ASSIGNMENT-2

Name-Anubhav Anand

Enrollment Number-2020CSB102

Subject- Assignment -2 of Computer Graphics

G-suite Id-
2020CSB102.anubhav@students.iiests.ac.in

1>Q. Part- I:

1. Draw straight line using the following line drawing methods keeping the same grid structure in order

to view resolution for each case.

i) DDA ii) Bresenham's iii) Midpoint

Ans-Line Drawing Algorithm Using Midpoint Theorem-

Midpoint.java-

```
/*This is the program for zoom-out and zoom-in using graphics */
/*Thus program can run only on Java-jdk(8) */


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

public class Midpoint extends Applet implements
ActionListener,MouseWheelListener{
    //It is for generating a rectangle corresponding to a particular
point in cartesian coordinate syatem
        int gap = 4;
    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),
            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;
  }
  //function to return line
  public void midpoint(Graphics g,int X1,int Y1,int X2,int Y2)
  {
       int dx = X2 - X1;
  int dy = Y2 - Y1;
  double m=(double)dy/dx;
  if(m>=0)
  {
  if(Math.abs(dy)<=Math.abs(dx)){
  // initial value of decision parameter d
  double d = (double)1/2-(m);
  int x=X1,y=Y1;
  plotPoint(g,x,-y,Color.green);

  // iterate through value of X
  while (x < X2)
  {
      x++;

      // E or East is chosen
      if (d < 0)
      {
          d = d + 1-(m);
          y++;
      }

      // NE or North East is chosen
      else
      {
          d =d-(m);
      }
      plotPoint(g,x,y,Color.green);
  }
  }


else if(Math.abs(dx)<Math.abs(dy))
{
  // initial value of decision parameter d
  double d = 1-(double)(m/2);
  int x=X1,y=Y1;
```

```
   plotPoint(g,x,y,Color.green);

   // iterate through value of X
   while (y < Y2)
   {
       y++;

       // E or East is chosen
       if (d < 0)
       {
           d = d + 1;
           x++;
       }

       // NE or North East is chosen
       // NE or North East is chosen
       else
       {
           d += 1-(m);
       }
       plotPoint(g,x,y,Color.green);
   }
}
   }
else
{
    if(Math.abs(dy)<=Math.abs(dx)){
   // initial value of decision parameter d
   double d = (double)1/2+(m);
   int x=X1,y=Y1;
   plotPoint(g,x,y,Color.green);

   // iterate through value of X
   while (x > X2)
   {
       x--;

       // E or East is chosen
       if (d < 0)
       {
           d = d + 1+(m);
           y++;
       }

       // NE or North East is chosen
       else
       {
           d =d+(m);
```

