = new Button("Ear");
    Button buttspots = new Button("Spots on body");
    Button buttail = new Button("Tail");
    Button butlegspot = new Button("SpotonLegs");
    Button beak = new Button("beak");
    Button hairb = new Button("hair");

    // It is for initialisation purpose
    public void init() {
        add(plusbutton);
        add(minusbutton);
        add(butteeth);
        add(butear);
        add(buttspots);
        add(buttail);
        add(butlegspot);
        add(beak);
        add(hairb);
        plusbutton.setBackground(Color.green);
        minusbutton.setBackground(Color.red);
        plusbutton.setForeground(Color.white);
        minusbutton.setForeground(Color.white);
        butteeth.setBackground(Color.black);
        butear.setBackground(Color.black);
        buttspots.setBackground(Color.black);
```

```
        buttail.setBackground(Color.black);

        plusbutton.addActionListener(this);
        minusbutton.addActionListener(this);
        butteeth.addActionListener(this);
        butear.addActionListener(this);
        buttspots.addActionListener(this);
        buttail.addActionListener(this);
        butlegspot.addActionListener(this);
        beak.addActionListener(this);
        hairb.addActionListener(this);
        addMouseWheelListener(this);
        setForeground(Color.green);
        setBackground(Color.black);
    }

    // It is for mouse wheel operation
    public void mouseWheelMoved(MouseWheelEvent e) {
        int z = e.getWheelRotation();
        gap += z;
        if (gap == 0)
            gap = 1000;
        if (gap > 1000)
            gap = 1;
        repaint();
    }

    // it is for implementing button function
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == plusbutton) {
            int z = gap / 2;
            gap += z;
            if (gap > 1000)
                gap = 1;
            if (gap < 2)
                gap = 10;
            repaint();
        }
        if (e.getSource() == minusbutton) {
            int z = gap / 2;
            gap -= z;
            if (gap == 0)
                gap = 1000;
            repaint();
        }
        if (e.getSource() == butteeth) {
            if (tempteeth == 0)
                tempteeth = 1;
```

```
            else
                tempteeth = 0;

            repaint();
        }
        if (e.getSource() == butear) {
            if (tempear == 0)
                tempear = 1;
            else if (tempear == 1)
                tempear = 2;
            else
                tempear = 0;
            repaint();
        }
        if (e.getSource() == buttspots) {
            if (tspot == 0)
                tspot = 1;
            else
                tspot = 0;
            repaint();
        }
        if (e.getSource() == buttail) {
            if (ttail == 0)
                ttail = 1;
            else if (ttail == 1)
                ttail = 2;
            else
                ttail = 0;
            repaint();
        }
        if (e.getSource() == butlegspot) {
            if (spotonLegs == 0)
                spotonLegs = 1;
            else if (spotonLegs == 1)
                spotonLegs = 0;
            repaint();
        }
        if (e.getSource() == beak) {
            if (bigb == 0)
                bigb = 1;
            else if (bigb == 1)
                bigb = 0;
            repaint();
        }
        if (e.getSource() == hairb) {
            if (hair == 0)
                hair = 1;
            else if (hair == 1)
```

```
                hair = 0;
            repaint();
        }

    }

    public void plotPoint(Graphics g, int x, int y, Color c) {
        g.setColor(c);
        g.fillOval(
                (getX() + getWidth()) / 2 + (x * gap) - (gap / 2),
                (getY() + getHeight()) / 2 - (y * gap) - (gap / 2),
                gap, 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(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(radian))), Color.green);

            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) {

            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(radian))), Color.green);


            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.yellow);

        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);

        }
    }

    // function for making spotted cirecles
    public void makespot(Graphics g) {
        if (tspot == 0)
            return;

        Circles(g, 4, -60, 100);
        Circles(g, 4, -30, 100);
        Circles(g, 2, -20, 100);
        Circles(g, 3, -50, 110);
        Circles(g, 2, -40, 120);
        Circles(g, 6, -50, 90);
        Circles(g, 6, -10, 95);

        Circles(g, 6, 0, 85);
        Circles(g,4,20,20);
        Circles(g,4,-20,20);
        Circles(g,4,-40,20);
        Circles(g,5,0,20);
        Circles(g,4,15,50);

        Circles(g,4,0,35);

        Circles(g,8,0,65);
        Circles(g,6,-18,35);
        Circles(g,4,-19,52);
        Circles(g,5,-20,65);
        Circles(g,3,-20,80);
        Circles(g,10,-42,45);
        Circles(g,5,-50,65);


    }

    public void makeSpotOnLegs(Graphics g) {
        if (spotonLegs == 0)
            return;
        // front upper
        Circles(g, 4, -50, -70);
        Circles(g, 4, -30, -70);
        Circles(g, 4, -15, -49);
```

```
        Circles(g, 4, -30, -31);
        Circles(g, 4, -35, -53);
        Circles(g, 4, -5, -10);
        Circles(g, 4, -8, -29);
        // front lower
        Circles(g, 4, -85, -130);
        Circles(g, 4, -70, -131);
        Circles(g, 4, -73, -110);
        Circles(g, 4, -50, -110);
        // //back upper
        Circles(g, 4, 50, -80);
        Circles(g, 4, 40, -70);
        Circles(g, 4, 25, -49);
        Circles(g, 4, 40, -31);
        Circles(g, 4, 45, -53);
        Circles(g, 4, 15, -10);
        Circles(g, 4, 18, -30);
        // //back lower
        Circles(g, 4, 60, -130);
        Circles(g, 4, 44, -131);
        Circles(g, 4, 55, -153);
        Circles(g, 4, 43, -110);

    }

    // function to make ellipse tail
    public void maketail(Graphics g) {
        if (ttail == 0)
            midptellipse(g, (float) 34, (float) 10, (float) 55, (float)
25, (double) 20);
        if (ttail == 2) {
            DDALine(g, 22, 14, 120, 90);
            DDALine(g, 22, 14, 120, 60);
            DDALine(g, 22, 14, 130, 70);
            DDALine(g, 22, 14, 120, 50);
            DDALine(g, 22, 14, 130, 40);
            DDALine(g, 22, 14, 120, 30);
            DDALine(g, 22, 14, 130, 20);
        }
        if (ttail == 1) {
            DDALine(g, 22, 14, 80, 50);
            DDALine(g, 22, 14, 90, 30);
            DDALine(g, 80, 50, 90, 30);
        }
    }

