Unit : II

Polygon, Windowing and Clipping

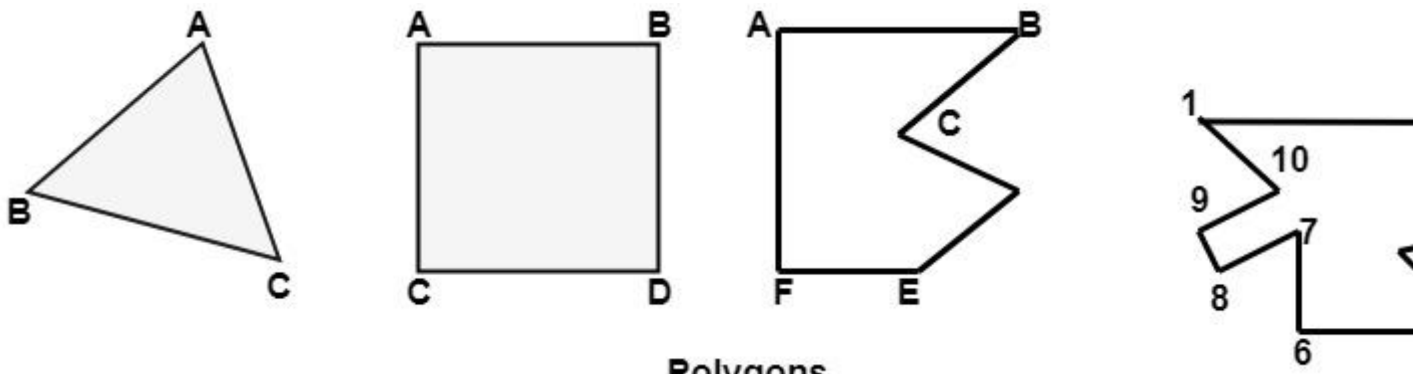
Polygon:

- Polygon is a representation of the surface. It is primitive which is closed in nature. It is formed using a collection of lines.
- It is also called as many-sided figure.
- The lines combined to form polygon are called sides or edges. The lines are obtained by combining two vertices.

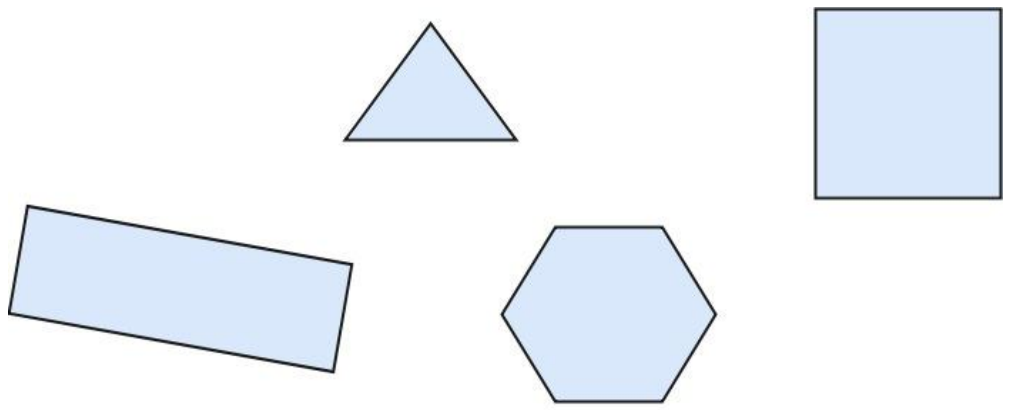
Example of Polygon:

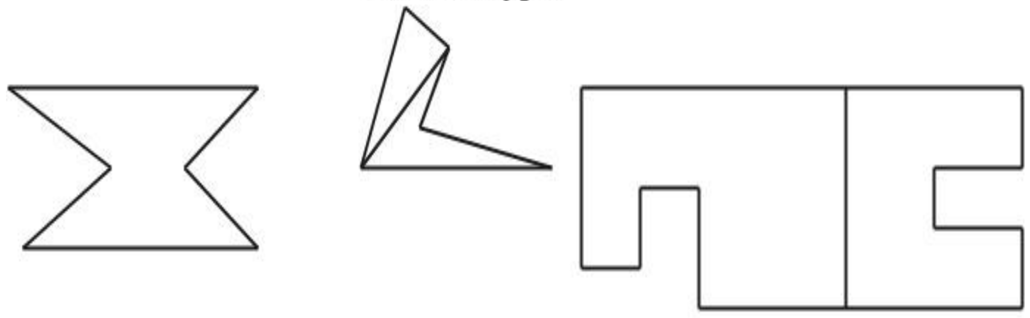
1. Triangle
2. Rectangle
3. Hexagon
4. Pentagon

Following figures shows some polygons.