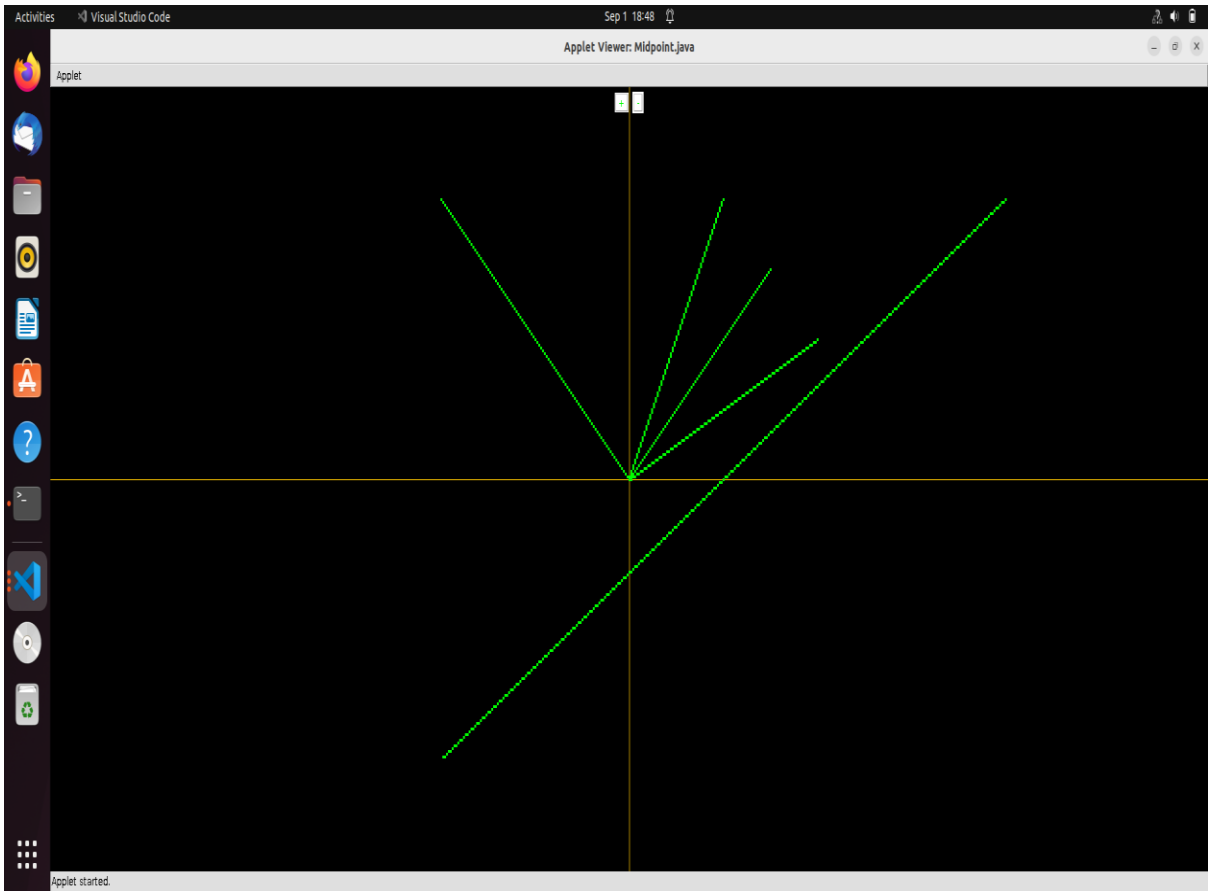
```
        }
        plotPoint(g,x,y,Color.green);
    }
    }


  else
  {
    // initial value of decision parameter d
    double d = 1+(double)(m/2);
    int x=X1,y=Y1;
    plotPoint(g,x,y,Color.green);

    // iterate through value of X
    while (y > Y2)
    {
        y--;

        // E or East is chosen
        if (d < 0)
        {
            d = d + 1;
            x++;
        }
        else
        {
            d += 1+(m);
        }
        plotPoint(g,x,y,Color.green);
    }
  }
  }

}
    //It is for initialisation purpose
    public void init(){
        addMouseWheelListener(this);
        button1 = new Button("+");
        add(button1);
        button1.addActionListener(this);
        button2 = new Button("-");
        add(button2);
        button1.setBackground(Color.white);
        button2.setBackground(Color.white);
        button2.addActionListener(this);
        setForeground(Color.green);
        setBackground(Color.black);
    }
```

```
    //it is for implementing button function
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == button1){
         gap+=gap+gap/10;
         repaint();
        }
        else if(e.getSource()==button2)
            {
             gap-=gap/10;
             repaint();
            }
    }
    //It is for mouse wheel operation
    public void mouseWheelMoved(MouseWheelEvent e)
    {
        int z=e.getWheelRotation();
        gap+=z;
        repaint();
    }

    Button button1, button2;
    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;
            // midpoint(g,1,2,200,300);
            // midpoint(g,1,2,300,200);
            int x1=200,y1=101;
            midpoint(g,0,0,50,100);
            midpoint(g,0,0,100,50);
            midpoint(g,0,0,75,75);
            midpoint(g,-100,-100,200,100);
            midpoint(g,0,0,-100,100);
            midpoint(g,0,0,-100,100);
    }
}
```

Midpoint.html-

```
<html>
    <head></head>
    <body>
        <applet code ="Midpoint.java"  height="500"
width="500"></applet>
    </body>
</html>
```

