Volume 3, Issue 11, November 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Efficient Fence Fill Algorithm using Inside-Outside Test

S.Anitha, D.Evangeline
Department of Information Technology,
School of Communication and Computer Sciences,
Kongu Engineering College, Perundurai, Erode-638052,
Tamil Nadu, India

Abstract*— In this paper, we introduce a modified fence fill algorithm for filling a polygon. Fence fill algorithm is a modified edge fill algorithm. As opposed to the edge fill algorithm that fills pixels starting from the polygon edge to the maximum screen coordinates, the fence fill algorithm fills pixels bounded by the polygon edge and the fence. This involves large number of calculations to find the intersection of the polygon edge with the scan line. Our method reduces the calculations by considering a single pixel in each region bounded by the fence and polygon edge. The inside-outside test is then applied to that pixel to determine whether the point lies inside or outside the bounded region. For large concave polygons, the older method is complicated as it requires more space and time. The new method promises a significant reduction in memory and time requirements.

Keywords— ***Polygon fill, Edge fill, Fence fill, Inside-outside test***

I. INTRODUCTION

Computer Graphics are graphical image created using computers with specialized graphics hardware and software. Images are composed of basic structures like points, lines, curves, polygons, etc., Knowledge of Graphics is essential in the field of image processing which finds its applications in research, medicine, steganography, satellite communication, remote sensing, biometrics and so on. A polygon is a closed contour. Polygons are classified as convex and concave (non-convex). Convex polygons are those polygons for which the line joining any two points within the polygon also lies completely within the polygon. In computer graphics, vector method and rotational method are two methods used to determine whether the polygon is concave or convex. If a polygon is concave, these methods split such polygons into a pair of convex polygons. Polygon filling is the process of coloring the pixels of a polygon. Many polygon filling algorithms are employed to fill polygons in various graphics packages. Some of the traditional polygon fill algorithms include Scan Line Polygon Fill, Seed Fill and Edge Fill. Such algorithms can be applied to both concave and convex polygons but is generally more complicated for concave polygons. The scan line algorithm computes the intersection of the polygon edge with each scan line. For each scan line crossing the polygon edge, the intersection coordinates are sorted left to right and the pixels in each scan line are colored in a very similar manner. Boundary Fill and Flood Fill algorithms fall under Seed fill since these algorithms commence with the assumption of a known interior pixel within the polygon. Seed Fill algorithms that fill interior defined regions are known as Flood Fill and those seed fill algorithms that fill boundary defined regions are known as Boundary Fill. Both the seed fill algorithms may be 4-connected or 8-connected. Edge Fill algorithm fills pixels from the intersection coordinate of polygon edge with the scan line to the extreme screen coordinates for each scan line. Hence, the pixels lying exterior to the polygon will be colored twice resulting in the initial unmarked condition. Fence Fill is a modified edge fill algorithm wherein a fence is constructed such that it passes through any vertex of the polygon. The pixels lying in the region bounded by the fence and the polygon edge is colored. All these algorithms work in raster coordinate systems which may be region-oriented or vector-oriented.

Considering polygon filling as a problem of coloring interior pixels of a polygon, we propose a modified and efficient fence fill algorithm.

(1) Construct a fence that passes through the vertex or center of a polygon edge;
(2) Apply inside-outside test to a single pixel lying in each region created by the polygon edge with respect to the fence;
(3) If the pixel of the region lies within a polygon, color all the pixels lying in the region.

The remainder of the paper is organized as follows. In section I, an overview of the conventional polygon splitting, inside-outside tests and filling algorithms required is discussed. In section II, the proposed algorithm is illustrated. We draw our conclusion and results in the last section.

II. BACKGROUND

A. Test to determine whether a polygon is concave or convex[2]

1. Vector Method

For a convex polygon, the cross product of adjacent edges (ie., the z-component) holds the same sign. But, for a concave polygon, the z-component turns out to be negative and the polygon is split along the line of the first edge vector in the cross product pair.

