Experiment No : 5

Aim : Using Bresenham's Line Drawing algorithm, draw a cupboard

Theory :
Bresenham's Line Drawing algorithm is an accurate and efficient raster line-generating algorithm, developed by Jack Elton Bresenham, that uses only incremental integer calculations. In addition, Bresenham's line algorithm can be adapted to display circles and other curves.

Bresenham's Line Drawing Algorithm for m <1:

1. Input the two line endpoints and store the left endpoint in (x0, y0).
2. Set the color for frame-buffer position (x0, y0); i.e., plot the first point.
3. Calculate the constants Δ x, Δ y, 2Δ y, and 2Δ y - 2Δ x, and obtain the starting value for the decision parameter as p0 = 2Δy − Δx
4. At each xk along the line, starting at k = 0, perform the following test: If pk ¡ 0, the next point to plot is (xk + 1, yk) and pk+1 = pk + 2Δy
Otherwise, the next point to plot is (xk + 1, yk + 1) and pk+1 = pk + 2Δy − 2Δx
5. Repeat step 4, Δx − 1 more times.

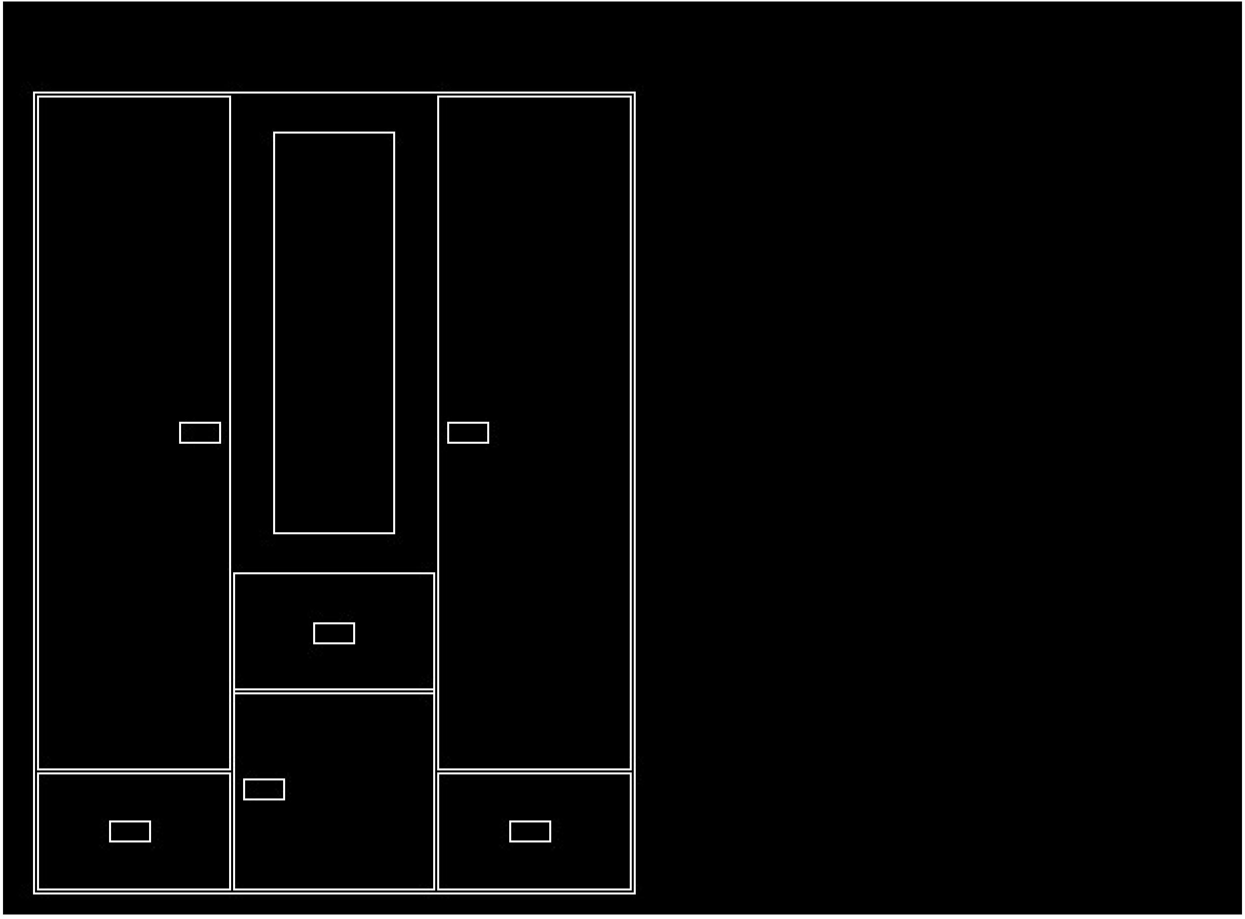
Code & Output :

```
#include<graphics.h>
#include<iostream>
#include<cmath>
using namespace std;

void bresenham(int a1, int b1, int a2, int b2);
void forPositiveSlope(int x1, int y1, int x2, int y2, float m);
void forNegativeSlope(int x1, int y1, int x2, int y2, float m);

int main()
{
  int gd=DETECT,gm;
  initgraph(&gd,&gm,NULL);
  bresenham(20,50,320,50);
  bresenham(20,50,20,450);
  bresenham(320,50,320,450);
  bresenham(20,450,320,450);

  bresenham(22,448,118,448);
  bresenham(22,390,22,448);
  bresenham(22,390,118,390);
  bresenham(118,390,118,448);
  bresenham(58,414,78,414);
  bresenham(58,414,58,424);
  bresenham(58,424,78,424);
  bresenham(78,414,78,424);

  bresenham(222,448,318,448);
  bresenham(222,390,222,448);
  bresenham(222,390,318,390);
  bresenham(318,390,318,448);
  bresenham(258,414,278,414);
  bresenham(258,414,258,424);
  bresenham(258,424,278,424);
  bresenham(278,414,278,424);
```

```
   bresenham(120,448,220,448);
   bresenham(120,348,120,448);
   bresenham(120,348,220,348);
   bresenham(220,348,220,448);
   bresenham(125,393,145,393);
   bresenham(125,393,125,403);
   bresenham(125,403,145,403);
   bresenham(145,393,145,403);

   bresenham(120,290,120,350);
   bresenham(120,350,220,350);
   bresenham(120,290,220,290);
   bresenham(220,290,220,350);
   bresenham(160,315,180,315);
   bresenham(160,315,160,325);
   bresenham(160,325,180,325);
   bresenham(180,315,180,325);

   bresenham(22,388,118,388);
   bresenham(22,52,22,388);
   bresenham(22,52,118,52);
   bresenham(118,52,118,388);
   bresenham(93,215,113,215);
   bresenham(93,215,93,225);
   bresenham(93,225,113,225);
   bresenham(113,215,113,225);

   bresenham(222,388,318,388);
   bresenham(222,52,222,388);
   bresenham(222,52,318,52);
   bresenham(318,52,318,388);
   bresenham(227,215,247,215);
   bresenham(227,215,227,225);
   bresenham(227,225,247,225);
   bresenham(247,215,247,225);

   bresenham(140,70,200,70);
   bresenham(140,70,140,270);
   bresenham(140,270,200,270);
   bresenham(200,70,200,270);