Applet View(Mid Point)-



Line Drawing Algorithm Using DDA Theorem-

DDA.java-

```
/*This is the program for zoom-out and zoom-in using graphics */
/*Thus program can run only on Java-jdk(8) */


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

public class DDA extends Applet implements
ActionListener,MouseWheelListener{
    //It is for generating a rectangle corresponding to a particular
point in cartesian coordinate syatem
        int gap = 50;
    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),
            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);
}
    //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;
```

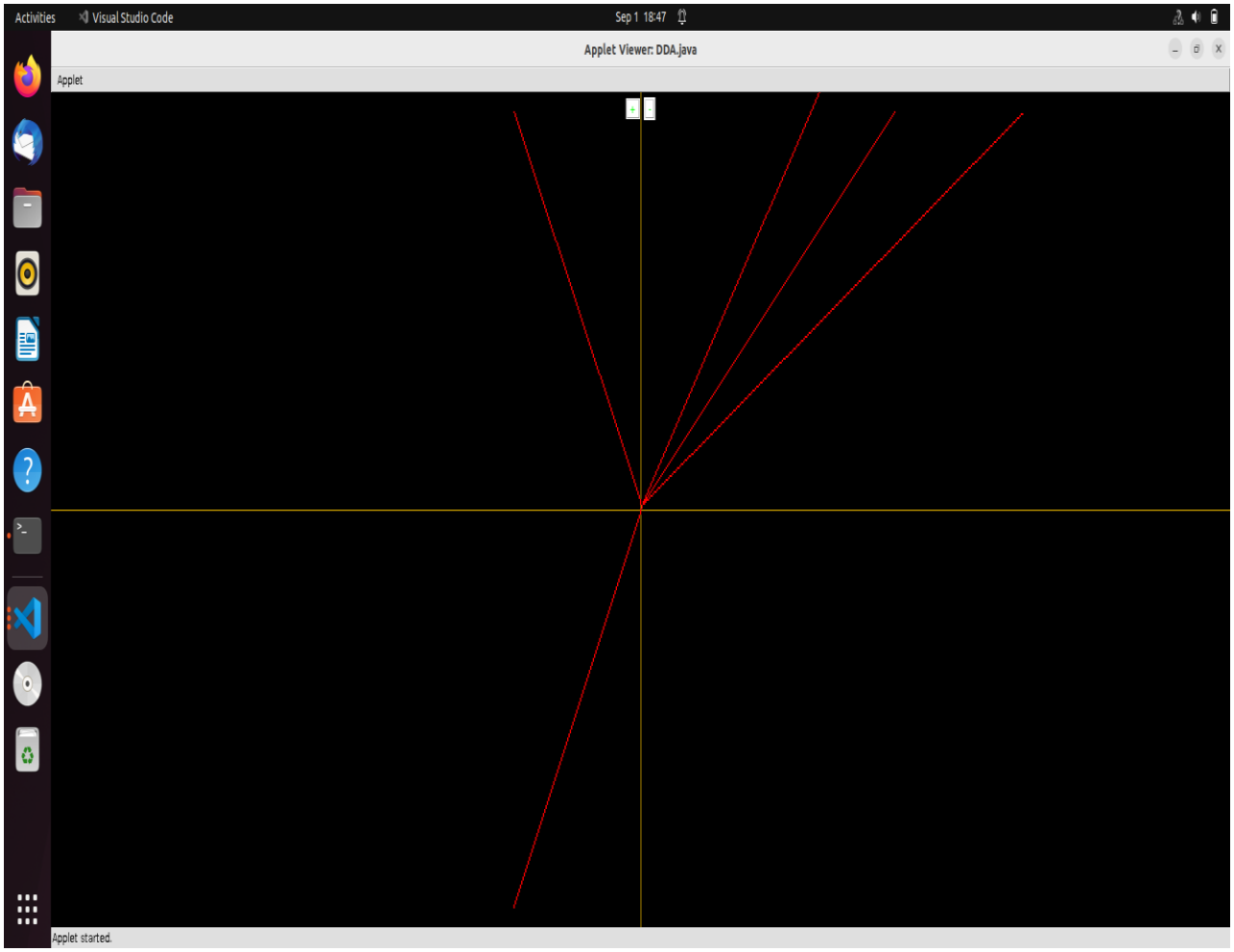
```
    int dy= y1 - y0;

    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.red);
        x += x_incr;
        y += y_incr;
        //delay(10);
    }
}
    //It is for initialisation purpose
    public void init(){
        setBackground(Color.black);
        addMouseWheelListener(this);
        button1 = new Button("+");
        add(button1);
        button1.addActionListener(this);
        button2 = new Button("-");
        add(button2);
        button1.setBackground(Color.white);
        button2.setBackground(Color.white);
        button2.addActionListener(this);
        setForeground(Color.green);
        // setBackground(Color.black);
    }
    //it is for implementing button function
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == button1){
         gap+=gap+gap/10;
         repaint();
```

```
        }
        else if(e.getSource()==button2)
            {
             gap-=gap/10;
             repaint();
            }
    }
    //It is for mouse wheel operation
    public void mouseWheelMoved(MouseWheelEvent e)
    {
        int z=e.getWheelRotation();
        gap+=z;
        repaint();
    }

    Button button1, button2;
    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);
            DDALine(g,1,2,200,300);
            DDALine(g,1,2,300,200);
            DDALine(g,1,2,200,201);
            DDALine(g,1,2,-100,-201);
            DDALine(g,1,2,-100,201);
    }
}
```