    // function for making circle of ear
    public void makeear(Graphics g) {
```

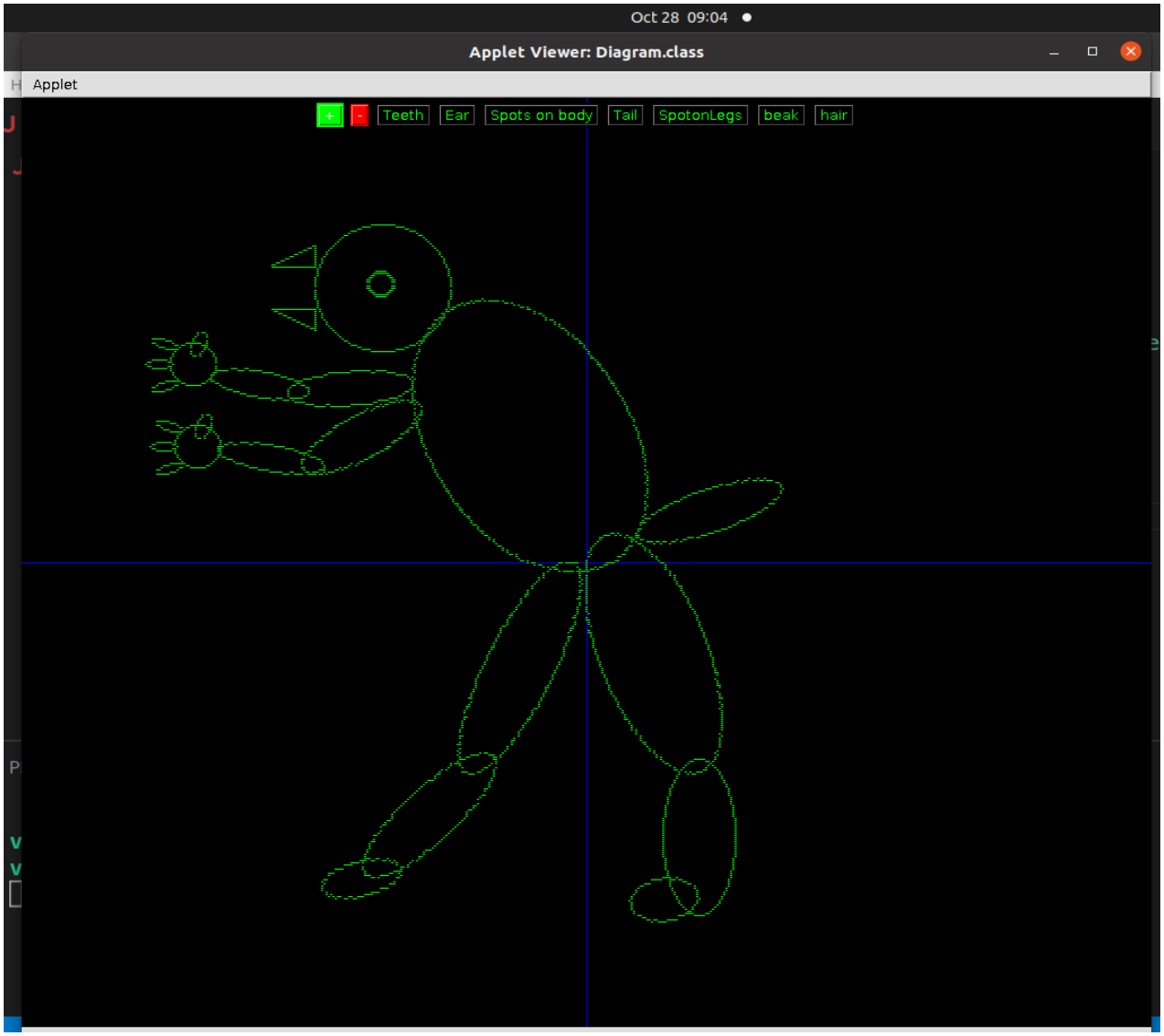
```
        if (tempear == 1)
            Circles(g, 16, -50, 150);
        if (tempear == 2) {
            DDALine(g, -65, 145, -40, 145);
            DDALine(g, -40, 145, -40, 170);
            DDALine(g, -65, 145, -40, 170);
        }
    }

    // function for teeth making
    public void maketeeth(Graphics g, int x1, int x2, int y1, int y2) {
        if (tempteeth == 1) {
            int diff = Math.abs(x1 - x2);
            int len = Math.abs(y1 - y2) / 2;
            int start = Math.min(x1, x2);
            for (int i = 0; i < diff; i += diff / 6) {
                DDALine(g, start + i, y1, start + i, y1 - len + 2);
                DDALine(g, start + i, y2, start + i, y2 + len - 2);
            }

        }
    }

    public void animalBody(Graphics g) {
        g.setColor(Color.blue);
        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);
        // animalbody
        // head
        Circles(g, 30, -90, 130);
        // eye
        Circles(g, 5, -91, 132);
        Circles(g, 6, -91, 132);

        beak(g);
        /* body */
        midptellipse(g, (float) 70, (float) 45, (float) -25, (float)
60, (double) 120);
        /* legs */
        midptellipse(g, (float) 55, (float) 15, (float) -30, (float) -
50, (double) 65);
        midptellipse(g, (float) 60, (float) 24, (float) 30, (float) -
43, (double) 110);
```

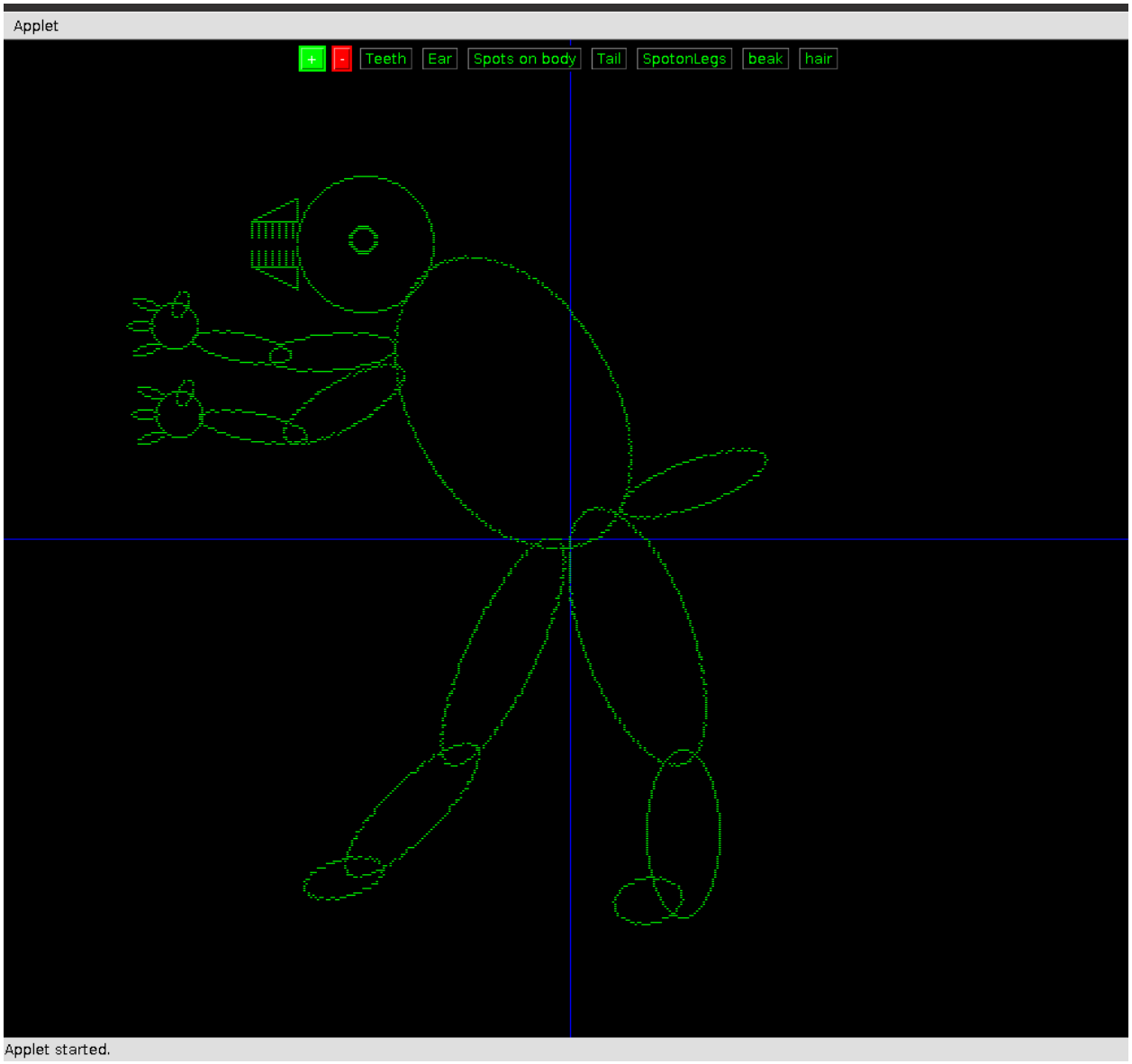
```
        midptellipse(g, (float) 40, (float) 12, (float) -70, (float) -
120, (double) 45);
        midptellipse(g, (float) 16, (float) 37, (float) 50, (float) -
130, (double) 0);
        // foot
        midptellipse(g, (float) 15, (float) 10, (float) 35, (float) -
160, (double) 10);
        midptellipse(g, (float) 18, (float) 8, (float) -100, (float) -
150, (double) 15);

        // arms
        midptellipse(g, (float) 30, (float) 10, (float) -100, (float)
60, (double) 30);
        midptellipse(g, (float) 24, (float) 6, (float) -140, (float)
50, (double) 170);
        midptellipse(g, (float) 28, (float) 8, (float) -105, (float)
83, (double) 5);
        midptellipse(g, (float) 22, (float) 6, (float) -145, (float)
85, (double) 170);