Polygons

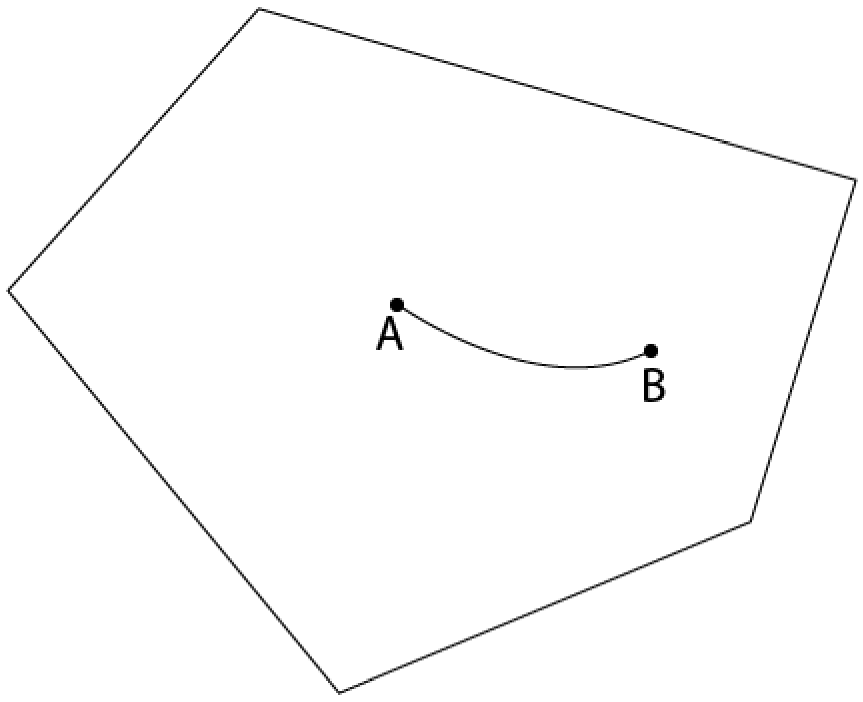

Convex Polygon


Concave Polygon

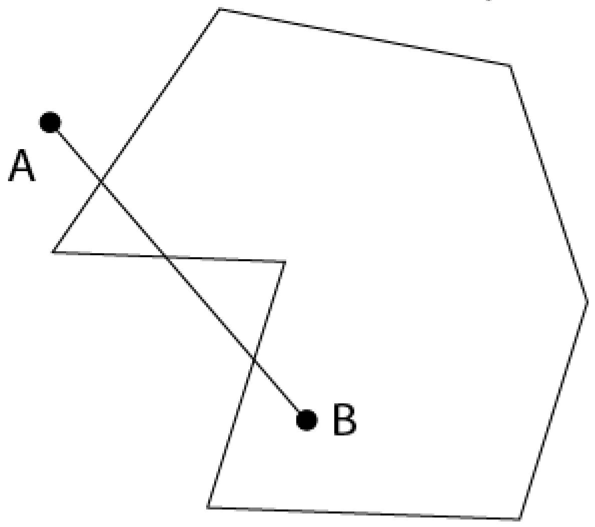
Types of Polygons

1. Concave
2. Convex

- A polygon is called convex of line joining any two interior points of the polygon lies inside the polygon.
- A non-convex polygon is said to be concave. A concave polygon has one interior angle greater than 180°.
- So that it can be clipped into similar polygons.

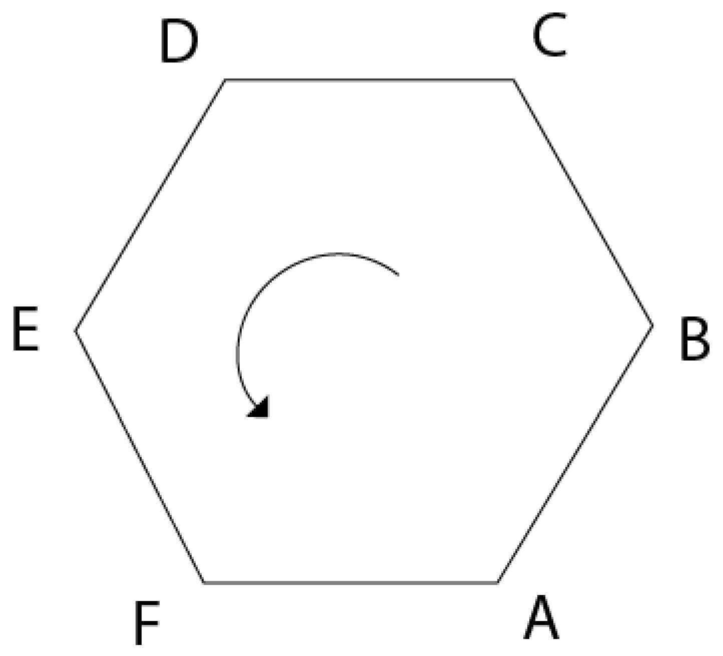


Convex polygon

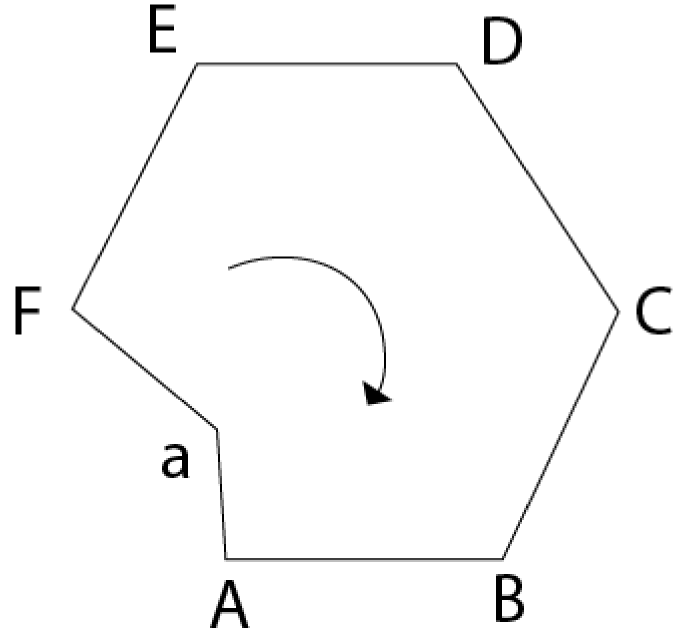


Concave polygon

A polygon can be positive or negative oriented. If we visit vertices and vertices visit produces counterclockwise circuit, then orientation is said to be positive.



Polygon with positive orientation



Polygon with negative orientation

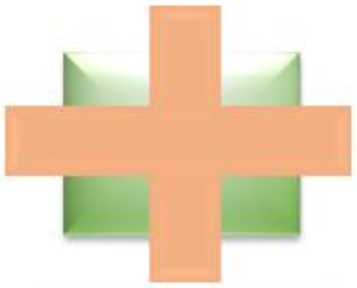
Sutherland-Hodgeman Polygon Clipping:

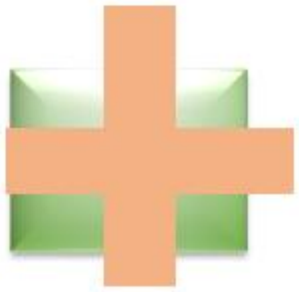
It is performed by processing the boundary of polygon against each window corner or edge. First of all entire polygon is clipped against one edge, then resulting polygon is considered, then the polygon is considered against the second edge, so on for all four edges.