2. Rotational Method

Process the polygon vertices in anti-clockwise direction. Translate the vertices of a polygon V_i to the coordinate origin. Rotate the polygon in clockwise direction such that the next vertex V_{i+1} lies on the x-axis. If the vertex V_{i+2} lies below, the polygon is concave and it is split along x-axis. If not, continue to rotate polygon vertices on x-axis and test for negative y values.

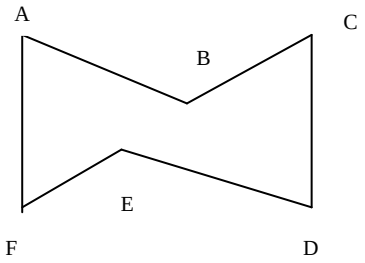

Figure-1. Concave Polygon

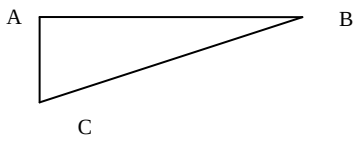

Figure-2. Convex Polygon

Using the vector and rotational method, the polygon ABC is determined as convex and the polygon ABCDEF is determined as concave and the polygon is split along the edge AB.

B. Inside Outside Tests

1. Odd Even Rule

If the number of polygon edges crossed by the line starting at a position P to a distant point outside the object, then P is an exterior point. Otherwise, P is an interior point.

2. Winding Number Rule

If the number of times the edges wind around any point P in anti-clockwise direction is non-zero, then P is an interior point. Otherwise, P is an exterior point.

When the region bounded by EA, AB, BC, DE and CD is considered exterior, Odd Even rule and Winding Number rule reveal that:

i. PP_1 and PP_4 are exterior
ii. PP_2 and PP_3 are interior

It is not necessary that both the rules may yield the same result.

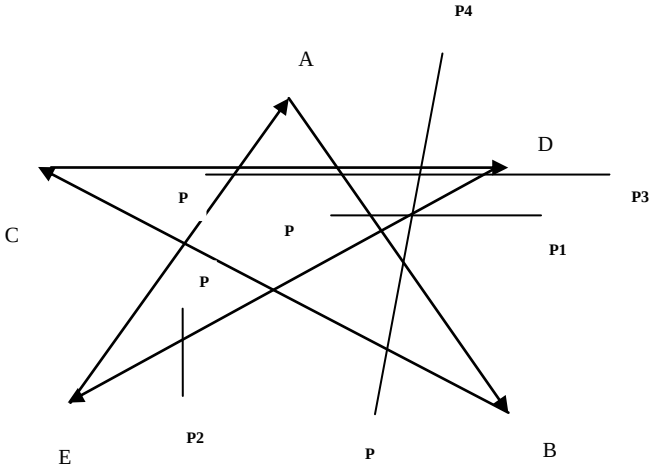

Figure-3 Inside –Outside Tests

C. Polygon Filling Algorithms

The simplest polygon filling algorithm considers every pixel in the raster, verifies whether it lies within the polygon and colors the pixel if it is interior. This technique is tedious since each and every pixel in the raster is considered. Alternately, a bounding rectangle for the polygon ie., the smallest rectangle containing the polygon is constructed and the pixels within the bounding rectangle is considered.

i. Scan Line Polygon Fill Algorithm

A polygon is scan converted by finding the intersection of all polygon edge on each scan line. The pixels within the polygon are then determined and colored.