        Circles(g, 10, -174, 94);
        fingers(g, -174, 94);
        Circles(g, 10, -172, 55);
        fingers(g, -172, 55);

        makeear(g);

        makespot(g);
        maketail(g);
        makeSpotOnLegs(g);
        makeHair(g);

    }

    public void fingers(Graphics g, int x, int y) {
        midptellipse(g, 6, 2, x - 15, y, 0.0);
        midptellipse(g, 6, 2, x - 13, y + 10, -20.0);
        midptellipse(g, 6, 2, x - 13, y - 10, 20.0);

        midptellipse(g, 3, 6, x + 2, y + 10, -20.0);
    }

    public void beak(Graphics g) {

        int x1;
        int y1;
        int x2;
        x1 = -120;
```

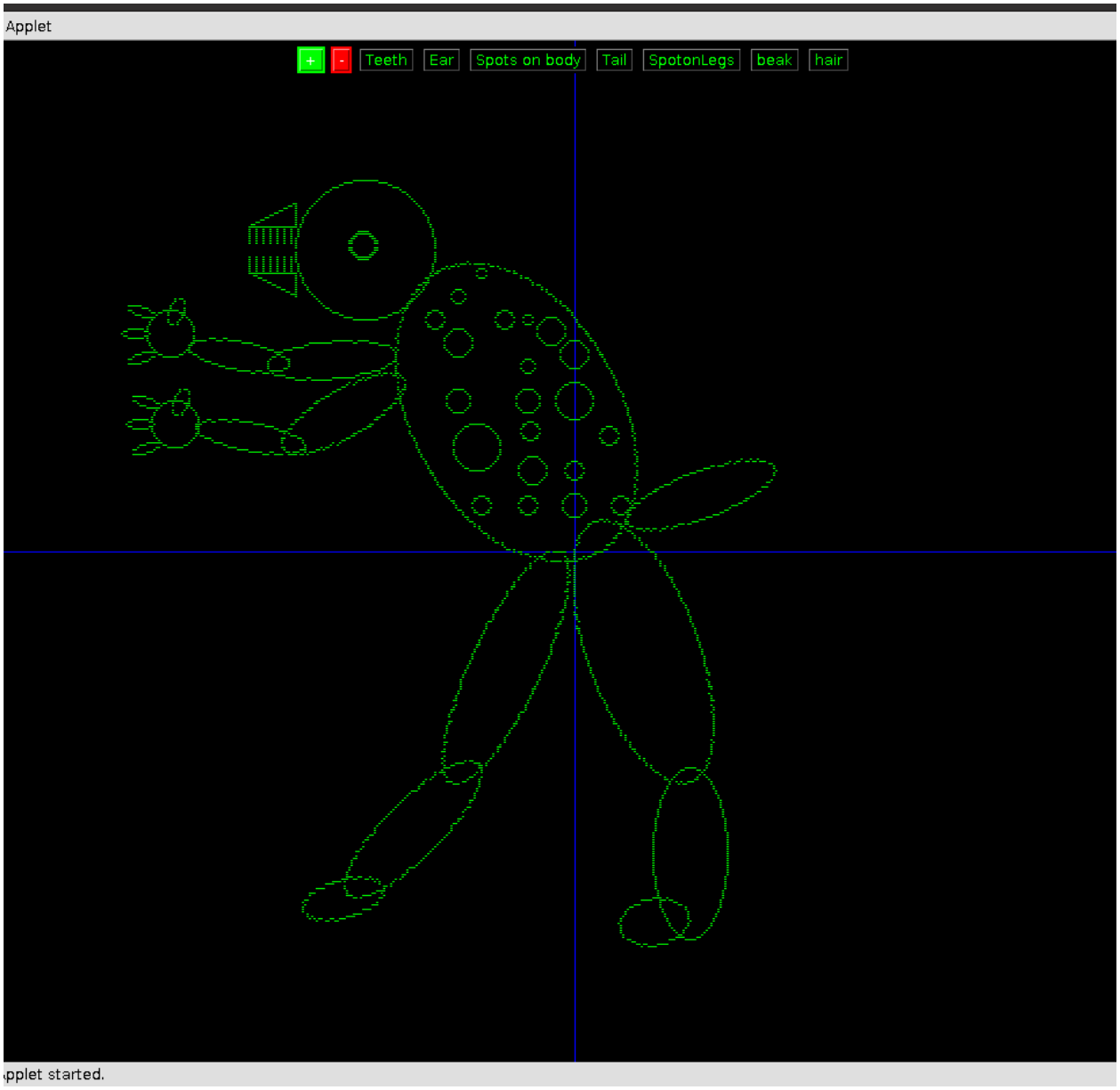
```
        x2 = -150;

        y1 = 140;
        int adv = 0;
        if (bigb == 1) {
            x2 = x2 - 20;
            adv = 5;
        } else {
            x2 = x2 + 10;
        }

        DDALine(g, x1, y1, x2, y1);
        DDALine(g, x1, y1 + 10 + adv, x2, y1);
        DDALine(g, x1, y1 + 10 + adv, x1, y1);

        DDALine(g, x1, y1 - 20, x2, y1 - 20);
        DDALine(g, x1, y1 - 20 - 10 - adv, x1, y1 - 20);
        DDALine(g, x1, y1 - 20 - 10 - adv, x2, y1 - 20);

        maketeeth(g, x1, x2, y1, y1 - 20);

    }
    public void makeHair(Graphics g)
    {
        if(hair ==0)
            return ;
        for(int i = -5;i<0 ;i++){
            // DDALine(g, 0+i*10,10 , 14+i*10, 20);
            // DDALine(g, 0+i*10,30 , 18+i*10, 42);
            // DDALine(g, 0+i*10,50 , 19+i*10, 67);
            // DDALine(g, 0+i*10,70 , 15+i*10, 87);
            // DDALine(g, 0+i*10,90 , 20+i*10, 107);
            int k= 44;
            DDALine(g, 0+i*15+k, 20, 14+i*15+k,10 );
            k-=10;
            DDALine(g, 0+i*15+k, 42, 18+i*15+k,30 );
            k-=10;
            DDALine(g, 0+i*15+k, 67, 19+i*15+k,50 );
            k-=10;
            DDALine(g, 0+i*15+k, 87, 15+i*15+k,70 );
            k-=10;
            DDALine(g, 0+i*15+k,107, 20+i*15+k,90 );
        }