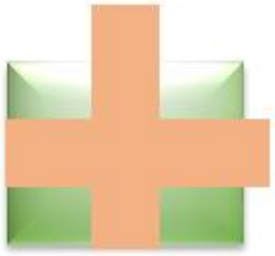
Four possible situations while processing

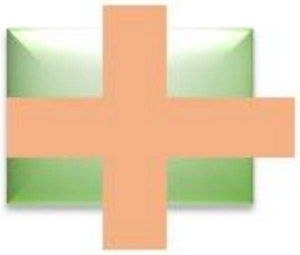
1. If the first vertex is an outside the window, the second vertex is inside the window. Then second vertex is added to the output list. The point of intersection of window boundary and polygon side (edge) is also added to the output line.
2. If both vertexes are inside window boundary. Then only second vertex is added to the output list.
3. If the first vertex is inside the window and second is an outside window. The edge which intersects with window is added to output list.
4. If both vertices are the outside window, then nothing is added to output list.

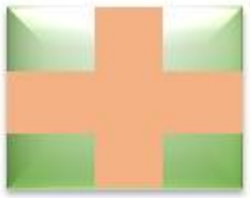
Following figures shows original polygon and clipping of polygon against four windows.

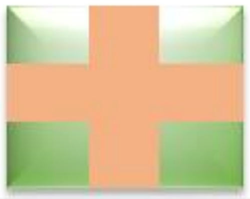

Original polygon


Clipping against left edge of window


Clipping against bottom corner of window


Clipping against Top of window


Clipping against right edge of window


Final polygon after Clipping

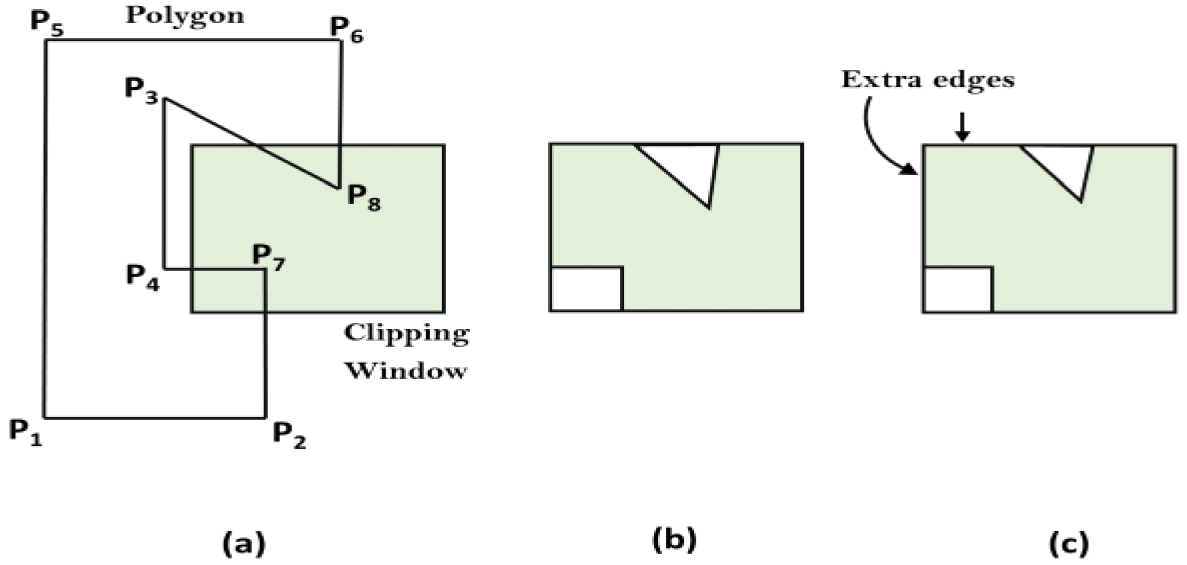
Disadvantage of Cohen Hodgmen Algorithm:

- This method requires a considerable amount of memory. The first of all polygons are stored in original form.
- Then clipping against left edge done and output is stored. Then clipping against right edge done, then top edge.
- Finally, the bottom edge is clipped. Results of all these operations are stored in memory.
- So wastage of memory for storing intermediate polygons.

Weiler-Atherton Polygon Clipping:

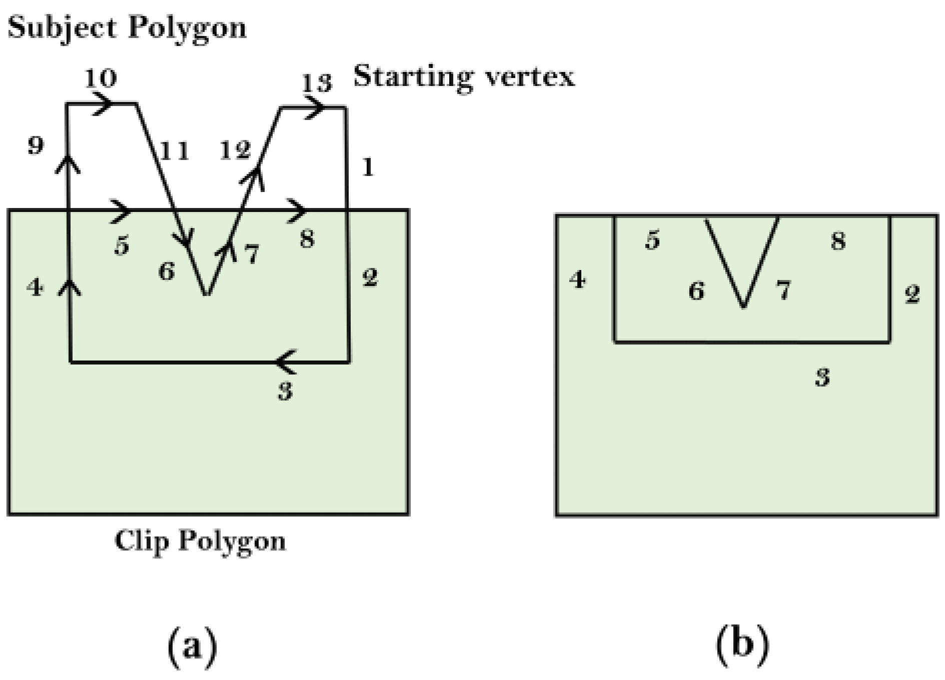
- When the clipped polygons have two or more separate sections, then it is the concave polygon handled by this algorithm.
- The vertex-processing procedures for window boundaries are modified so that concave polygon is displayed.
- Let the clipping window be initially called clip polygon and the polygon to be clipped the subject polygo.
- We start with an arbitrary vertex of the subject polygon and trace around its border in the clockwise direction until an intersection with the clip polygon is encountered:

1. If the edge enters the clip polygon, record the intersection point and continue to trace the subject polygon.



2. If the edge leaves the clip polygon, record the intersection point and make a right turn to follow the clip polygon in the same manner (i.e., treat the clip polygon as subject polygon and the subject polygon as clip polygon and proceed as before).

Whenever our path of traversal forms a sub-polygon we output the sub-polygon as part of the overall result. We then continue to trace the rest of the original subject polygon from a recorded intersection point that marks the beginning of a not-yet traced edge or portion of an edge. The algorithm terminates when the entire border of the original subject polygon has been traced exactly once.



For example, the number in fig (a) indicates the order in which the edges and portion of edges are traced. We begin at the starting vertex and continue along the same edge (from 1 to 2) of the subject polygon as it enters the clip polygon. As we move along the edge that is leaving the clip polygon, we make a right turn (from 4 to 5) onto the clip polygon, which is now considered the subject polygon. Following the same logic leads to the next right turn (from 5 to 6) onto the current clip polygon, this is the original subject polygon. With the next step done (from 7 to 8) in the same way, we have a sub-polygon for output in fig (b). We then resume our traversal of the original subject polygon from the recorded intersection point where we first changed our course. Going from 9 to 10 to 11 produces no output. After skipping the already traversed 6 and 7, we continue with 12 and 13 and come to an end. The fig (b) is the final result.

Clipping:

When we have to display a large portion of the picture, then not only scaling & translation is necessary, the visible part of picture is also identified. This process is not easy. Certain parts of the image are inside, while others are partially inside. The lines or elements which are partially visible will be omitted.

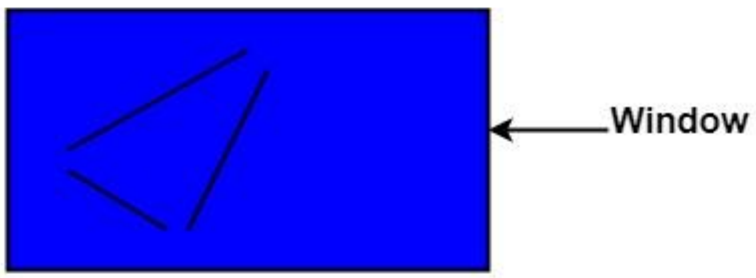
For deciding the visible and invisible portion, a particular process called clipping is used. Clipping determines each element into the visible and invisible portion. Visible portion is selected. An invisible portion is discarded.

Types of Lines:

Lines are of three types:

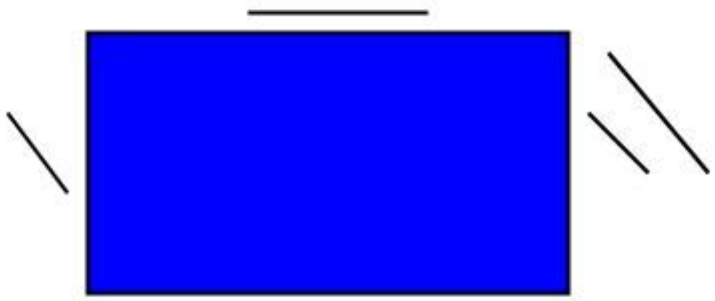
1. **Visible:** A line or lines entirely inside the window is considered visible
2. **Invisible:** A line entirely outside the window is considered invisible
3. **Clipped:** A line partially inside the window and partially outside is clipped. For clipping point of intersection of a line with the window is determined.

Case1:



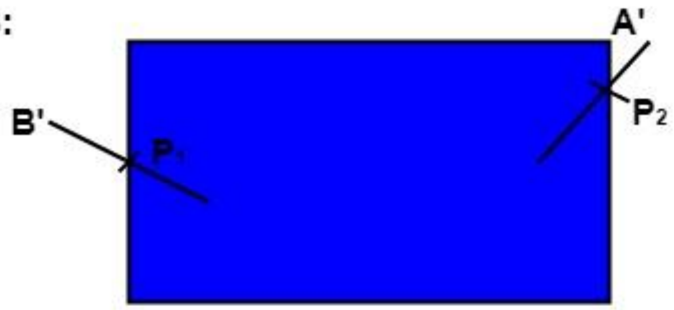
Visible Lines

Case2:



Invisible Lines

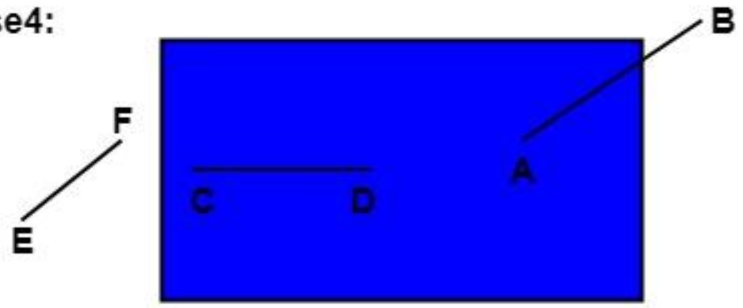
Case3:



In this figure P_1 and P_2 are point of intersection. The line P_2 to A' and P to B' is discarded or clipped.