DDA.html-

```
<html>
    <head></head>
    <body>
        <applet code ="DDA.java"  height="500" width="500"></applet>
    </body>
</html>
```

Applet View(DDA)-



Line Drawing Algorithm Using Bresenham Theorem-

Bresenham.java-

```
/*This is the program for zoom-out and zoom-in using graphics */
/*Thus program can run only on Java-jdk(8) */


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

public class Bresenham extends Applet implements
ActionListener,MouseWheelListener{
    //It is for generating a rectangle corresponding to a particular
point in cartesian coordinate syatem
        int gap = 50;
    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),
            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);
}
    //function to return line
    public void plotLineBresenham(Graphics g, int x1,int y1,int x2,int
y2)
    {
        int x = x1;
        int y = y1;
```

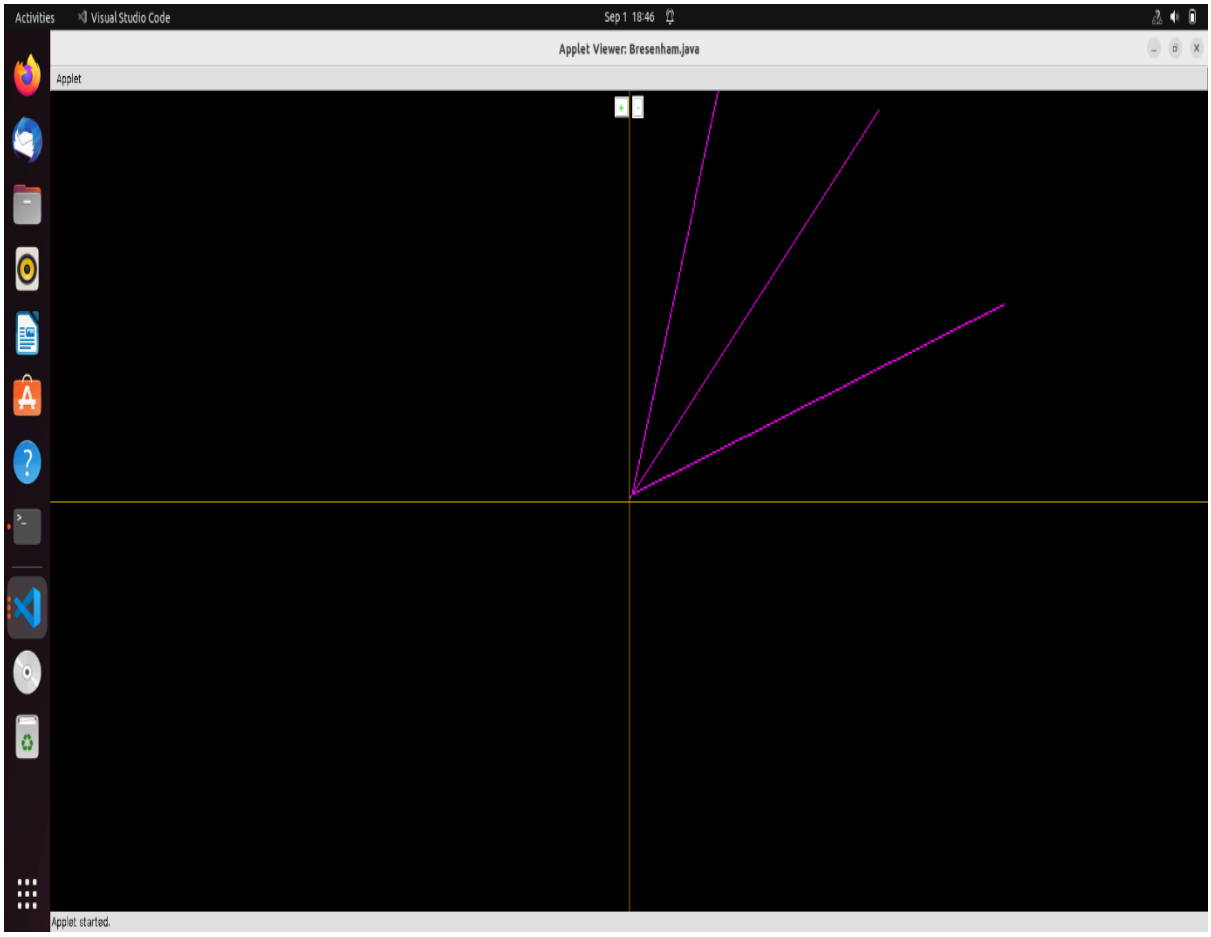
```
        int dy = y2 - y1;
        int dx = x2-x1;
        if(Math.abs(dx)>=Math.abs(dy)){
            int p = 2*dx - dy;
        while(x<x2)
        {
            plotPoint(g,x,y, Color.MAGENTA);
            x++;
            if(p<0)
                p = p+ 2*dy;
            else{
                p = p+2*dy-2*dx;
                y++;
            }
        }
        }
        else{
            int p = 2*dy - dx;
        while(y<y2)
        {
            plotPoint(g,x,y, Color.MAGENTA);
            y++;
            if(p<0)
                p = p+ 2*dx;
            else{
                p = p+2*dx-2*dy;
                x++;
            }
        }
        }
    }
    //It is for initialisation purpose
    public void init(){
        addMouseWheelListener(this);