    }
    public void paint(Graphics g) {
        int originx = getX() + getWidth() / 2;
        int originy = getY() + getHeight() / 2;
```

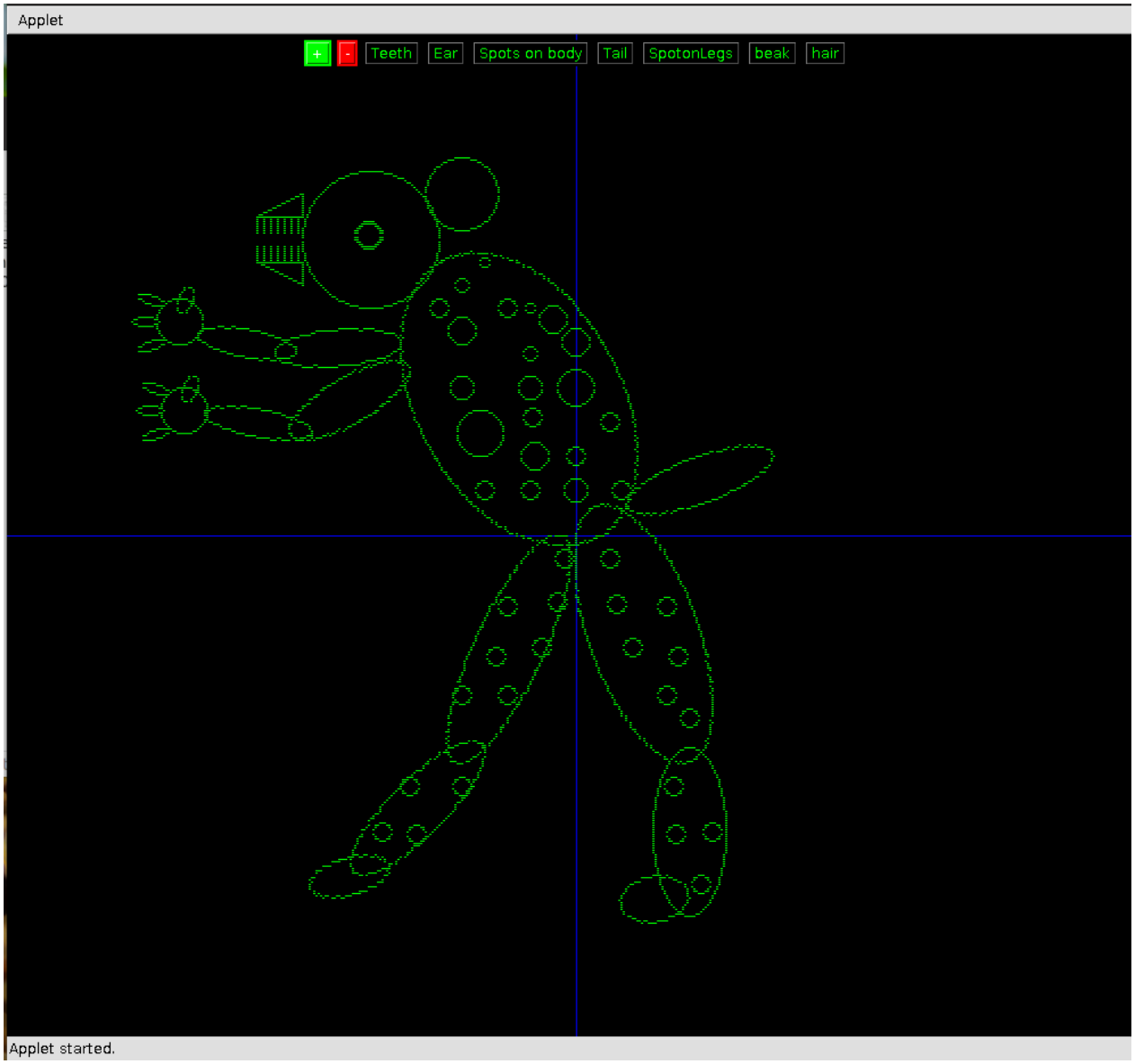
```
        g.drawLine(originx - getWidth() / 2, originy, originx +
getWidth() / 2, originy);
        g.drawLine(originx, originy - getHeight() / 2, originx, originy
+ getHeight() / 2);
        animalBody(g);
    }
}
```

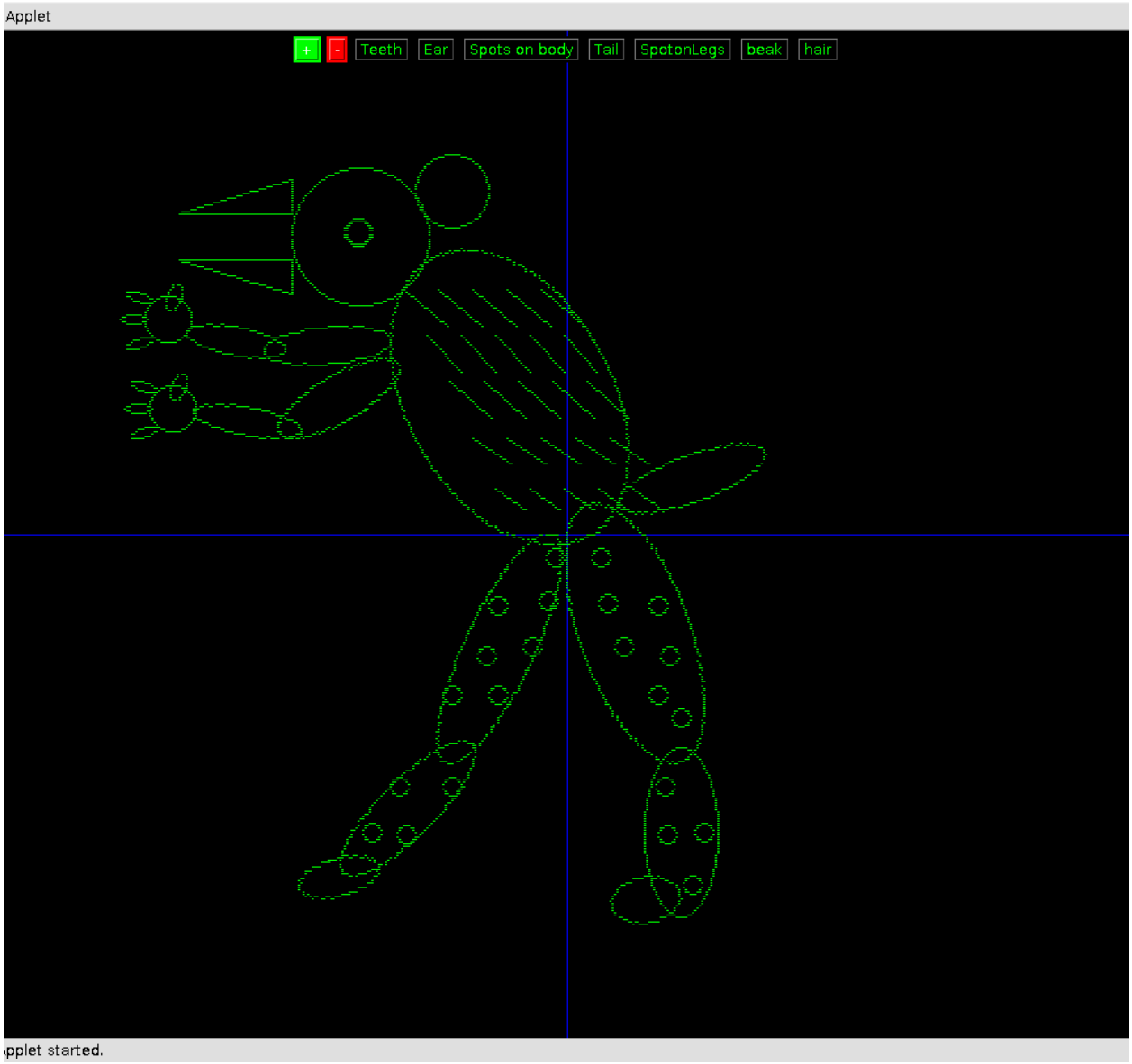
Output : button clicks gives different outputs

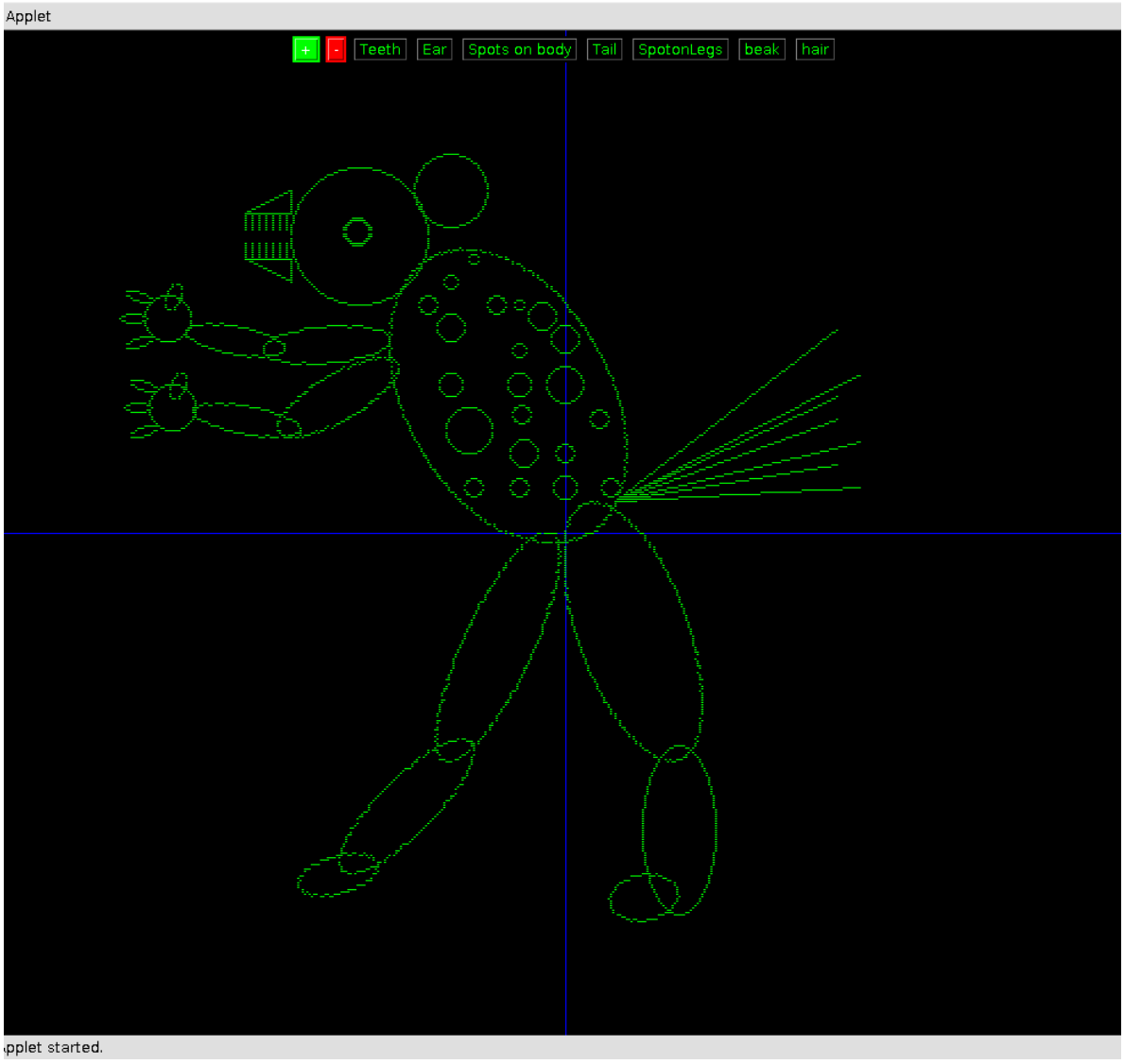


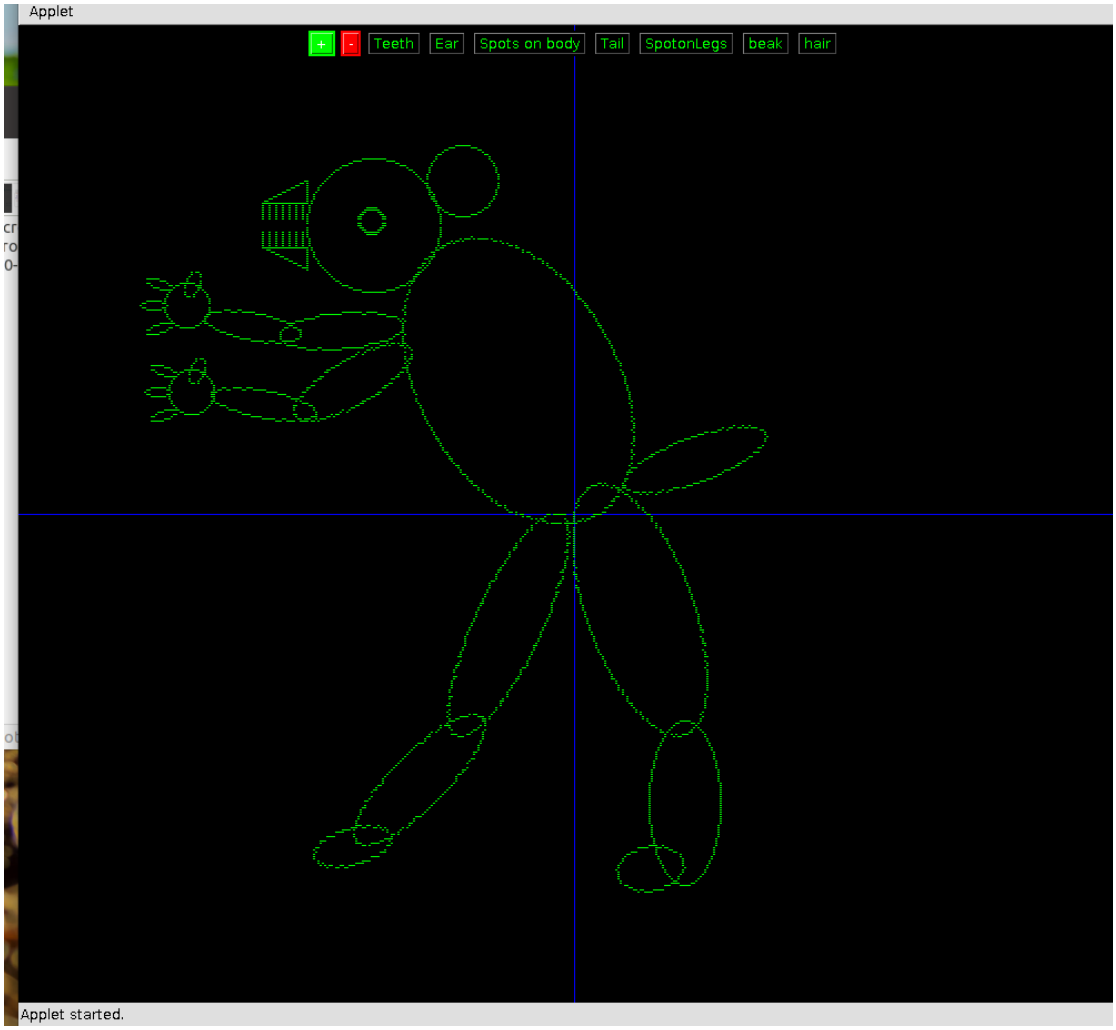

Applet
+
-
Teeth
Ear
Spots on body
Tail
SpotonLegs
beak
hair
Applet started.

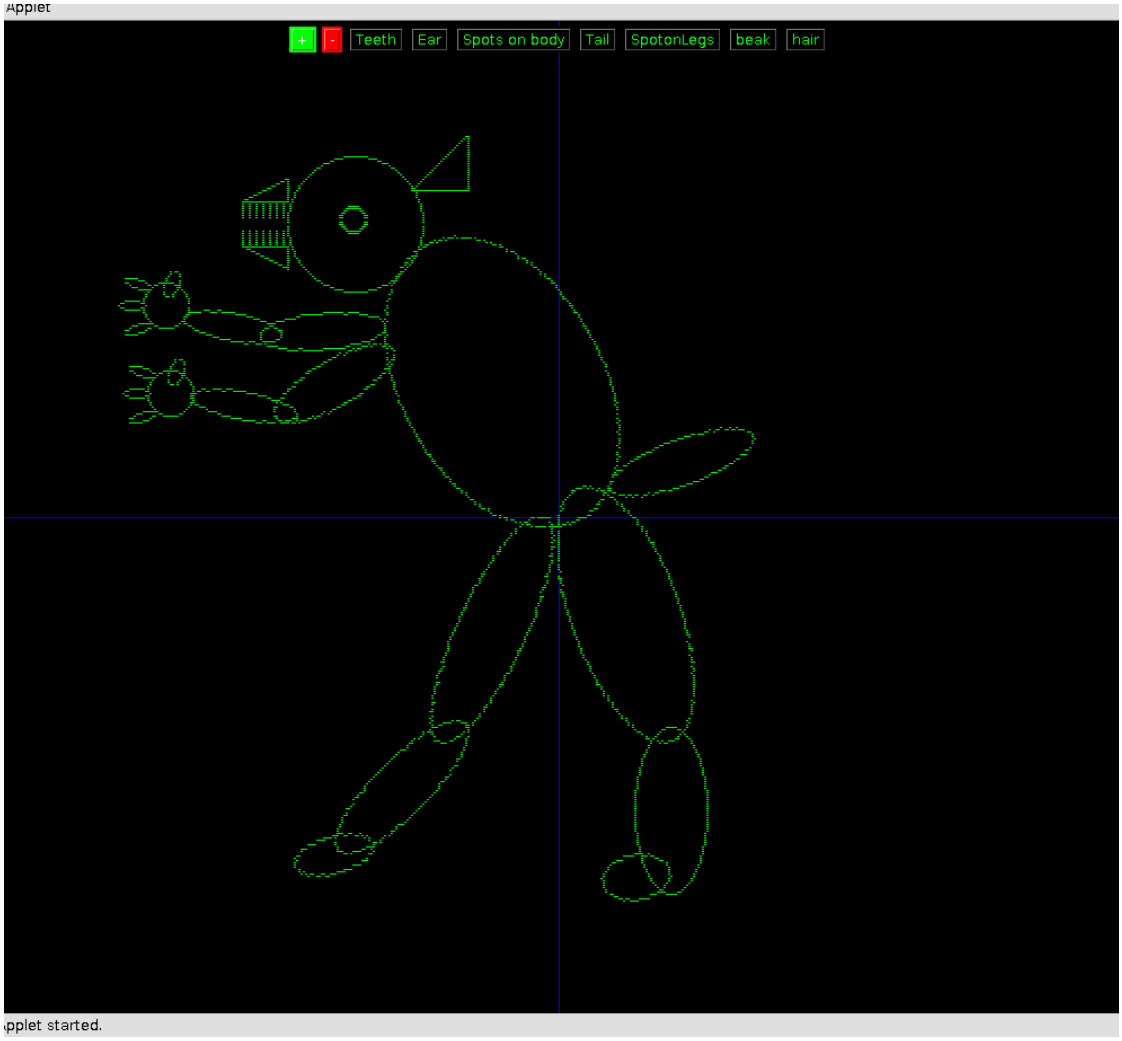

Applet
+
-
Teeth
Ear
Spots on body
Tail
SpotonLegs
beak
hair
pplet started.

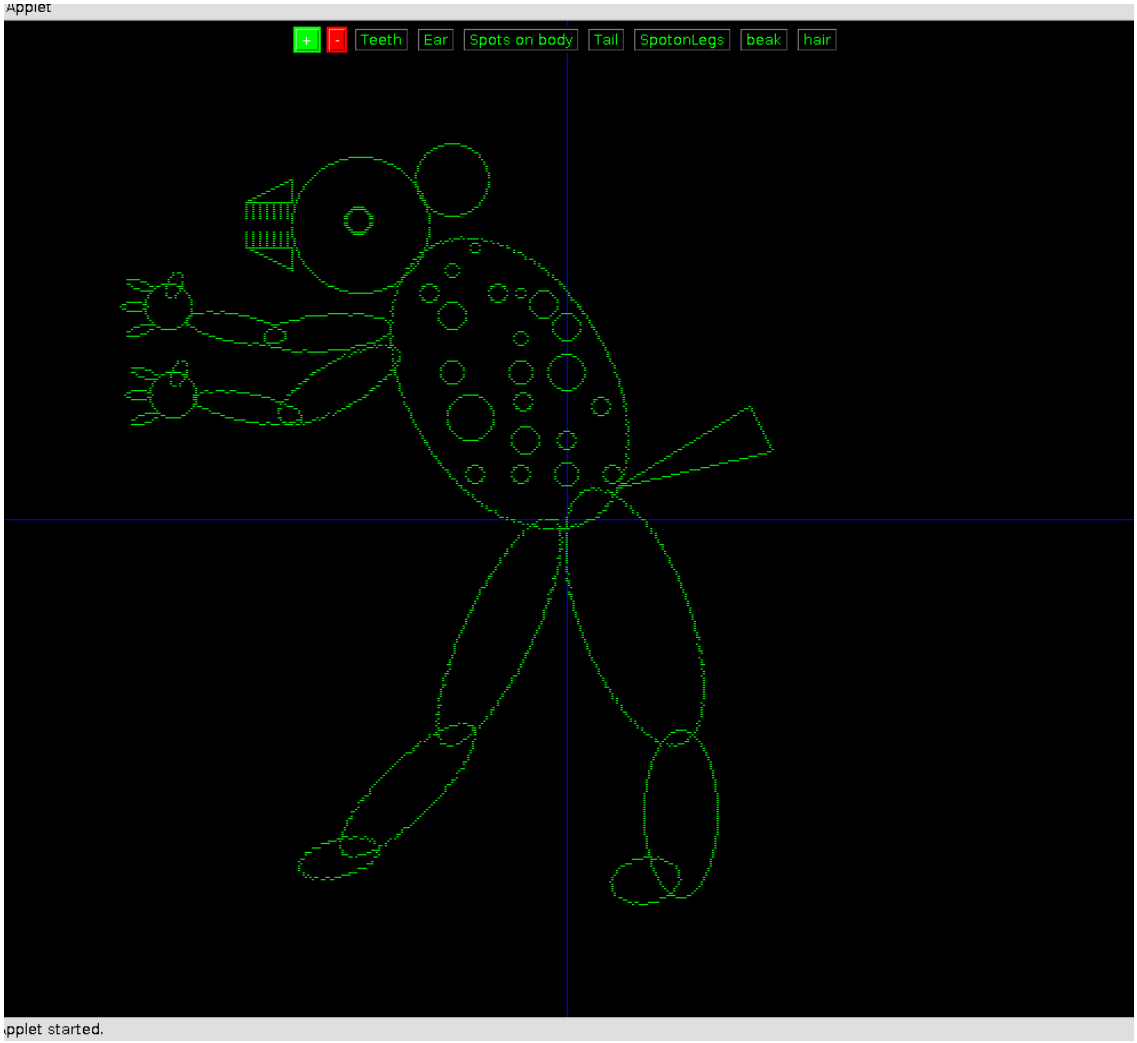

Applet
+
-
Teeth
Ear
Spots on body
Tail
SpotonLegs
beak
hair
Applet started.


Applet
+
-
Teeth
Ear
Spots on body
Tail
SpotonLegs
beak
hair
pplet started.

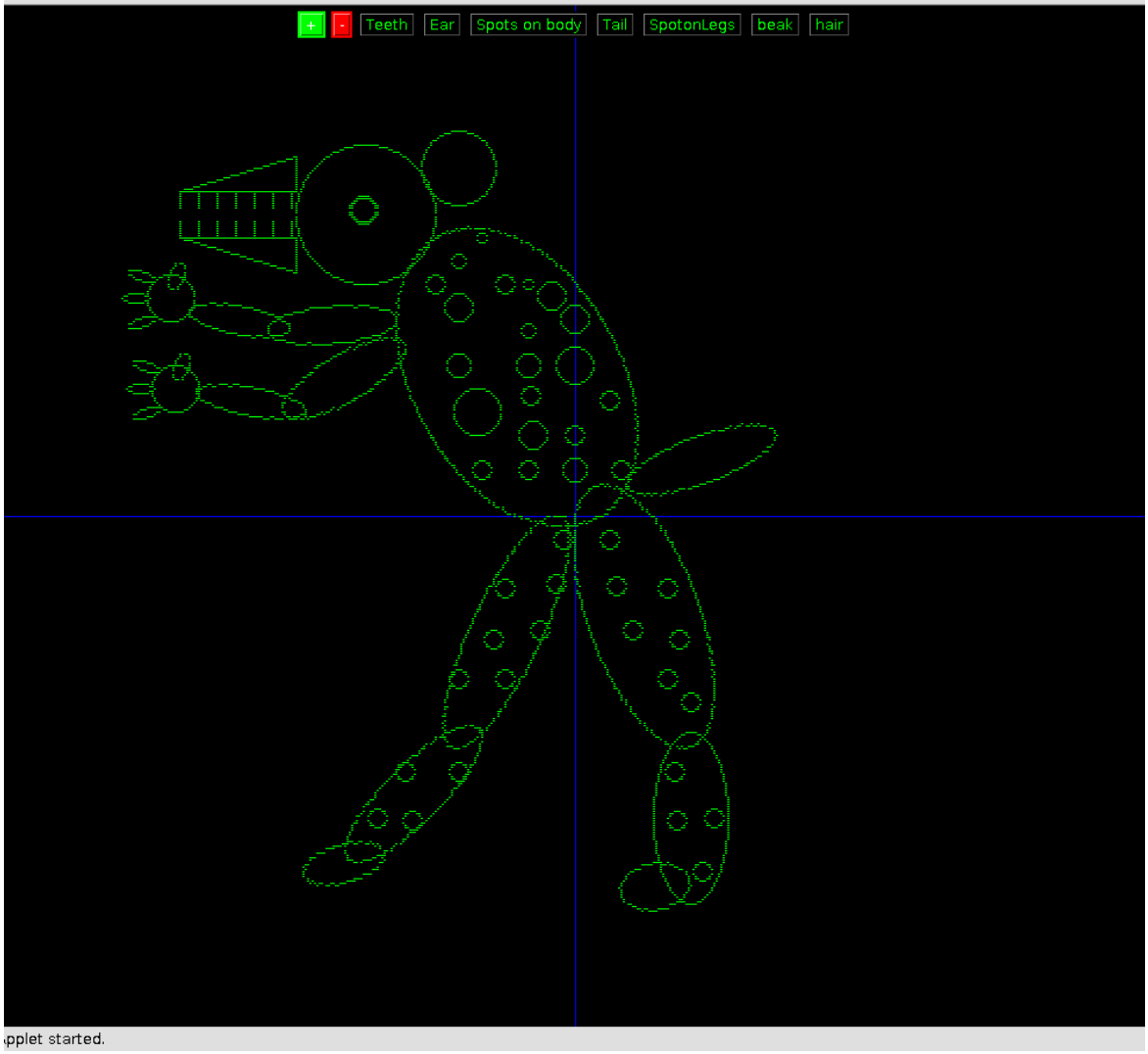

Applet
+
-
Teeth
Ear
Spots on body
Tail
SpotonLegs
beak
hair
pplet started.


Applet
+
-
Teeth
Ear
Spots on body
Tail
SpotonLegs
beak
hair
Applet started.

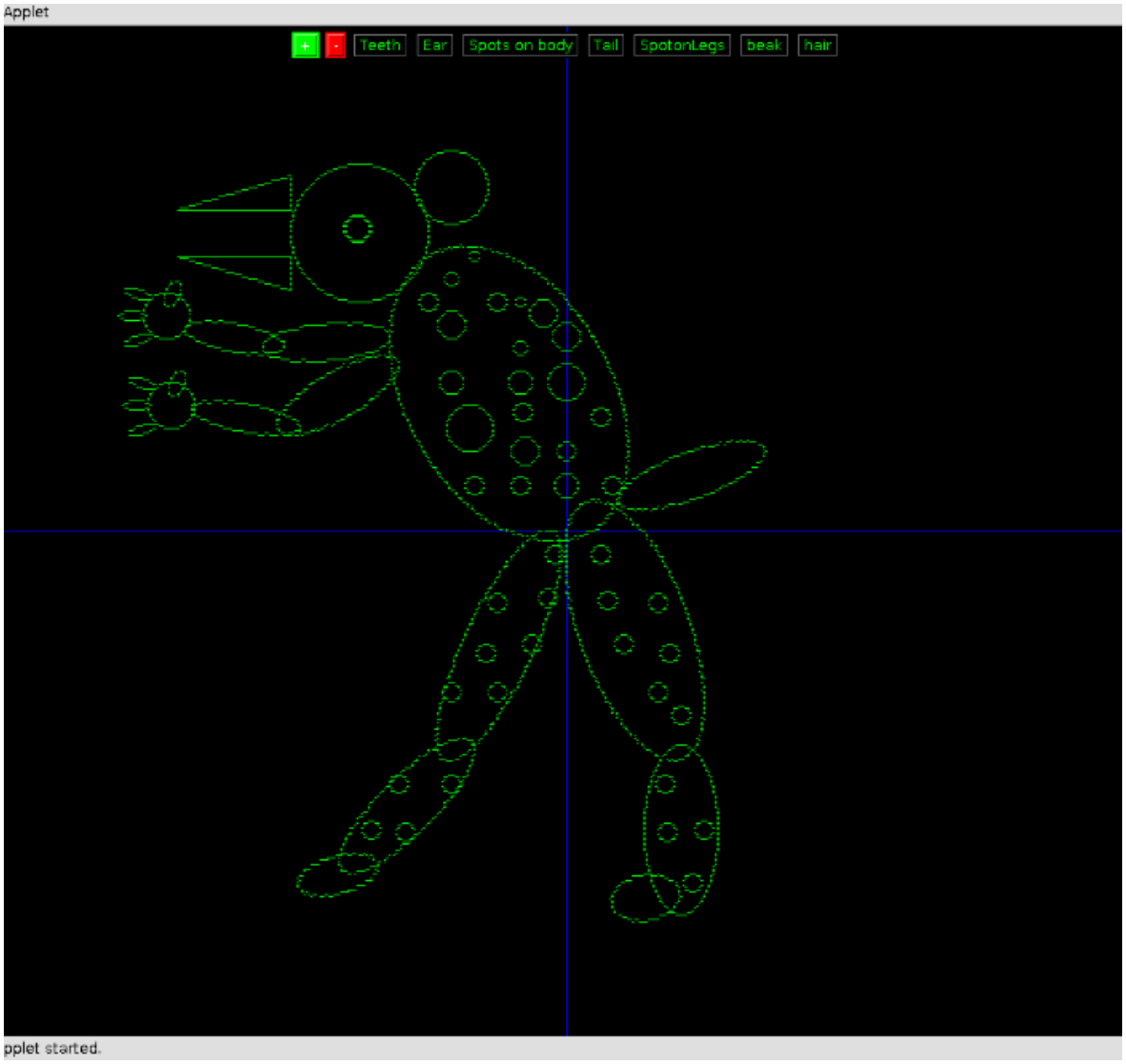

Applet
+
-
Teeth
Ear
Spots on body
Tail
SpotonLegs
beak
hair
pplet started.


Applet
+
-
Teeth
Ear
Spots on body
Tail
SpotonLegs
beak
hair
pplet started.

Applet



pplet started.


Applet
+
-
Teeth
Ear
Spots on body
Tail
SpotonLegs
beak
hair
pplet started.

Part 2:

Code:

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

public class Diagram extends Applet implements
ActionListener,MouseWheelListener {
    int gap = 2;
    int tempteeth = 0;
    int tempear = 0;
    int tspot = 0;
    int ttail = 0;
    int bigb = 0;
    int spotonLegs = 0;
    int hair = 0;
    int shift = 0;
    Button plusbutton = new Button("+");
    Button minusbutton = new Button("-");

    // It is for initialisation purpose
    public void init() {
        add(plusbutton);
        add(minusbutton);

        plusbutton.setBackground(Color.green);
        minusbutton.setBackground(Color.red);

        plusbutton.addActionListener(this);
        minusbutton.addActionListener(this);

        addMouseWheelListener(this);
        setForeground(Color.green);
