Line Drawing Algorithms

Upasana Ghosh
UE143110

U.I.E.T, Panjab University
Email: ghoshupasana05@gmail.com

AIM – Study and implement line drawing algorithm.

INTRODUCTION

A line drawing algorithm is a graphical algorithm used for drawing a line segment on a discrete graphical media. A graphical media consist of scan lines and scan columns. A good line drawing algorithm should appear straight, should end accurately, and should be independent of line length estimate and angle of inclination. It should have constant density and should be drawn rapidly. Some of the line drawing algorithms are Digital Differential Analyzer, Bresenham's line algorithm, Xiaolin Wu's line algorithm, mid point line drawing algorithm, Gupta-Sproull algorithm, etc.

This paper provides an analysis of Digital Differential Analyzer, Bresenham's line algorithm and mid point line drawing algorithm

APPROACH

Digital Differential Analyzer (DDA) algorithm is of two types: Simple Digital Differential Analyzer and Symmetric Digital Differential Analyzer

DDA Algorithm: DDA Algorithm is described in the following steps:
Step 1 −Take the input of two end points (x_0,y_0) and (x_1,y_1).
Step 2 − Calculate the difference between two end points as: $dx = x_1 - x_0$
$dy = y_1 - y_0$
Step 3 − Based on the calculated difference in Step 2, the number of steps to putpixels is determined. If dx >dy, then more steps are needed in x coordinate; otherwise in y coordinate.

```
if (dx >dy)
        Steps = absolute(dx);
            else
Steps = absolute(dy);
```

Step 4 − The increment in x coordinate and y coordinate is calculated as:
$X_{increment} = dx / steps;$
$Y_{increment} = dy / steps;$

Step 5 − The pixels are printed consecutively by incrementing x and y coordinates accordingly.

```
for(i=0; i< steps; i++)
{   x = x + X_increment;
    y = y + Y_increment;
    putpixel(x,y);
}
```

Bresenham's Algorithm: Bresenham's Algorithm generates line by plotting one pixel at a time. An advantage of using this algorithm is that it uses only integer calculations since float calculations are more complex.
For a line having slope (m) < 1, Bresenham's Algorithm chooses between two different but consecutive y coordinates (i.e., y and y+1)for each increment in x coordinate. This choice is based on whether the passing line is closer to y or y+1 coordinate. The coordinate which is closer to the line is illuminated. In this way the entire line is generated.
For lines with slope (m) > 1, it is decided whether to increment x or not for each increment in y.

Mid Point Line Drawing Algorithm: This algorithm also generates line by plotting one pixel at a time. An advantage of using this algorithm is that it uses only integer calculations since float calculations are more complex and time consuming.
In this, a hypothetical midpoint between scan line and column line is used to check whether the point should be plotted in next scan line or column line or in same scan line or column line.

Machine Specifications:

Operating System: Microsoft Windows 10
Architecture: 64 bit (x64)
Coding Platform: Turbo C++ 4.0
Processor: Intel Core i3-4005U CPU@ 1.70GHz
Physical Memory (RAM): 4.00 GB

IMPLEMENTATION

A C++ program has been implemented to draw the lines. It chooses a particular algorithm and line style and then takes the end points of line to be drawn as input on mouse click and then draws a line between the two points. The program uses a graphics.h header file which contains the function to implement the graphical interface.

graphics.h header functions can be used to draw different shapes, display text in different fonts, colours and many more. This header file can be used for animations, projects and games. Different sizes of circle, rectangle, ellipse, lines, bars and other geometrical figures can be drawn with this header file. The header file dos.h of C++ language contains functions for handling interrupts, producing sound, date and time functions etc. It is borland specific and works in turbo c compiler.

Buttons are created using bar3d() function which takes horizontal and vertical axis as input and display a rectangular bar object in graphical screen. Setfillstyle() is used to fill the graphical object with a specific color and style. Delay() is used to introduce delay in the program.