        button1 = new Button("+");
        add(button1);
        button1.addActionListener(this);
        button2 = new Button("-");
        add(button2);
        button1.setBackground(Color.white);
        button2.setBackground(Color.white);
        button2.addActionListener(this);
        setForeground(Color.green);
        setBackground(Color.black);
    }
    //it is for implementing button function
    public void actionPerformed(ActionEvent e)
    {
```

```
        if (e.getSource() == button1){
         gap+=gap+gap/10;
         repaint();
        }
        else if(e.getSource()==button2)
            {
             gap-=gap/10;
             repaint();
            }
    }
    //It is for mouse wheel operation
    public void mouseWheelMoved(MouseWheelEvent e)
    {
        int z=e.getWheelRotation();
        gap+=z;
        repaint();
    }

    Button button1, button2;
    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);
            plotLineBresenham(g,1,2,100,300);
            plotLineBresenham(g,1,2,300,100);
            plotLineBresenham(g,1,2,200,201);
    }
}
```

Bresenham.html-

```
<html>
    <head></head>
    <body>
        <applet code ="Bresenham.java"  height="500"
width="500"></applet>
    </body>
</html>
```

Applet View(Bresenham)-



2. (a) Prepare a class ‘Fire’ following instructions below.

i. Fire (Fig. 2) is created by collection of straight lines which are very closed together.

ii. Use any line drawing algorithm that is implemented in Part-I, Assignment 2.

iii. Height of the straight lines change over time by changing endpoints away from the source of fire

iv. Colour of fire may vary as the flame is away from the source.