        setBackground(Color.black);
    }

    // It is for mouse wheel operation
    public void mouseWheelMoved(MouseWheelEvent e) {
        int z = e.getWheelRotation();
        gap += z;
        if (gap == 0)
            gap = 1000;
        if (gap > 1000)
            gap = 1;
```

```
        repaint();
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == plusbutton) {
            int z = gap / 2;
            gap += z;
            if (gap > 1000)
                gap = 1;
            if (gap < 2)
                gap = 10;
            repaint();
        }
        if (e.getSource() == minusbutton) {
            int z = gap / 2;
            gap -= z;
            if (gap == 0)
                gap = 1000;
            repaint();
        }
    }

    public void plotPoint(Graphics g, int x, int y, Color c) {
        g.setColor(c);
        g.fillOval(
                (getX() + getWidth()) / 2 +shift+ (x * gap) - (gap /
2),
                (getY() + getHeight()) / 2 - (y * gap) - (gap / 2),
                gap, 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(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(radian))), Color.green);
```

```
            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) {

            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(radian))), Color.green);


            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.yellow);

        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);

        }
    }

    // function for making spotted cirecles
    public void makespot(Graphics g) {
        if (tspot == 0)
            return;

        Circles(g, 4, -60, 100);
        Circles(g, 4, -30, 100);
        Circles(g, 2, -20, 100);
        Circles(g, 3, -50, 110);
        Circles(g, 2, -40, 120);
        Circles(g, 6, -50, 90);
        Circles(g, 6, -10, 95);

        Circles(g, 6, 0, 85);
        Circles(g,4,20,20);
        Circles(g,4,-20,20);
        Circles(g,4,-40,20);
        Circles(g,5,0,20);
        Circles(g,4,15,50);

        Circles(g,4,0,35);

        Circles(g,8,0,65);
        Circles(g,6,-18,35);
        Circles(g,4,-19,52);
```

```
        Circles(g,5,-20,65);
        Circles(g,3,-20,80);
        Circles(g,10,-42,45);
        Circles(g,5,-50,65);

