Assignment No - A-4.

Student Name :- Sameer Manoj Bramhecha
Roll No :- 21115
Batch :- E-1
Date of Performance :- 01/10/2021
Date of Submission :- 05/10/2021

Title :- A concave polygon filling using scan fill algorithm.

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

Learning objective :- To understand and implement scanline polygon fill algorithm.

Learning Outcomes :- After completion of this assignment, student will be able to implement scanline fill algorithm for polygon.

S/W and H/W Requirements :-
1.) Basic programming skills of C++
2.) 64-bit Open Source Linux.
3.) Open Source C++ Programming tool like G++/GCC.

Date
Page

Theory:-

Polygon:- A polygon is a closed planar path composed of a finite number of sequential line segments. A polygon is a two-dimensional shape formed with more than three straight lines. When starting point and terminal point is same then it is called polygon.

Types of Polygon:-

1.) Concave
2.) Convex
3.) Complex.

A convex polygon is a simple polygon whose interior is a convex set. In a convex polygon, all interior angles are less than 180 degrees.

The following properties of a simple polygon are all equivalent to convexity.

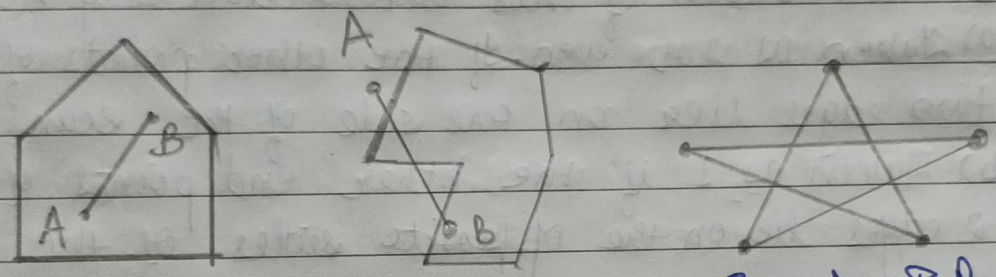
- Every internal angle is less than or equal to 180 ~~deg~~ degrees.
- Every line segment between two vertices remains ~~inside~~ inside or on the boundary of the polygon.

In a convex polygon, any line segment joining ~~A~~ any two inside points lies inside the polygon.

A straight line drawn through a convex polygon crosses at most two sides.

A concave polygon will always have an interior angle greater than 180 degrees. It is possible to cut a concave polygon into a set of convex polygon. You can draw at least one straight line through a concave polygon that crosses more than two sides.

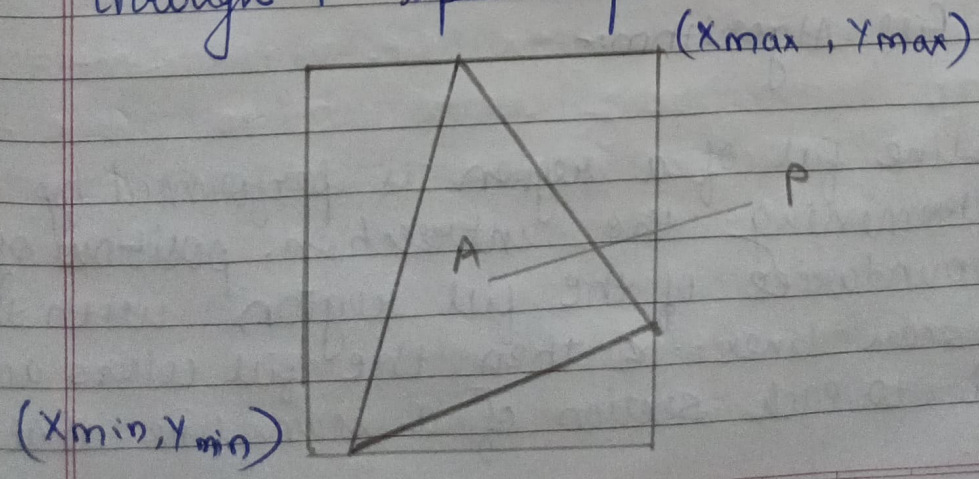
Complex polygon is a polygon whose sides cross over each other one or more times.



Convex polygon Concave Polygon Complex Polygon.

• Inside-Outside test [Even-Odd Test]:-
We assume that the vertex list for the polygon is already stored and proceed as follows:-

1.) Draw any point outside the ~~edge~~ range Xmin and Xmax and Ymin and Ymax. Draw a scan line through P upto a point A under study.