As the lines are drawn using mouse and keyboard click, graphic.h and dos.h header files are added in this program. Mouse.h header file consist of several functions like initMouse(), showMouse(), hideMouse() and Isclick(). All these functions uses AX register to access mouse pointer and keyboard keys. Interrupts are called to use graphics and provide delay in taking inputs. Mouse events are tracked by using software interrupt ‘0x33’. Here ‘0x’ means that the number is in hexadecimal. The interrupt call is made using int86( ) function. This function is defined in dos.h. They are of type union REGS. The union inreg and outreg hold values of registers before and after the call has been made.
To initialize a mouse, the value of AX register is set as ‘0x00’. This tells the system that when the ‘0x33’ interrupt is called, it should initialize the mouse (basically get it ready for other functions). showMouse() function set the AX register value to 1 and display the mouse pointer in graphical screen. hideMouse() function set the AX register value to 2 and is used to hide the mouse pointer from the graphical screen. Isclick() function set the AX register value to 3 and is used to get the x and y coordinate of the screen where user has clicked with mouse pointer. It also tells the mouse button (left, right or middle) the user has pressed on the graphical screen. kbhit( ) function returns nonzero integer if any key is pressed else it returns zero.

Graph: The area where UI items are displayed is the graph which is initialized by initgraph() function. getmaxx() and getmaxy() functions are used to find maximum value of coordinates of graphical screen. The origin point (0, 0) by default is located at the top-left corner of the graph. The closegraph() function closes the currently opened graph window.

Text: The text is displayed in the graph using outtextxy() function. This function takes 3 arguments:
The x-coordinate
The y-coordinate
Text to be displayed

Buttons: Buttons are created using bar3d() function. Creating a rectangular bar of known dimensions using bar3d() function takes four arguments:
X-coordinate of left edge
Y-coordinate of top edge
X-coordinate of right edge
Y-coordinate of bottom edge

Style : Along with drawing lines on graphical screen, user can also use different styles to draw lines. User can select the width, colour and pattern for the lines.

RESULT

Figure 1 to Figure 7 represents the formation of line using Simple DDA algorithm an