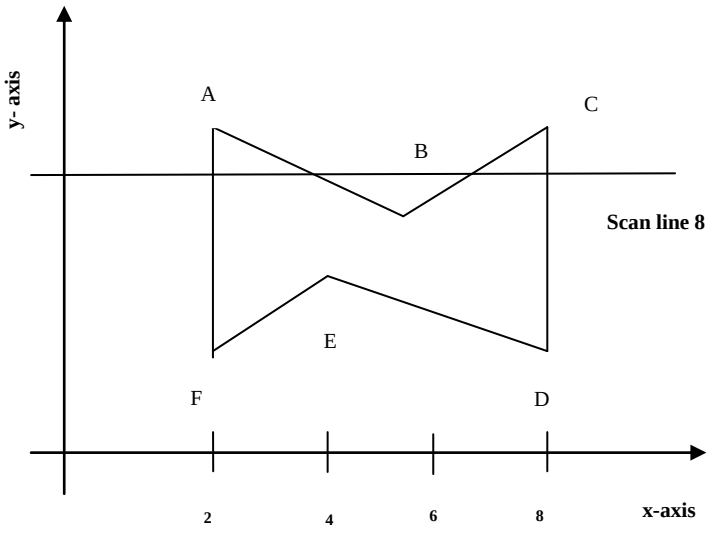

Figure-4. Scan Line Polygon Fill

The intersection of polygon edge AF, AB, BC and CD on scan line 8 is found. The pixels on scan line 8 spanning from x=2 to x=4 and x=7 to x=8 are colored.

ii. Seed Fill Algorithm

The seed fill algorithms (boundary fill and flood fill) are highly recursive. These algorithms may be 8-connected or 4-connected.

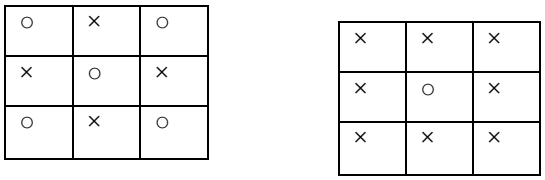

Figure-5. Seed Fill

Seed Fill algorithms accept as input: a known interior pixel (seed pixel) and fill color. Boundary Fill algorithms start with the seed pixel, check whether their neighbors (4-connected or 8-connected) are of boundary color and if not, they are filled with fill color. This is recursively done until pixel with boundary color is encountered. When the interior region is marked with multi-colored or no boundary, flood fill algorithm is adopted. Here, the interior pixels are filled with a specific color. Check neighbors (4-connected or 8-connected) if they are of interior color. If so, fill such pixels with fill color.

iii. Edge Fill algorithm

The algorithm finds the intersection of the polygon edge with each scan line, colors pixels from the intersection to the extreme screen coordinates for each scan line. This is done for all polygon edges. Interior pixels within the polygon are colored once and exterior pixels are colored twice.

This algorithm can be restated as:"For each scan line intersecting the polygon edge at (x_1,y_1), complement all the pixels whose midpoints lie to the right of (x_1,y_1) i.e., for (x,y_1), $x+1/2>x_1$."[3]

iv. Fence Fill algorithm

This algorithm also finds the intersection of the polygon edge with each scan line. However, coloring is done based on the position of the pixels with respect to the fence (a line that passes through any vertex or polygon edge). Each pixel in the region created by the polygon edge and fence is checked whether it is interior or exterior.

III. PROPOSED ALGORITHM

The algorithm focuses on filling concave polygons.A fence location is usually considered passing through any polygon vertex for simplicity. A bounding rectangle for the polygon is constructed. The regions bounded by the polygon edge with the fence are identified. A single pixel in each region is chosen and inside-outside tests are carried on it. If the test returns interior, the pixel is colored. Otherwise, the pixel remains uncolored.

The algorithm is illustrated in the following figures.