   delay(100000);
   closegraph();
    return 0;
}

void bresenham(int a1, int b1, int a2, int b2)
{
    float m;
    if(a1 < a2)
    {
        m = float(b2 - b1)/float(a2 - a1);
        if(m >= 0)
            forPositiveSlope(a1,b1,a2,b2,m);
        else if(m < 0)
            forNegativeSlope(a1,b1,a2,b2,m);

    }
    else if(a2 < a1)
    {
        m = float(b1 - b2)/float(a1 - a2);
         if(m >= 0)
            forPositiveSlope(a2,b2,a1,b1,m);
        else if(m < 0)
            forNegativeSlope(a2,b2,a1,b1,m);

    }
    else if(a1 == a2)
    {
        m = float(b2 - b1)/float(a2 - a1);
        if(b1 < b2)
            forPositiveSlope(a1,b1,a2,b2,m);
        else if(b2 > b1)
            forNegativeSlope(a1,b1,a2,b2,m);
    }

}

void forPositiveSlope(int x1, int y1, int x2, int y2, float m)
{       int p0, pk, deltaX, deltaY;
        deltaX = x2 - x1;
        deltaY = y2 - y1;
        if(m < 1)
        {
            p0 = 2*deltaY - deltaX;
            pk = p0;
            while((x1 <= x2) && (y1 <= y2))
            {
                putpixel(x1,y1,WHITE);
                if(pk < 0)
                {
                    x1 = x1 + 1;
                    y1 = y1;
                    pk = pk + 2*deltaY;
                }
                else
                {
                    x1 = x1 + 1;
                    y1 = y1 + 1;
                    pk = pk + (2*deltaY) - (2*deltaX);
                }

            }
        }
        else if(m >= 1)
        {
            p0 = 2*deltaX - deltaY;
            pk = p0;
            while((x1 <= x2) && (y1 <= y2))
            {
                putpixel(x1,y1,WHITE);
                if(pk < 0)
                {
                    x1 = x1;
                    y1 = y1 + 1;
                    pk = pk + 2*deltaX;
                }
                else
                {
```

```
          x1 = x1 + 1;
          y1 = y1 + 1;
          pk = pk + (2*deltaX) - (2*deltaY);
        }
      }
    }
}

void forNegativeSlope(int x1, int y1, int x2, int y2, float m)
{   int p0, pk, deltaX, deltaY;
    deltaX = x2 - x1;
    deltaY = y2 - y1;
    if(abs(m) < 1)
    {
      p0 = (-2*deltaY) - deltaX;
      pk = p0;
      while((x1 <= x2) && (y1 >= y2))
      {
        putpixel(x1,y1,WHITE);
        if(pk < 0)
        {
          x1 = x1 + 1;
          y1 = y1;
          pk = pk - 2*deltaY;
        }
        else
        {
          x1 = x1 + 1;
          y1 = y1 - 1;
          pk = pk - (2*deltaY) - (2*deltaX);
        }
      }
    }
    else if(abs(m) >= 1)
    {
      p0 = (-2*deltaX) - deltaY;
      pk = p0;
      while((x1 <= x2) && (y1 >= y2))
      {
        putpixel(x1,y1,WHITE);
        if(pk > 0)
        {
          x1 = x1;
          y1 = y1 - 1;
          pk = pk - 2*deltaX;
        }
        else
        {
          x1 = x1 + 1;
          y1 = y1 - 1;
          pk = pk - (2*deltaY) - (2*deltaX);
        }
      }
    }
}
```

Output :



Conclusion : Program to draw a house using DDA was successfully written and executed

Deepraj Bhosale
181105016