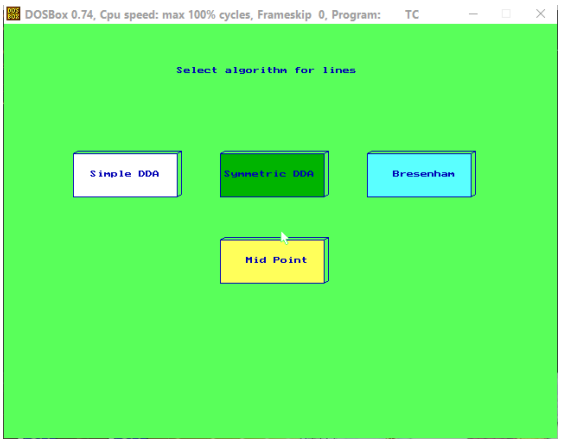

Fig 1: Algorithm selection menu

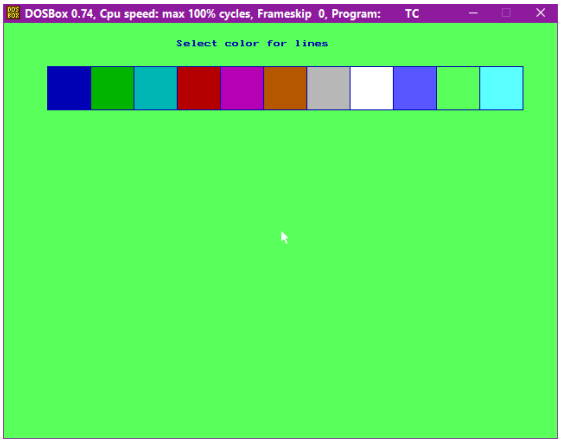

Fig 2: Color selection menu

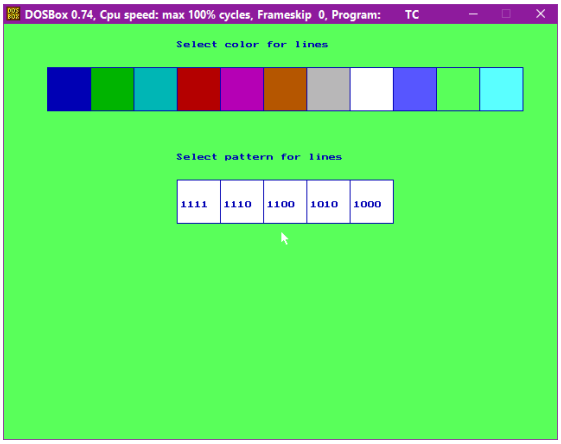

Fig 3: Pattern selection menu

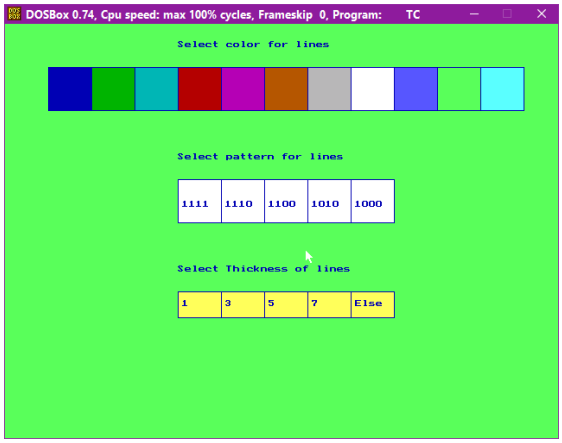

Fig 4: Thickness selection menu

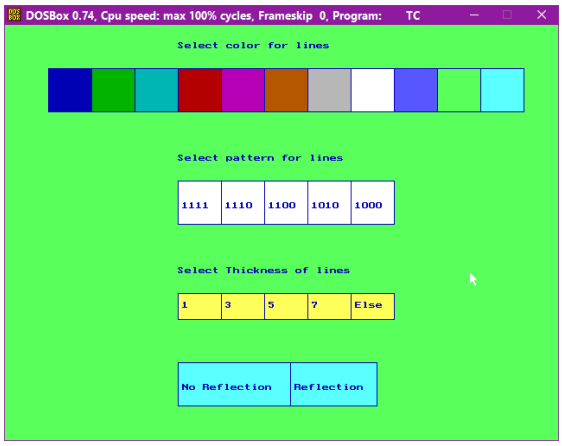

Fig 5: Reflection selection menu

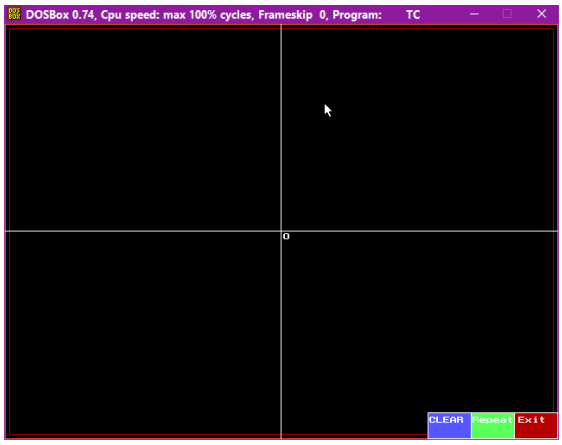

Fig 6: Graphical screen to draw line

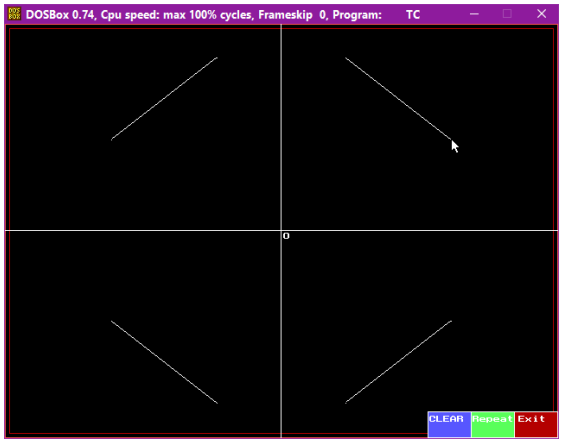

Fig 7: Line is drawn using Simple DDA algorithm

DISCUSSION

In DDA line drawing algorithm, any line that can be represented by ordinary differential equation can be drawn using this algorithm. In symmetrical DDA there are chances of getting duplicate points which is highly undesirable. In simple DDA, duplicate points are not generated. Hence, simple DDA algorithm is better than symmetric DDA algorithm. But the disadvantage of DDA algorithm is the calculation of fractional values which is time consuming.

In Bresenham's algorithm, duplicate points are not generated and it does not include fractional calculations as well. Similarly, midpoint algorithm does not generate duplicate points and it includes only integer calculations. Although, midpoint line drawing algorithm uses an hypothetical concept to draw a line but it also generate the same points as generated by Bresenham's algorithm. The line generated by both these algorithms are straight, accurate and of constant density.

CONCLUSION

Bresenham's line drawing algorithm and midpoint algorithm draws a good line on graphical screen.