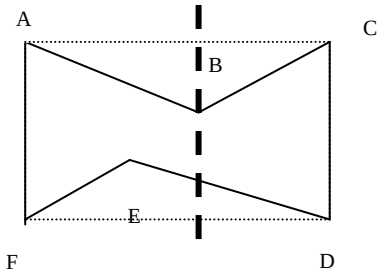

Figure-6 Fence and bounding rectangle

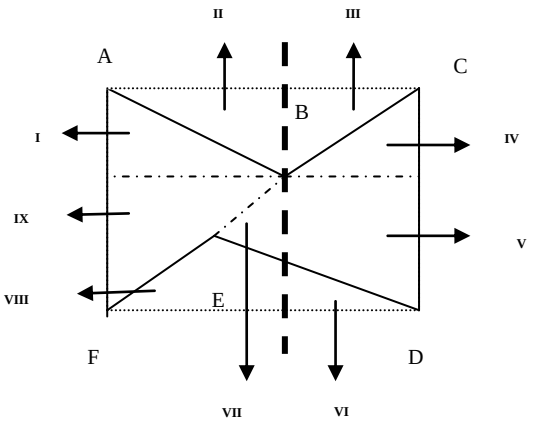

Figure-7 Identification of regions

Consider an imaginary line passing through the intersected point between the edge and the fence along the scan line. For those edges not intersecting at the fence, extend the corresponding edge towards the fence. The different regions are considered based on closed area formed between imaginary line, edge and fence. Inside-outside test is conducted on a single pixel on each region. Since the regions I, IV, V, VII and IX are interior, the pixels in these regions are filled.
The proposed method shown in the Figure-7 which reduces the number of calculations compared to existing algorithms.
Consider a large screen display A(0,1000), B(1000,1000), C(1000,2000) and D(0,2000).

Convex Polygon

Case 1: Three sided A(100,100) B(100,200) C(150,150).
Case 2: Four sided A(50,150) B(100,100) C(100,200) D(150,150).
Case 3: Five sided A(50,150) B(100,220) C(150,180) D(120,100) and E(100,50).

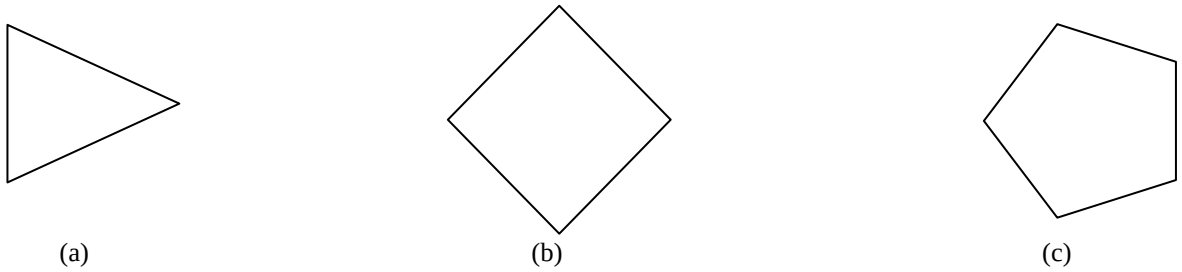

Fig 8 Convex Polygon (a) Case 1 (b) Case 2 (c) Case 3

Concave Polygon

Case 1: Four sided A(125,150) B(100,120) C(125,200) D(150,120).
Case 2: Five sided A(100,120) B(85,205) C(120,180) D(165,215) and E(150,150).
Case 3: Six sided A(50,100) B(50,150) C(90,140) D(125,155) , E(100,120) and F(75,125).