Clipped Lines

Case4:

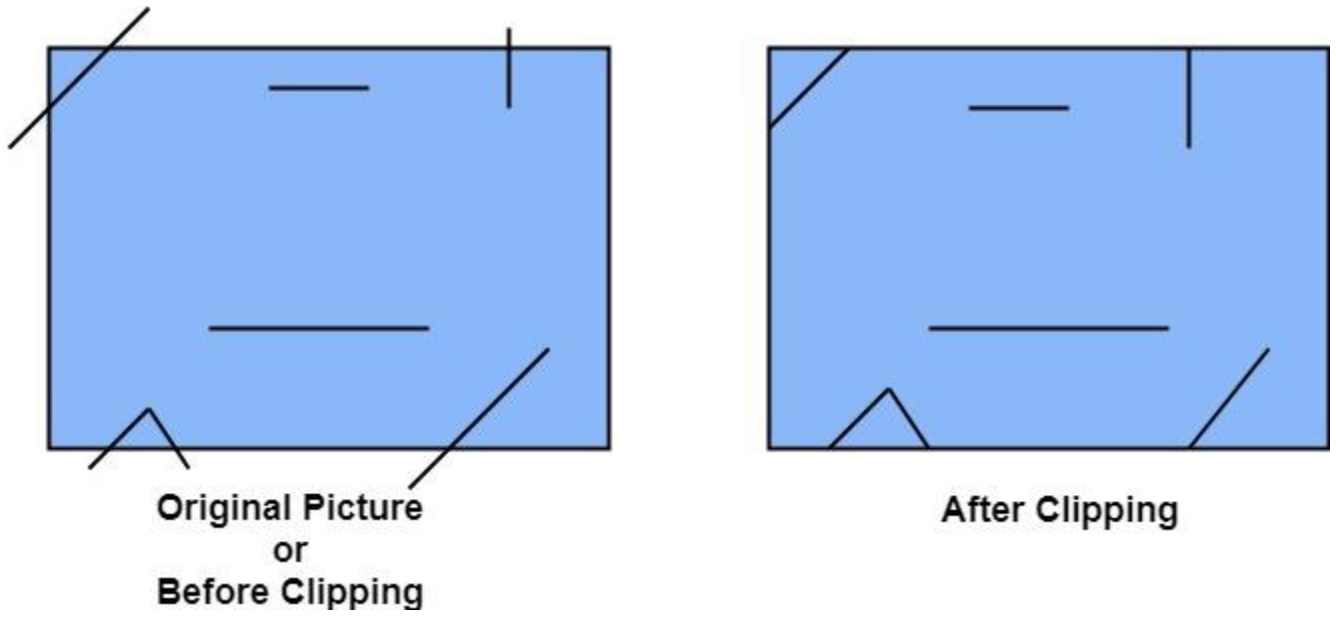


In this figure AB is clipped case
CD is visible line.
EF is invisible line.

Mixed Lines

Clipping can be applied through hardware as well as software. In some computers, hardware devices automatically do work of clipping. In a system where hardware clipping is not available software clipping applied.

Following figure show before and after clipping



The window against which object is clipped called a clip window. It can be curved or rectangle in shape.

<u>Applications of clipping:</u>

1. It will extract part we desire.
2. For identifying the visible and invisible area in the 3D object.
3. For creating objects using solid modeling.
4. For drawing operations.
5. Operations related to the pointing of an object.
6. For deleting, copying, moving part of an object.

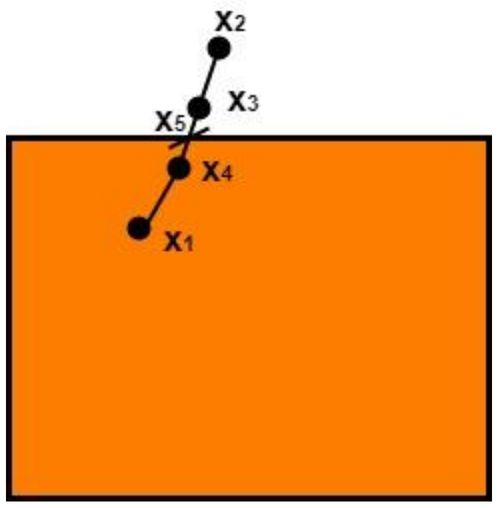
Clipping can be applied to world co-ordinates. The contents inside the window will be mapped to device co-ordinates. Another alternative is a complete world co-ordinates picture is assigned to device co-ordinates, and then clipping of viewport boundaries is done.

Types of Clipping:

1. Point Clipping
2. Line Clipping
3. Area Clipping (Polygon)
4. Curve Clipping
5. Text Clipping
6. Exterior Clipping

Mid Point Subdivision Line Clipping Algorithm:

- It is used for clipping line.
- The line is divided in two parts.
- Mid points of line is obtained by dividing it in two short segments.
- Again division is done, by finding midpoint.
- This process is continued until line of visible and invisible category is obtained.
- Let (xi,yi) $x_m = \frac{x_1+x_2}{2}$ $y_m = \frac{y_1+y_2}{2}$ are midpoint



Step1: Find $\frac{x_2+x_1}{2}$ i.e. $x_3 = \frac{x_2+x_1}{2}$

Step2: Find $x_4 = \frac{x_3+x_1}{2}$

Step3: Find $x_5 = \frac{x_4+x_3}{2}$

$x_5$lie on point of intersection of boundary of window.

Advantage of midpoint subdivision Line Clipping:

It is suitable for machines in which multiplication and division operation is not possible. Because it can be performed by introducing clipping divides in hardware.

Algorithm of midpoint subdivision Line Clipping:

Step1: Calculate the position of both endpoints of the line

Step2: Perform OR operation on both of these endpoints

Step3: If the OR operation gives 0000

```
then
        Line is guaranteed to be visible
  else
      Perform AND operation on both endpoints.
      If AND $\neq$ 0000
  then the line is invisible
```

```
    else
        AND=6000
        then the line is clipped case.
```

Step4: For the line to be clipped. Find midpoint

$X_m=(x_1+x_2)/2$

$Y_m=(y_1+y_2)/2$

$X_m$is midpoint of X coordinate.

$Y_m$is midpoint of Y coordinate.

Step5: Check each midpoint, whether it nearest to the boundary of a window or not.

Step6: If the line is totally visible or totally rejected not found then repeat step 1 to 5.

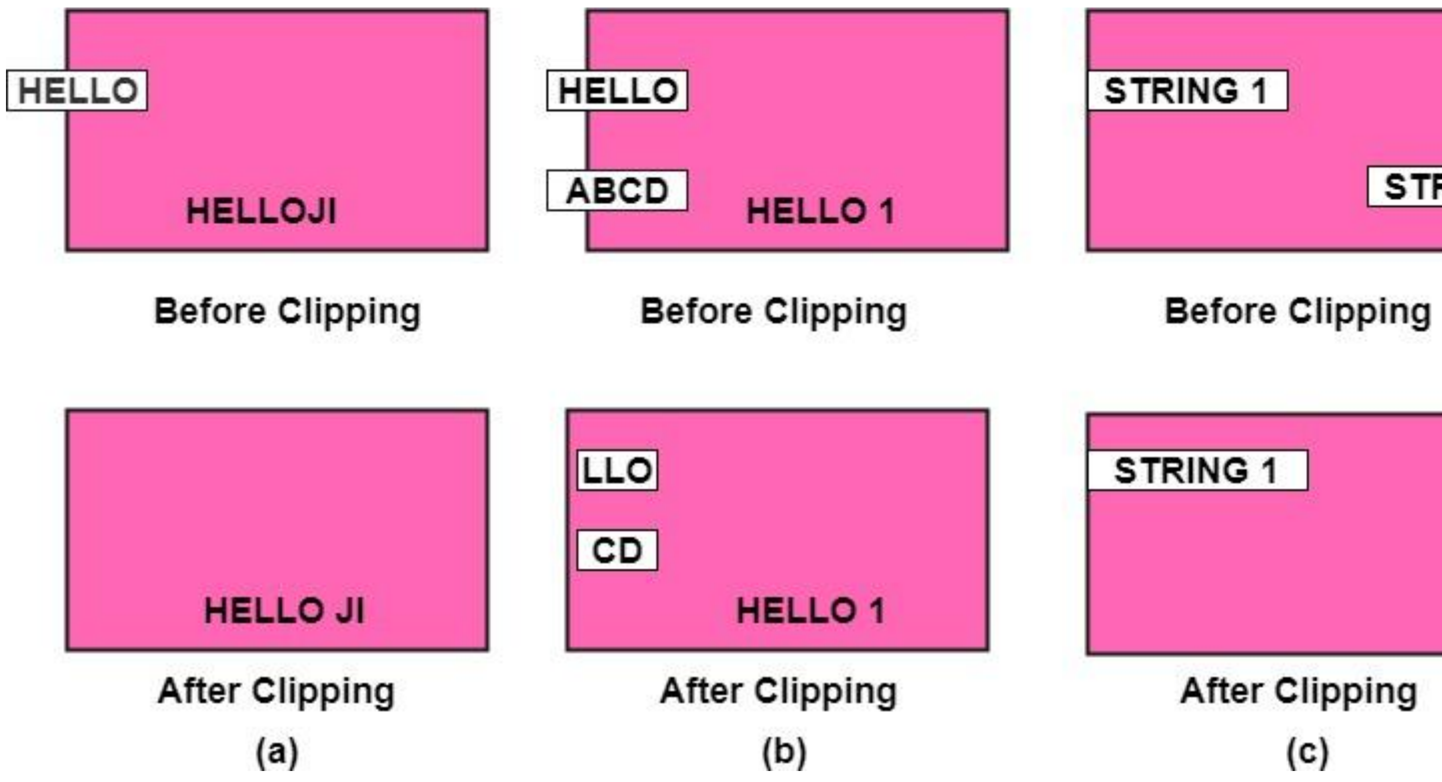
Step7: Stop algorithm.

Text Clipping:

Several methods are available for clipping of text. Clipping method is dependent on the method of generation used for characters. A simple method is completely considered, or nothing considers method. This method is also called as all or none. If all characters of the string are inside window, then we will keep the string, if a string character is outside then whole string will be discarded in fig (a).

Another method is discarded those characters not completely inside the window. If a character overlap boundary of window. Those will be discarded in fig (b).

In fig (c) individual character is treated. Character lies on boundary is discarded as which it is outside the window.

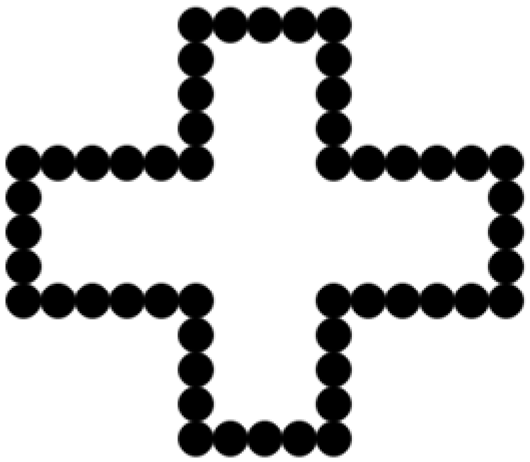


Polygon Clipping:

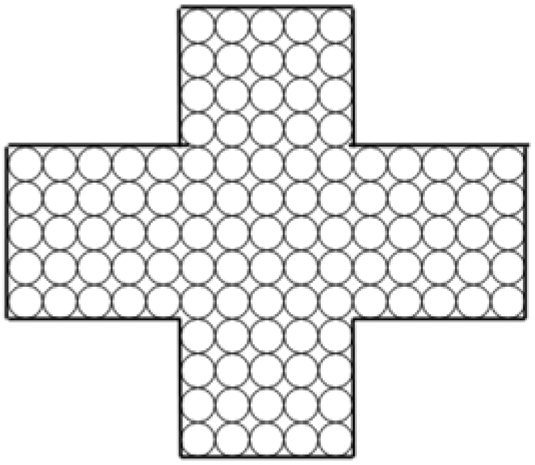
Polygon clipping is applied to the polygons. The term polygon is used to define objects having outline of solid. These objects should maintain property and shape of polygon after clipping.

Filled Area Primitives:

Region filling is the process of filling image or region. Filling can be of boundary or interior region as shown in fig. Boundary Fill algorithms are used to fill the boundary and flood-fill algorithm are used to fill the interior.



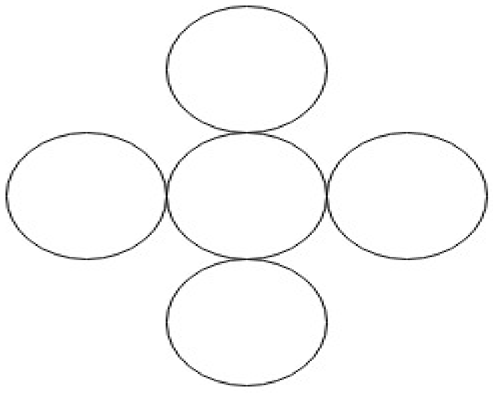
Boundary Filled Region



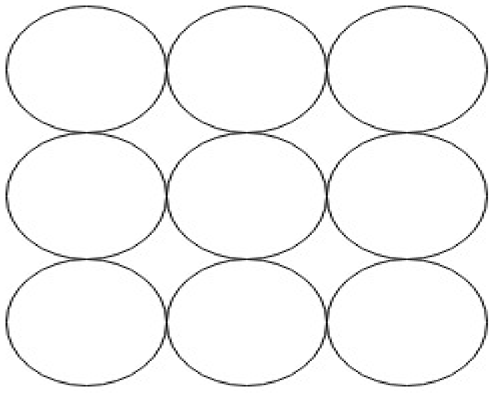
Interior or Flood Filled Region

Boundary Filled Algorithm:

- This algorithm uses the recursive method.
- First of all, a starting pixel called as the seed is considered.
- The algorithm checks boundary pixel or adjacent pixels are colored or not.
- If the adjacent pixel is already filled or colored then leave it, otherwise fill it.
- The filling is done using four connected or eight connected approaches.



Four Connected



Eight Connected

Four connected approaches is more suitable than the eight connected approaches.

1. Four connected approaches: In this approach, left, right, above, below pixels are tested.

2. Eight connected approaches: In this approach, left, right, above, below and four diagonals are selected.

Boundary can be checked by seeing pixels from left and right first. Then pixels are checked by seeing pixels from top to bottom. The algorithm takes time and memory because some recursive calls are needed.

Problem with recursive boundary fill algorithm:

It may not fill regions sometimes correctly when some interior pixel is already filled with color. The algorithm will check this boundary pixel for filling and will found already filled so recursive process will terminate. This may vary because of another interior pixel unfilled.

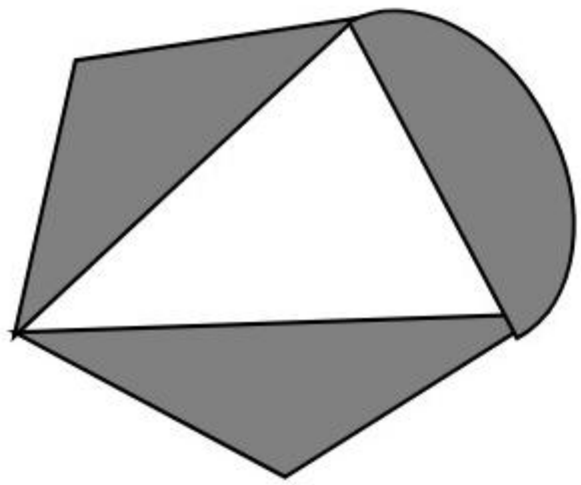
So check all pixels color before applying the algorithm.

Algorithm:

```
Procedure fill (x, y, color, color1: integer)
int c;
c=getpixel (x, y);
if (c!=color) (c!=color1)
{
    setpixel (x, y, color)
    fill (x+1, y, color, color 1);
    fill (x-1, y, color, color 1);
    fill (x, y+1, color, color 1);
    fill (x, y-1, color, color 1);
}
```

Flood Fill Algorithm:

- In this method, a point or seed which is inside region is selected.
- This point is called a seed point.
- Then four connected approaches or eight connected approaches is used to fill with specified color.
- The flood fill algorithm has many characters similar to boundary fill.
- But this method is more suitable for filling multiple colors boundary.
- When boundary is of many colors and interior is to be filled with one color we use this algorithm.



- In fill algorithm, we start from a specified interior point (x, y) and reassign all pixel values are currently set to a given interior color with the desired color.
- Using either a 4-connected or 8-connected approaches, we then step through pixel positions until all interior points have been repainted.

Disadvantage:

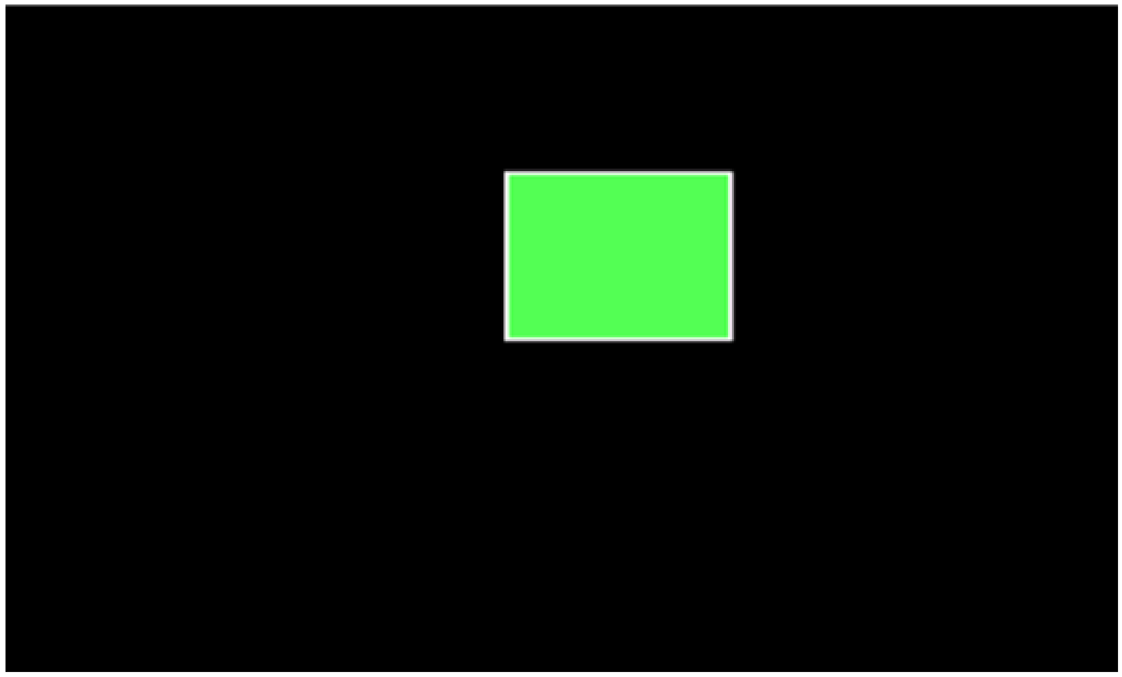
- Very slow algorithm
- May be fail for large polygons
- Initial pixel required more knowledge about surrounding pixels.
- Algorithm:

```
Procedure floodfill (x, y,fill_ color, old_color: integer)
   If (getpixel (x, y)=old_color)
  {
   setpixel (x, y, fill_color);
   fill (x+1, y, fill_color, old_color);
    fill (x-1, y, fill_color, old_color);
   fill (x, y+1, fill_color, old_color);
   fill (x, y-1, fill_color, old_color);
    }
      }
```

Program1: To implement 4-connected flood fill algorithm:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void flood(int,int,int,int);
void main()
{
   intgd=DETECT,gm;
   initgraph(&gd,&gm,"C:/TURBOC3/bgi");
        rectangle(50,50,250,250);
        flood(55,55,10,0);
        getch();
    }
    void flood(intx,inty,intfillColor, intdefaultColor)
    {
        if(getpixel(x,y)==defaultColor)
        {
           delay(1);
           putpixel(x,y,fillColor);
           flood(x+1,y,fillColor,defaultColor);
           flood(x-1,y,fillColor,defaultColor);
           flood(x,y+1,fillColor,defaultColor);
           flood(x,y-1,fillColor,defaultColor);
        }
    }
```

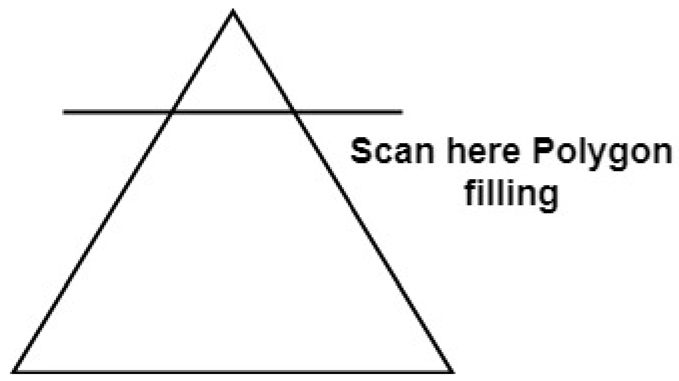
Output:



<u>Scan Line Polygon Fill Algorithm:</u>

This algorithm lines interior points of a polygon on the scan line and these points are done on or off according to requirement. The polygon is filled with various colors by coloring various pixels.

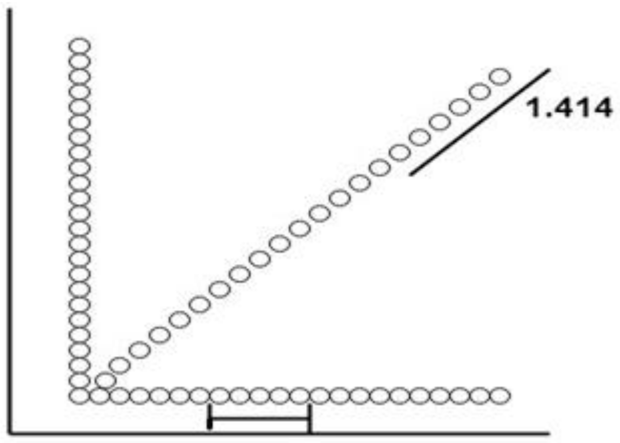
In above figure polygon and a line cutting polygon in shown. First of all, scanning is done. Scanning is done using raster scanning concept on display device. The beam starts scanning from the top left corner of the screen and goes toward the bottom right corner as the endpoint. The algorithms find points of intersection of the line with polygon while moving from left to right and top to bottom. The various points of intersection are stored in the frame buffer. The intensities of such points is keep high. Concept of coherence property is used. According to this property if a pixel is inside the polygon, then its next pixel will be inside the polygon.



Side effects of Scan Conversion:

1. Staircase or Jagged: Staircase like appearance is seen while the scan was converting line or circle.

2. Unequal Intensity: It deals with unequal appearance of the brightness of different lines. An inclined line appears less bright as compared to the horizontal and vertical line.



Pixels along with horizontal line are 1 unit apart and vertical.
Pixels along diagonal line are 1.414 units.