    }

    public void makeSpotOnLegs(Graphics g) {
        if (spotonLegs == 0)
            return;
        // front upper
        Circles(g, 4, -50, -70);
        Circles(g, 4, -30, -70);
        Circles(g, 4, -15, -49);
        Circles(g, 4, -30, -31);
        Circles(g, 4, -35, -53);
        Circles(g, 4, -5, -10);
        Circles(g, 4, -8, -29);
        // front lower
        Circles(g, 4, -85, -130);
        Circles(g, 4, -70, -131);
        Circles(g, 4, -73, -110);
        Circles(g, 4, -50, -110);
        // //back upper
        Circles(g, 4, 50, -80);
        Circles(g, 4, 40, -70);
        Circles(g, 4, 25, -49);
        Circles(g, 4, 40, -31);
        Circles(g, 4, 45, -53);
        Circles(g, 4, 15, -10);
        Circles(g, 4, 18, -30);
        // //back lower
        Circles(g, 4, 60, -130);
        Circles(g, 4, 44, -131);
        Circles(g, 4, 55, -153);
        Circles(g, 4, 43, -110);

    }

    // function to make ellipse tail
    public void maketail(Graphics g) {
        if (ttail == 0)
            midptellipse(g, (float) 34, (float) 10, (float) 55, (float)
25, (double) 20);
        if (ttail == 2) {
            DDALine(g, 22, 14, 120, 90);
            DDALine(g, 22, 14, 120, 60);
            DDALine(g, 22, 14, 130, 70);
```

```
            DDALine(g, 22, 14, 120, 50);
            DDALine(g, 22, 14, 130, 40);
            DDALine(g, 22, 14, 120, 30);
            DDALine(g, 22, 14, 130, 20);
        }
        if (ttail == 1) {
            DDALine(g, 22, 14, 80, 50);
            DDALine(g, 22, 14, 90, 30);
            DDALine(g, 80, 50, 90, 30);
        }
    }

    // function for making circle of ear
    public void makeear(Graphics g) {
        if (tempear == 1)
            Circles(g, 16, -50, 150);
        if (tempear == 2) {
            DDALine(g, -65, 145, -40, 145);
            DDALine(g, -40, 145, -40, 170);
            DDALine(g, -65, 145, -40, 170);
        }
    }

    // function for teeth making
    public void maketeeth(Graphics g, int x1, int x2, int y1, int y2) {
        if (tempteeth == 1) {
            int diff = Math.abs(x1 - x2);
            int len = Math.abs(y1 - y2) / 2;
            int start = Math.min(x1, x2);
            for (int i = 0; i < diff; i += diff / 6) {