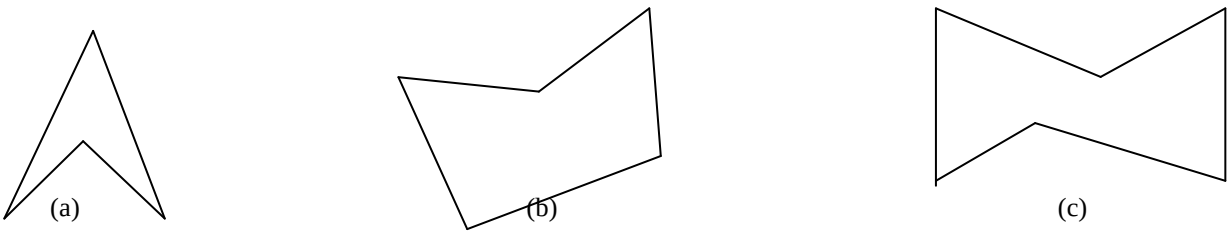

Fig 9 Concave Polygon (a) Case 1 (b) Case 2 (c) Case 3

IV. PERFORMANCE EVALUATION

In Table 1, a comparison on Edge Fill, Fence Fill and the proposed algorithm is made based on the number of calculations done to find the intermediate pixels on the polygon's edge intersecting the scan line and the number of comparisons required to determine the position of the pixel (interior or exterior to the polygon). For comparison, three cases in both concave and convex polygons are considered. The edge fill and fence fill algorithms require calculations for determination of intermediate pixels. In our proposed algorithm does not require any such calculations. In all the three

algorithms, the position of the pixels must be identified. The edge fill algorithm requires large number of computations. In cases where a small polygon is displayed on a large screen area, the number of calculations is very high. The fence fill algorithm also requires such calculations but the number of calculations is considerably very much lesser than edge fill algorithm. This is due to the construction of the fence at the most appropriate location. Our proposed algorithm also requires such calculations but it selects only one pixel per region and determines the position of the pixel. Thereby, the number of comparisons made is negligible.

Table 1 Performance Evaluation of various Polygon Filling Algorithms

Type of polygon	Cases	Algorithms						
		Edge Fill		Fence Fill			Proposed	
		No of intersection calculations	No of comparisons	No of intersection calculations	No of comparisons	Fence Location (Vertical Line)	No of intersection calculations	No of comparisons
Convex	1	200	260100	200	7550	150	0	3
	2	200	262650	200	5100	100	0	8
	3	340	171345	340	4115	100	0	8
Concave	1	250	69679	250	5000	125	0	8
	2	240	22377	240	6815	120	0	4
	3	140	133800	140	1750	140	0	9

V. CONCLUSION

Our algorithm considerably reduces number of comparisons for polygon filling using Edge Fill and Fence Fill algorithms. In traditional Edge Fill algorithm, not only the pixels interior to the polygon but also the pixels exterior to the polygon are considered for comparison to fill the polygon. This increases both the memory requirements and execution time. In cases where a small polygon is displayed on large screen, the number of pixels considered for comparison greatly increases. But in Fence Fill, appropriate selection of fence drastically reduces such comparisons. Yet, the number of comparisons is more. Our algorithm focuses only the polygonal area for comparison to fill it. Irrespective of the size of the screen, the number of comparisons made is proportional to the size of the polygon (ie., the number of pixels interior to the polygon). Moreover, not all the pixels interior to the polygon are considered for comparison. Only a pixel per region is considered for comparison in our proposed algorithm. However, the accuracy of the algorithm totally depends on the results of the Inside-Outside Tests.

REFERENCES

[1] Michael R Dunlavey. *Efficient Polygon Filling Algorithms for Raster Displays*. ACM Transactions on Graphics, Vol.2, No 4, October 1983, Pages 264-273.
[2] Donald D.Hearn, M.Pauline Baker, *Computer Graphics C Version*, Second Edition, Pearson Education.
[3] David F. Rogers, *Procedural Elements of Computer Graphics*, First Edition, McGraw Hill
[4] Foley van Dam, Feiner Hughes, *Computer Graphics Principles & Practice*, Second Edition in C, Pearson Education.
[5] Steven Harrington,*Computer Graphics A Programming Approach*, Second Edition, McGraw Hill
[6] A New Polygon Based Algorithm for Filling Regions, Hoo-Cheng Liu, Mu-Hwa Chen, Shou-Yiing Hsu, Chaoyin Chien, Tsu-Feng Kuo and Yih-Farn Wang, *Tamkang Journal of Science and Engineering, Vol. 2, No. 4, pp. 175-186 (2000).*