2.) If this scan line

i) Does not pass through any of the vertices then its contribution is equal to the number of times it intersects the edges of the polygon. Say C if:

a) C is odd then A lies inside the polygon.

b) C is even then it lies outside the polygon.

ii) If it passes through any of the vertices then the contribution of this intersection say V is,

a) Taken as 2 or even. If the other points of the two edges lies on one side of the scan line.

b.) Taken as 1 if the other end points of the 2 edges lie on the opposite sides of the scan-line

c.) Here will be total contribution is C+V.

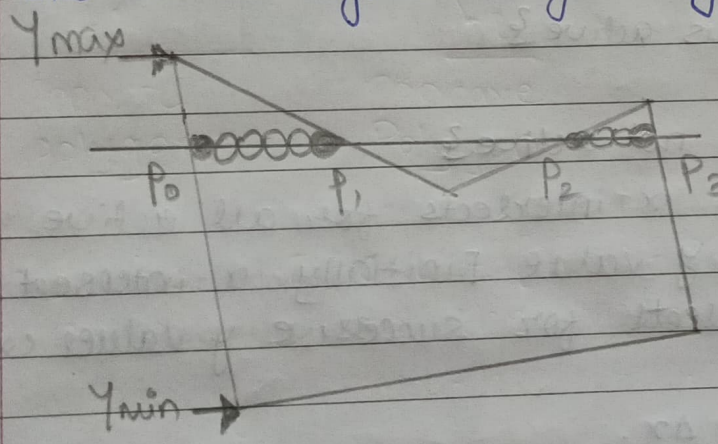
* Polygon Filling:→

For filling polygons with particular colors, you need to determine pixels falling on the border of the polygon and those which fall inside the polygon.

* Scan-Fill Algorithm:-

A scan-line fill of a region is performed by first determining the intersection positions of the boundaries of the fill region with the screen scan lines. Then the fill colors are applied to each section of a scan line that

lies within the interior of the fill region. The scan line fill algorithm identifies the same interior regions as the odd-even rule.
It is an image space algorithm. It processes one line at a time rather than one pixel at a time. It uses the concept area of coherence. This algorithm records edge list, active edge list. So accurate ~~bookkep.~~ bookeeping is necessary. The edge list or edge table contains the coordinate of two endpoints. Active Edge List (AEL) contain edges a given scan line ~~to~~ intersects during its sweep. The active edge list (AEL) should be sorted in increasing order of x. The AEL is dynamic, growing and shrinking.



Algorithm:→

1.) Start.
2.) Read n, the number of vertices of polygon.
3.) Read x and y coordinates of all vertices in array x[n] and y[n].
4.) Find ymin and ymax.
5.) Store the initial x value (x_1), y values y_1 and y_2 for two endpoints and x increment Δx from scan line to scan for each edge in the array edges [n][4].

~~[n][4]~~ While doing this check that $y_1 > y_2$, if not interchange y_1 and y_2 and corresponding x_1 and x_2 so that for each edge y_1 represents its maximum y-coordinate and y_2 represents its minimum y coordinate.

6.) Sort the rows of array, edges [n][4] in descending order of y_1, descending order of y_2 & ascending order of x_2.

7) Set $y = y_{max}$

8.) Find the active edges and update active edge list:

if ($y > y_2$ and $y \le y_1$)
{ edge is active }
else
{ edge is not active }

9.) Compute the x intersects for all active edges for current y value [initially x-intersect is x_1 and x intersects for successive y values can be given as

$x_{i+1} \leftarrow x_i + \Delta x$.

where $\Delta x = \frac{-1}{m}$ & $m = \frac{y_2 - y_1}{x_2 - x_1}$. i.e. slope of a line segment.

10.) If x intersect is vertex i.e. x-intersect $= x_1$ and $y = y_1$ then apply vertex test to check whether to consider one intersect or two intersects. Store all x-intersects in the x-intersect [] array.

11.) Sort x-intersect [] array in the ascending order.

12.) Extract pairs of intersect from the sorted x-intersect [] array.

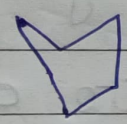
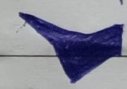
13.) Pass pairs of x values to line drawing routine to draw corresponding line segments.

14.) Set $y = y - 1$

15.) Repeat steps 7 → 13 until $y \geq y_{min}$

16.) ~~Stop~~ Stop.

Test Cases:→

Test case No.	Test case Description	Input.	Expected Output.	Actual Output	Result.
1.	Concave Polygon				Pass
2.	Convex Polygon				Pass.

Conclusion:-

We have successfully learnt and implemented scan line fill algorithm for polygon filling.