Graphics Assignment 01

21118

DATE

Title: scanline algorithm for polygon filling

Problem Statement:
Write C++ program to draw a concave polygon & fill it with desired colour using scan-fill algorithm. Apply the concept of inheritance.

Objectives
- to fill the polygon by using scan-line algorithm
- to use the mouse click even

Outcomes
- students will be able to implement scan-line algorithm & will know how to fill the polygon
- students will be able to use mouse click event & will able to draw a polygon with no. of slides

Software & Hardware Requirement:
- Qt creator (latest version)
- Windows 10, 64bit, GCC/g++ compiler

Theory:
- It is an image space algorithm. It processes one line rather than one pixel at a time
- It uses the concept of coherence. This algorithm records edge list, active edge list.
The edge list (or) edge table contains the co-ordinates of two endpoints

Advantages:
- It only draws visible pixels
- It requires less memory & takes advantage of each edge coherence. Such results in faster execution.

Disadvantages

- It is more complex
- it requires all polygon should be sent to render before drawing.

Algorithm

1. Set y_{scan} to smallest y for which there is an empty entry in ET
2. AET is initially empty
3. repeat both stops of AET & ET are empty
   - i) move edges from ET to AET whose $y_{min} = y_{scan}$
   - ii) fill the visible span.
   - iii) Remove edges with $y_{max} = y_{scan}$
   - iv) Increment y by 1.
   - v) for each edges in AET, update x-co-ordinate
   - vi) sort all edges in x.

Pseudo Code

~~Algorithm Bresenham(x1, y1, x2, y2) {~~
~~1. set dx~~

Algorithm Scanline ( ) {

1. a[var] = a[0]
2. b[var] = b[0]
3. for i←0 to i←var do
   - 3.1 set dy = b[i+1] - b[i]
   - 3.2 set dx = a[i+1] - a[i]
   - 3.3 if (dy==0) m[i] = 1.0
   - 3.4 if (dx==0) m[i] = 0.0
   - 3.5 if ((dy != 0.0f) && (dx != 0.0f))
     - 3.5.1 m[i] = dx/dy
4. for (int y = 0; y < var; y++) do
   - 4.1. k←0

4.2 for i←0 to i←var1 do

4.2.1 if ((b[i]<=y && b[i+1]>y) || (b[i]>y && b[i+1]<=y))

4.2.1.1 xmin[k] = inf(a[i] + m[i] * (y-b[i]))

4.2.1.2 k←k+1

4.3 for j←0 to j←k-2 do

4.3.1 for i←0 to i←k-j-2 do

4.3.1.1 if (xmin[i] > xmin[i+1])

4.3.1.1.1 temp ← xmin[i]

4.3.1.1.2 xmin[i] = xmin[i+1]

4.3.1.1.3 xmin[i+1] = temp

4.4 for i←0 to j←k+1

4.4.1 drawline (xmin[i], y, xmin[i+1], y)