                DDALine(g, start + i, y1, start + i, y1 - len + 2);
                DDALine(g, start + i, y2, start + i, y2 + len - 2);
            }

        }
    }

    public void animalBody(Graphics g) {
        g.setColor(Color.blue);
        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);
        // animalbody
        // head
        Circles(g, 30, -90, 130);
```

```
        // eye
        Circles(g, 5, -91, 132);
        Circles(g, 6, -91, 132);

        beak(g);
        /* body */
        midptellipse(g, (float) 70, (float) 45, (float) -25, (float)
60, (double) 120);
        /* legs */
        midptellipse(g, (float) 55, (float) 15, (float) -30, (float) -
50, (double) 65);
        midptellipse(g, (float) 60, (float) 24, (float) 30, (float) -
43, (double) 110);

        midptellipse(g, (float) 40, (float) 12, (float) -70, (float) -
120, (double) 45);
        midptellipse(g, (float) 16, (float) 37, (float) 50, (float) -
130, (double) 0);
        // foot
        midptellipse(g, (float) 15, (float) 10, (float) 35, (float) -
160, (double) 10);
        midptellipse(g, (float) 18, (float) 8, (float) -100, (float) -
150, (double) 15);

        // arms
        midptellipse(g, (float) 30, (float) 10, (float) -100, (float)
60, (double) 30);
        midptellipse(g, (float) 24, (float) 6, (float) -140, (float)
50, (double) 170);
        midptellipse(g, (float) 28, (float) 8, (float) -105, (float)
83, (double) 5);
        midptellipse(g, (float) 22, (float) 6, (float) -145, (float)
85, (double) 170);

        Circles(g, 10, -174, 94);
        fingers(g, -174, 94);
        Circles(g, 10, -172, 55);
        fingers(g, -172, 55);

        makeear(g);

        makespot(g);
        maketail(g);
        makeSpotOnLegs(g);
        makeHair(g);

    }
    public void fingers(Graphics g, int x, int y) {
```

```
        midptellipse(g, 6, 2, x - 15, y, 0.0);
        midptellipse(g, 6, 2, x - 13, y + 10, -20.0);
        midptellipse(g, 6, 2, x - 13, y - 10, 20.0);

        midptellipse(g, 3, 6, x + 2, y + 10, -20.0);
    }
    public void beak(Graphics g) {

        int x1;
        int y1;
        int x2;
        x1 = -120;
        x2 = -150;

        y1 = 140;
        int adv = 0;
        if (bigb == 1) {