(b) Hence create a class ‘Candle’ (Fig. 3) having at least two methods light_candle () and

put_out_candle ()

Ans-

Candle.java-

```
/*This is the program for zoom-out and zoom-in using graphics */
/*Thus program can run only on Java-jdk(8) */


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

public class Candle extends Applet implements
ActionListener,MouseWheelListener{
    //It is for generating a rectangle corresponding to a particular
point in cartesian coordinate syatem
        int gap = 4;
        int flame = 0;
    public void plotPoint(Graphics g,int x,int y,Color c)
    {
        int r=255,gr=255,b=255;
        gr= 255-(x*x +y*y);
        if(gr<=0){
            b = 255 +gr/8;
             gr=0;
        }
        if(b<=0)
            b= 0;

        c = new Color(r,b,gr);
        g.setColor(c);
        if(y>=x*x/10)
            g.fillOval(
                (getX()+getWidth())/2+(x*gap)-(gap/2),
                (getY()+getHeight())/2-(y*gap)-(gap/2),
                gap,gap
            );
    }
    public void plotRect(Graphics g,int x,int y,Color C)
    {
        g.setColor(Color.orange);
        g.fillRect(
            (getX()+getWidth())/2-50,
            (getY()+getHeight())/2,
            100,300);
    }
    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);
}
    //function to return line
    public void DDALine(Graphics g,int x0, int y0, int x1, int y1,Color
c) {

      //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), c);
        x += x_incr;
        y += y_incr;
        //delay(10);
    }
}
    //It is for initialisation purpose
    public void init(){
        setBackground(Color.black);
        addMouseWheelListener(this);
        button1 = new Button("ON");
        add(button1);
```

```
        button1.addActionListener(this);
        button2 = new Button("OFF");
        add(button2);
        button1.setBackground(Color.white);
        button2.setBackground(Color.white);
        button2.addActionListener(this);
        setForeground(Color.green);
        // setBackground(Color.black);
    }
    //it is for implementing button function
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == button1){
         lightCandle();
        }
        else if(e.getSource()==button2)
            {
                putOutCandle();
            }
    }
    //It is for mouse wheel operation
    public void mouseWheelMoved(MouseWheelEvent e)
    {
        int z=e.getWheelRotation();
        gap+=z;
        repaint();
    }

    Button button1, button2;
    public void infinite(Graphics g)
    {

        int x1=-200,x2=200;
        int a=400;
        Color c1;
        while(x1!=x2)
        {
            if(a-(x1*x1)>0)
            {
                int r = (int)(Math.random()*10);


            c1=new Color(255,255,255);
            DDALine(g,0,1,x1,((a-x1*x1)/10)+r,c1);
            DDALine(g,0,1,x1,((a-x1*x1)/10)+r+7,c1);
            DDALine(g,0,1,x1,((a-x1*x1)/10)+r+10,c1);
            DDALine(g,0,1,x1,((a-x1*x1)/10)+r+10,c1);
            DDALine(g,0,1,x1,((a-x1*x1)/10)+r+20,c1);
```

```
            DDALine(g,0,1,x1,((a-x1*x1)/10)+r+30,c1);
            }
            x1++;
        }

    }
    public void lightCandle(){
        flame = 1;
        repaint();
    }
    public void putOutCandle(){
        flame = 0;
        repaint();
    }
    public void paint(Graphics g){
            plotRect(g,-10,0,Color.red);
            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;
        if(flame ==1){
            try
            {

                Thread.sleep(100);
                repaint();
                //millisecond
                //...YOUR LOGIC
                  infinite(g);
            }


            catch (InterruptedException ie)
            {
            ie.printStackTrace();

            }
        }

    }
}
```

Candle.html-

```
html>
    <head></head>
    <body>
        <applet code ="Candle"  height="500" width="500"></applet>
    </body>
</html>
```