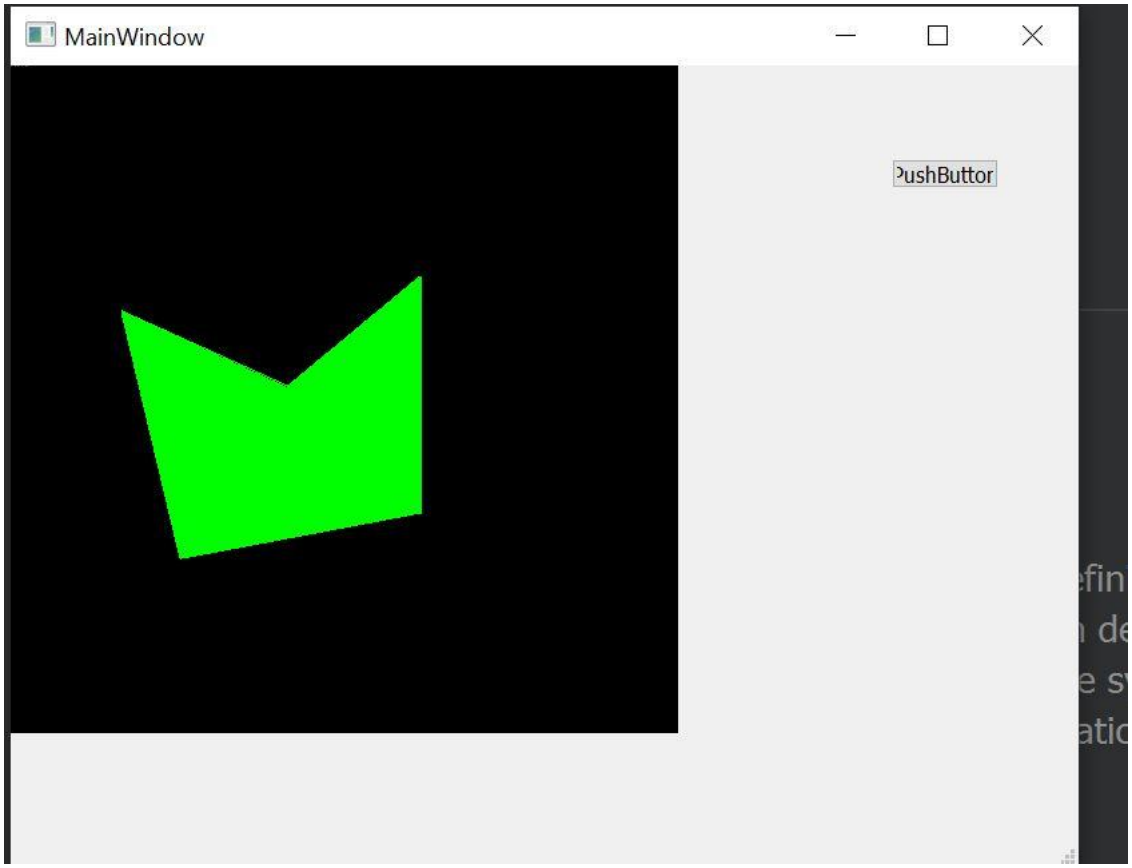
Testcases

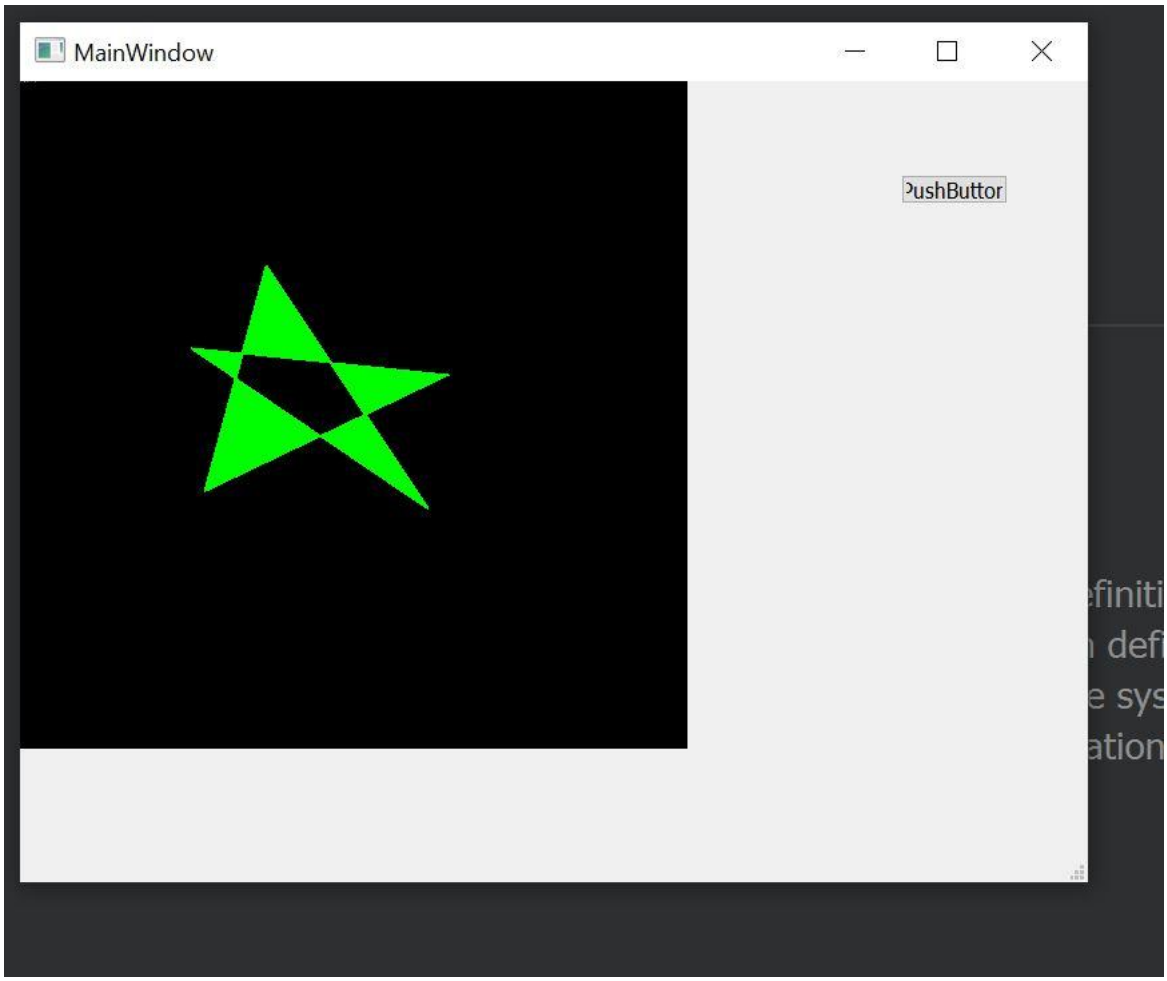
SNo	input	expected output	Actual o/p	Result
1.				Passed
2.				Passed

Time complexity: $O(n^2)$

Conclusion

We learned how to use scan-line algorithm to fill the polygon & also learned mouse click events to draw polygon.


MainWindow
PushButtor


MainWindow
PushButtor