            x2 = x2 - 20;
            adv = 5;
        } else {
            x2 = x2 + 10;
        }

        DDALine(g, x1, y1, x2, y1);
        DDALine(g, x1, y1 + 10 + adv, x2, y1);
        DDALine(g, x1, y1 + 10 + adv, x1, y1);

        DDALine(g, x1, y1 - 20, x2, y1 - 20);
        DDALine(g, x1, y1 - 20 - 10 - adv, x1, y1 - 20);
        DDALine(g, x1, y1 - 20 - 10 - adv, x2, y1 - 20);

        maketeeth(g, x1, x2, y1, y1 - 20);

    }
    public void makeHair(Graphics g)
    {
        if(hair ==0)
            return ;
        for(int i = -5;i<0 ;i++){
            // DDALine(g, 0+i*10,10 , 14+i*10, 20);
            // DDALine(g, 0+i*10,30 , 18+i*10, 42);
            // DDALine(g, 0+i*10,50 , 19+i*10, 67);
            // DDALine(g, 0+i*10,70 , 15+i*10, 87);
            // DDALine(g, 0+i*10,90 , 20+i*10, 107);
            int k= 44;
            DDALine(g, 0+i*15+k, 20, 14+i*15+k,10 );
            k-=10;
            DDALine(g, 0+i*15+k, 42, 18+i*15+k,30 );
```

```
            k-=10;
            DDALine(g, 0+i*15+k, 67, 19+i*15+k,50 );
            k-=10;
            DDALine(g, 0+i*15+k, 87, 15+i*15+k,70 );
            k-=10;
            DDALine(g, 0+i*15+k,107, 20+i*15+k,90 );
        }

    }
    public void setVars(int arr[])
    {
        tempteeth = arr[0];
        tempear = arr[1];
        tspot = arr[2];
        ttail = arr[3];
        bigb = arr[4];
        spotonLegs = arr[5];
        hair = arr[6];
        shift = arr[7];
    }
    public void setChild(int p1[],int p2[],int c[])
    {
        c[0] = p1[0];
        c[1] = p1[1];
        c[2] = p1[2];
        c[3] = p1[3];

        c[4] = p2[4];
        c[5] = p2[5];
        c[5] = p2[5];
        c[6] = p2[6];
    }
    public void paint(Graphics g)
    {
        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);
        int parent1[] = {1,1,1,1 , 0,0,0,0};
        int parent2[] = {0,0,0,0 , 1,1,1,1};
        int child[] = new int[8];

        setVars(parent1);
        //parent 1
        shift=-500;
        animalBody(g);
```

```
        setVars(parent2);
        //parent 2
        shift = 0;
        animalBody(g);

        setChild(parent1,parent2,child);
        setVars(child);
        //child
        shift = 500;
        animalBody(g);

    }
}
```

Output: