

Chapter - 3: Polygons

Basic computer graphics applications are using the elements like lines, points, curves, circles with surfaces, colors, shapes shades, patterns etc. In this chapter we term the pixel at vertex and the line segment as an edge.

3.1 Polygons

Polygon is a word derived from two Greek words **poly** and **Gon**, where **poly** means Multi and **Gon** means Angle. "Polygon, is a closed figure with many vertices and edges (line segments), and at each vertex exactly two edges meet and no edge crosses each other".

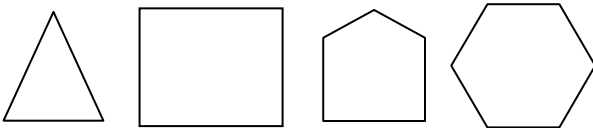
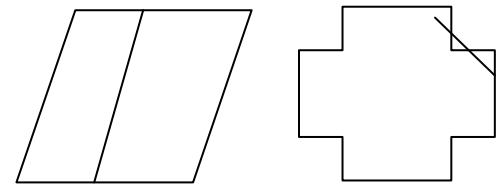


Fig. 3.1 Regular Polygons

In the figure shown besides in fig. 3.1 are called regular polygons as the total no of edges is equal to the total no of angles.

Basically polygons are divided in two classes (i) Convex Polygons and (ii) concave polygons. A **convex polygon** is a polygon in which if you take any two points of polygon then all the points on the line segment joining these two points fall within the polygon itself. A concave polygon is a polygon in which a line segment joining any two points in the polygon does not fall completely in the polygon.



Convex Polygon

Concave Polygon

Fig. 3.2 Classes of Polygons

For displaying such type of figure of polygons in computer graphics many display devices treat the entire polygon as single unit. Many languages take an array of the vertices for constructing the polygons on display device.

3.2 Polygon Inside Tests

A polygon is a solid object in computer graphics so it is required to turn on all the pixels which are interior points of the polygon. In such scenario it is required to test whether the point or pixel is inside the polygon or not. Basically there are two methods used to determine this; (1) Even-Odd Method and (2) Winding Number Method

3.2.1 Even-Odd Method

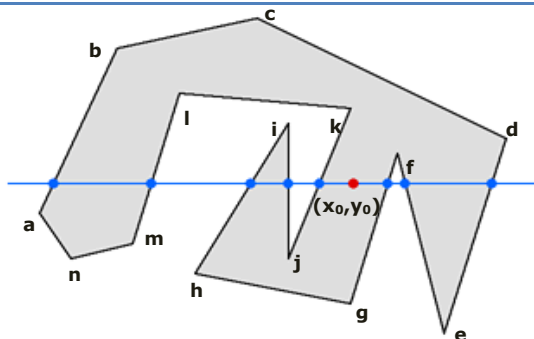


Fig. 3.3 Polygon Inside Test

This method is also known as Counting Number Method. If we want to test a point (x_0, y_0) is inside a polygon or not. Here the polygon is with 14 vertices $a, b, c, \dots, n$. The even-odd method uses the scan line, which passes through the point (x_0, y_0) . Let assume that the horizontal line $x_i = y_0$.

In the fig.3.3 the intersection points are being displayed by the blue dotted points. The total no of

intersection on the left of the point (x_0, y_0) is 5 whereas the total no. of intersection points on the right is 3. That is on the both sides the no. of intersection point is odd. The test says that if total no of intersections on both the side of the point to be test is odd then the point is an interior point of the polygon.

There are certain cases when this particular method works with some special implementations which are shown below.

Case: 1

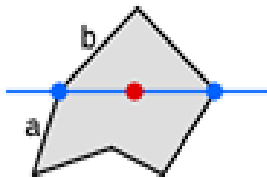


Fig. 3.4 – Test Case1

Figure 3.4 demonstrates the problem that results when a vertex of the polygon falls directly on the intersection points if the two edges. Since sides **a**, **b** both touch the intersection points, should they both generate a node? No, because then there would be total no of intersection will be 2 on each side of the test point and so the test would say it was outside of the polygon.

The solution to this situation is simple. Points which are intersecting here will be checked according to the edge residing. Here the edges are on opposite sides of each other. So the total no of intersection which has been displayed in blue color will be considered as odd which are 1 on both the side of the point in test (red) and the point is said to be interior.

Case: 2

Figure 3.5 shown besides is having a problem that the scan line passing from the point in test is passing through the intersection point of two edges of the polygon. Similar situation is occurring here that is when the scan line is passing from the intersection point of two edges then if we count it as odd then it become an even no. of intersection. And according to the method it will be considered as an exterior point which is not exactly.

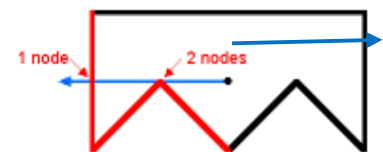
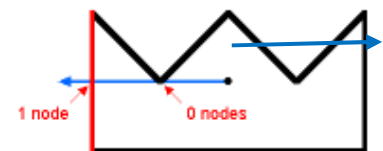


Fig. 3.5 – Test Case2

In this case the solution is, whenever the scan line is passing from the intersecting edge and both the edges are in the same direction of the scan line then we have to consider the total no. of intersection will be considered as even. So in the above figure the total no. of intersection on the left and right side of the point in test is 1 and 3 in above figure 3.5 respectively in two sub-images. Which means the point will be considered as an interior point.

Case: 3

Figure 3.6 shows the case of a polygon in which one of its sides lies entirely on the scan line passing from the point in test. Simply follow the rule as described concerning Figure 3.6. Side **c** intersects in on point, because it has one endpoint below the scan line and its other endpoint on-or-above the scan line. Side **d** does not intersect, because it has both endpoints on-or-above the scan line. And side **e** also does not intersect, because it has both endpoints on-or-above the scan line.

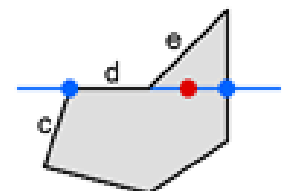


Fig. 3.6 – Test Case3

3.2.2 Winding Number Method

This method is also used with the simple polygons to test the given point is interior or not. It can be simply understand with the help of pin and rubber band. Fixed up the pin on one of the edge of the polygon and tie-up the rubber band in it and then stretch the rubber band along the edges of the polygon. When all the edges of the polygon are covered by the rubber band, check out the pin which has been fixed up at the point to be test. If we find at least one wind at the point considered within the polygon, else we can say that the point is not inside the polygon. To explain this situation with programmable manner is not possible.

For this we have one more alternative that is we have to give the directions to all the edges of the polygon. In this we have to draw a scan line from the point to be test towards the left most of X direction. We have to give the value 1 to all the edges which are going to upward direction and all other -1 as direction values. After giving direction value to all the edges now check the edge direction values from which the scan line is passing and sum up them. If the total sum of this direction value is non-zero then this point to be test is said to be an interior point otherwise it is said to be a n exterior point. In the fig. 3.7 we sum up the direction value from which the scan line is passing then the total is $1 - 1 + 1 = 1$; which is non-zero so the point is said to be an interior point.

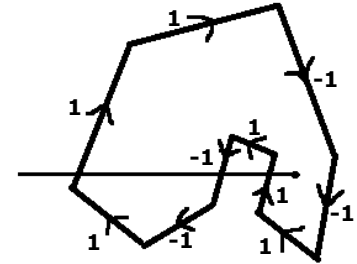


Fig. 3.7 – Winding Number Method

3.2.3 Some Other Testing Methods

If the polygon is convex, then we can test whether the point is inside or outside by checking the side of the point relative to the edges of the polygon. There are two facts to be considered here; end point of one edge is the start point of the consecutive edge and start point of the polygon will be the end point of the polygon since it is a closed figure.

Let assume that a point to test is point (x, y) and polygon points are (x_0, y_0) and (x_i, y_i) , where $i = 1(1)^n$, (n is the total no. of edges in the polygon). We have to find the side index S_i for the successive points (x_0, y_0) and (x_i, y_i) with the help of following equation;

$$S_i = (y - y_0) (x_1 - x_0) - (x - x_0) (y_1 - y_0)$$

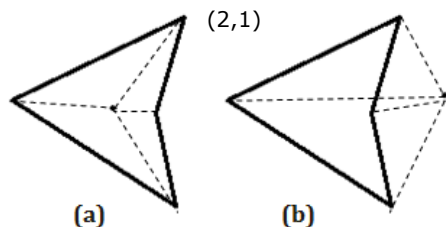


Fig. 3.9 – angles created with the end points of polygon edges

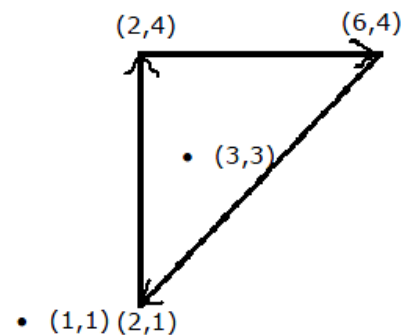


Fig. 3.8 – polygon inside test

If S_i is less than zero then the point is said to be on the right side of the edge. If S_i is greater than zero then the point is said to be on the left side of the edge. If S_i is equal to zero then the point is said to be on the edge of the polygon. If all the side index values are on the same side then the point is said to be the interior point else an exterior point. In the above fig. 3.8 the value of side index for point $(1,1)$ are 3, -12 and -3 where as for point $(3,3)$ is -3, -4 and

-5. So we can say that the point (3,3) is an interior point.

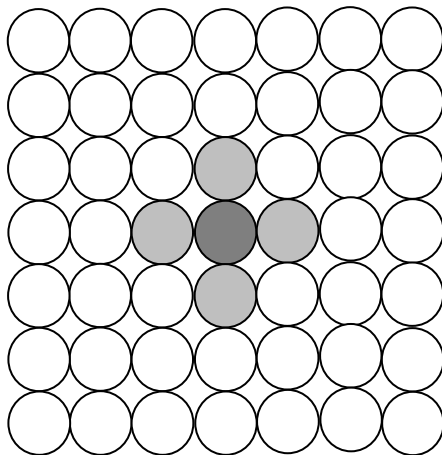
There is one more method to test the point is inside the polygon or not. It can be tested by finding the sum of all the angles created with the end points of all the edges of the polygons. If the sum of all the angles is 2π , then the point is said to be the interior point, otherwise exterior point. In the fig.3.9 (a) the sum of the angles created is 2π so it is an interior point. This method is the simplest one as well as effective with the self-intersecting polygons too.

3.3 Polygon Area Filling Primitives

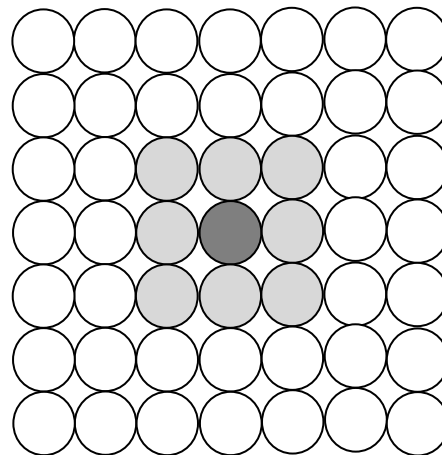
Basically all closed figure in real world are solid objects, where as the polygons we are discussing till now are just the outline figures. For making them solid object it is required to fill them with certain colors or the patterns. This filling can be done by changing the colors of the pixels belonging to the polygons. Different methods for filling the polygons are: (1) Flood Fill Method (2) Scan-line Fill Method and (3) Pattern Fill Method.

3.3.1 Flood Fill Method

The flood fill method is also known as Seed Fill Method. We need a point inside the polygon here for starting the filling. This point is called the seed point. From the seed point the point is taken and the default color is being replaced by the fill color. The same process is repeated till the boundary color pixel is encountered. This process is preformed in all the directions. There are basically two ways to fill the polygon: Four connected point neighborhood and eight connected point neighborhood methods. In four connected point method color of four surrounded points of the seed point will be changed with the fill color are left-right and upper-lower of the seed point. Whereas in eight connected point neighborhood method all surrounded points of the seed point will be replaced with the fill color (see fig. 3.10). There is a limitation of the flood fill algorithm; it does not work with the overlapping and self-intersecting polygons. Such polygons are treated as a collection of different polygons.



(a) Four Connected neighborhood points



(b) Eight Connected neighborhood points

Fig. 3.10 – Flood Fill Methods

In case if there is no seed point defined then the fill can be done by turning all the pixels on within the boundary of the polygon. The difficulty in this manner is first we have to test the point is inside the polygon or not, if it is inside the polygon then turn the pixel on and change the default color with the fill color. It seems to be a very costly alternative. The algorithm can be written as follows:

Flood Fill Algorithm [FloodFill (seedx, seedy, fcol, dcol)]**(4 connected neighborhood point)**

- Step-1:** Initialize the value of seed point (seedx, seedy), fcolor and dcol.
- Step-2:** Define the boundary values of the polygon.
- Step-3:** Check if the current seed point is of default color then repeat the steps 4 and 5 till the boundary pixels reached
If `getpixel(x,y) = dcol` then repeat step 4 and 5
- Step-4:** Change the default color with the fill color at the seed point.
`setPixel(seedx, seedy, fcol)`
- Step-5:** Recursively follow the procedure with four neighborhood points.
`FloodFill (seedx - 1, seedy, fcol, dcol)`
`FloodFill (seedx + 1, seedy, fcol, dcol)`
`FloodFill (seedx, seedy - 1, fcol, dcol)`
`FloodFill (seedx - 1, seedy + 1, fcol, dcol)`
- Step-6:** Exit

(8 connected neighborhood point)

- Step-1:** Initialize the value of seed point (seedx, seedy), fcolor and dcol.
- Step-2:** Define the boundary values of the polygon.
- Step-3:** Check if the current seed point is of default color then repeat the steps 4 and 5 till the boundary pixels reached
If `getpixel(x,y) = dcol` then repeat step 4 and 5
- Step-4:** Change the default color with the fill color at the seed point.
`setPixel(seedx, seedy, fcol)`
- Step-5:** Recursively follow the procedure with four neighborhood points.
`FloodFill (seedx - 1, seedy, fcol, dcol)`
`FloodFill (seedx + 1, seedy, fcol, dcol)`
`FloodFill (seedx, seedy - 1, fcol, dcol)`
`FloodFill (seedx, seedy + 1, fcol, dcol)`
`FloodFill (seedx - 1, seedy + 1, fcol, dcol)`
`FloodFill (seedx + 1, seedy + 1, fcol, dcol)`
`FloodFill (seedx + 1, seedy - 1, fcol, dcol)`
`FloodFill (seedx - 1, seedy - 1, fcol, dcol)`
- Step-6:** Exit

The flood fill method can be used with any one algorithm as per the requirement of the user. The 8 connected neighborhood point method is more fast then the other one.

3.3.2 Scan-Line Fill Method

The scan-line fill method is very efficient and cheaper alternative. It is also known as alternative fill method. In this method scan lines are used for filling the polygons. It is working effectively with the self intersecting polygons.

In this we have to use the smallest rectangle which fits the polygon. Which needs the rectangle to be created from the lower left corner to the upper right corner coordinates of the polygon. For generating the scan lines we have to consider the largest Y value and move towards the smallest Y value and draw a horizontal line among all possible values of Y. Since we are using scan-lines to fill the polygon this method is known as scan-line fill method.

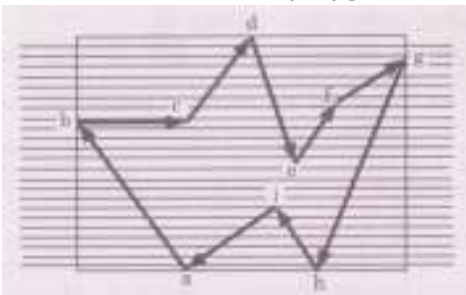


Fig. 3.11 – A Rectangle around Polygon

For a start we have to assume that we have boundary of a blank polygon in empty frame buffer, which is bounded by the smallest rectangle around it. In that fig.3.11 we can see the scan-lines are line by line intersecting the polygon edges.

When we encounter the pixel with the intensity of boundary pixel, we have to change the default color to fill color till we encounter other boundary pixel as on shown in the fig.3.12. The major problem here is when two edges of the polygon intersecting and the scan-line is passing from that intersection point. Here we have to consider the values of the edges endpoint which are stored in the frame buffer.

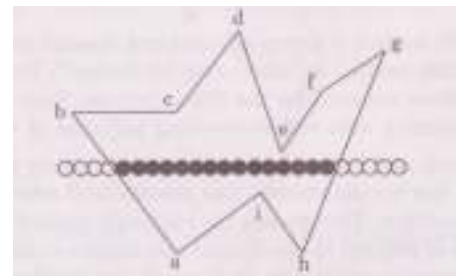


Fig. 3.12- Pixels Among two boundaries of polygon

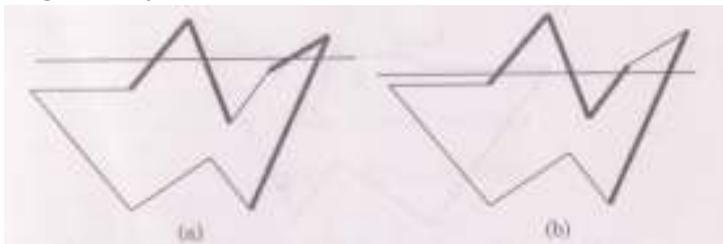


Fig. 3.13- Edges under consideration at different scan lines

When we are moving from scan-line by scan-line we have to check which edges are being considered and which edges to be dropped from the consideration. This can be done by checking the frame buffer for different set of y indexes in the frame buffer. Fig. 3.13 shows the edges which are under consideration for different scan-lines.

Now there is a problem when the scan-line is passing from the polygon and certain portion of the scan-line is not inside the polygon where as other portion is inside the polygon (see figure 3.14). For solving this we have to draw a line between two successive intersections starting with the odd number to the next successive point. In other words,

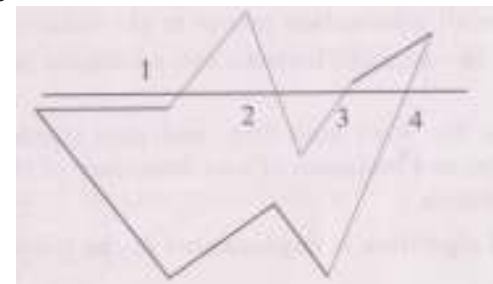


Fig. 3.14- Scan lines with more than 2 intersections

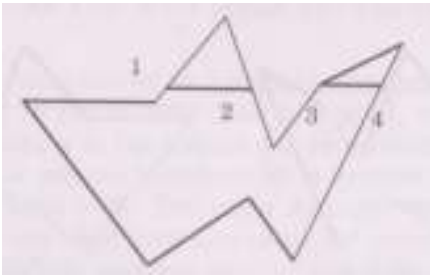


Fig. 3.14– line segments for 2
Successive intersections

arrange all intersection points in the order of their x values; the smallest x value is the left-most boundary and the largest x value will be on the right-most boundary of the polygon. Then draw a line segment between two successive pairs of the x values (see figure 3.15).

Note that whenever our scan-line is passing from an intersection of two edges the points will be considered as per the even-odd method. So the steps of the algorithm can be mentioned as follow:

Scan-line Fill Algorithm

- Step-1:** Determine the smallest possible rectangle which fits in the polygon to be filled.
- Step-2:** Arrange all the edges of polygon in the order of y values.
- Step-3:** Determine all the polygon edges which are required to be considered with current scan-line.
- Step-4:** Find the intersection points between the scan-line and all the edges under consideration.
- Step-5:** Arrange all intersection points in the order of their x values.
- Step-6:** Draw a line segment between two successive pairs of intersection points.
- Step-7:** Shift to the next scan-line, and also check for the possible exclusion and inclusion of new boundary of the polygon for the consideration.

Another approach to this algorithm is finding the pixels of boundary color on the scan-line, which is nothing but the intersection points of the scan-lines and the polygon edges. It will work when only one polygon of the given color is there on the screen.

3.3.3 Filling with Pattern

Sometime it is required to fill the polygon with certain patterns instead of colors. In such cases other polygon filling method cannot be used. As we are using the flood fill method to change the color of the interior points of the polygon, in similar way we have to use an appropriate pattern pixel and accordingly it will be colored.

For generating such patterns we are using the pattern matrices as shown in the figure besides.

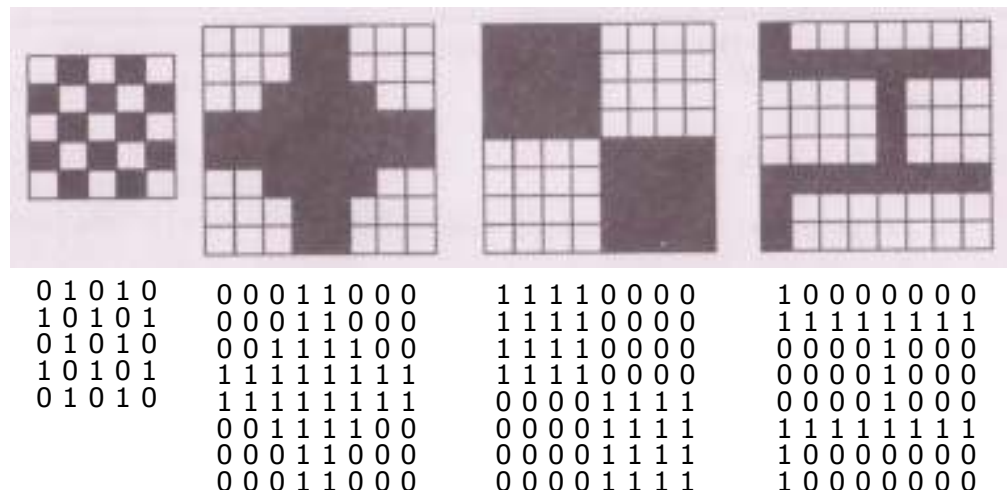


Fig. 3.16 – various patterns and their respective frame buffers.

According to this algorithm we have to first construct a smallest rectangle which fits the polygon in it. Then take the first pixel from the pattern matrix and check the first pixel of the rectangle and if the pixel of rectangle is inside the polygon then copy the details of pattern matrix

to the first pixel of the polygon. We have to repeat this procedure for all the pixels which are interior to the polygon (see figure 3.17).

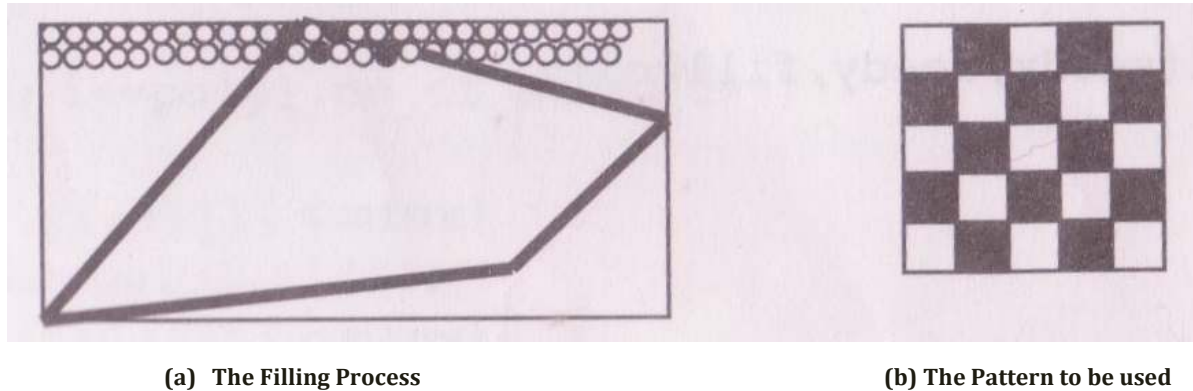


Fig. 3.17 – Pattern Fill Method

Pattern Fill Algorithm

- Step-1:** Determine the smallest possible rectangle which fits the polygon in it.
- Step-2:** Generate the pattern matrix to be used for the polygon filling.
- Step-3:** Check the first pixel of rectangle with the polygon if it is interior of the polygon then move to step-4 otherwise move to step-5.
- Step-4:** Copy the details of the pattern matrix to the point we found as an interior point of the polygon.
- Step-5:** repeat the procedure from step-3 until the complete polygon filled with the pattern.
- Step-6:** Exit