# Introduction to Computer Graphics

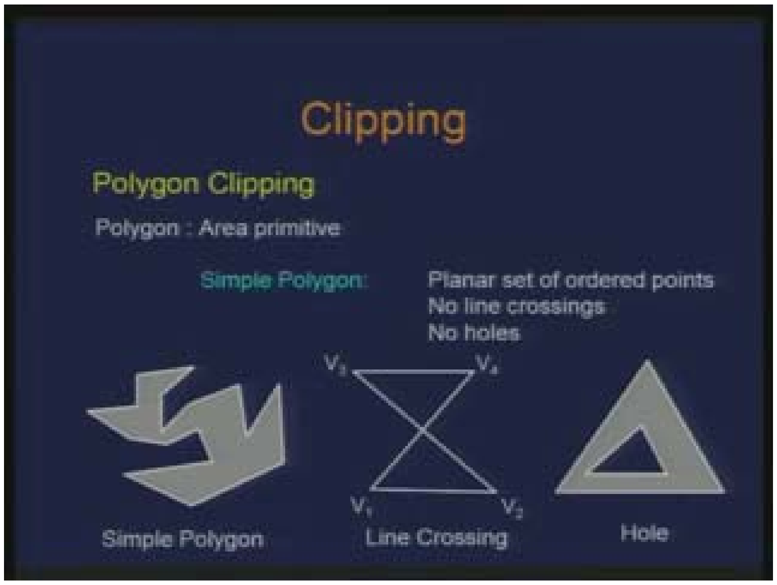
**Dr. Prem Kalra**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Delhi**

**Lecture - 5**

**Polygon Clipping and Polygon Scan Conversion**

We have been basically talking about clipping. So we covered point clipping and then we talked about line clipping. Today we are going to talk about polygon clipping. Here are the different kinds of polygons.

(Refer Slide Time: 1:21)



## Clipping

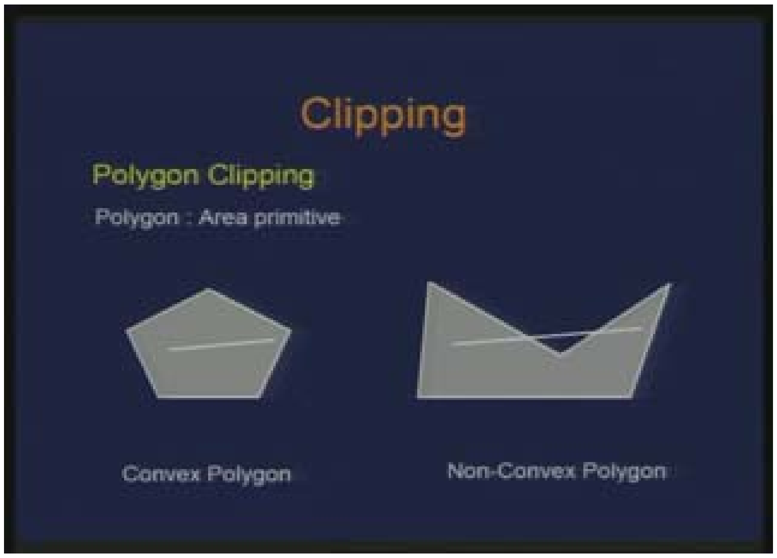
Polygon Clipping

Polygon : Area primitive

Simple Polygon: Planar set of ordered points
No line crossings
No holes

Basically we are looking at simple polygon. What I mean by simple polygon is it is basically a planar set of a ordered points $V_1$ $V_2$ $V_3$ up to $V_n$ and I do not want lines to cross for the polygon and also I do not want any hole in that polygon. This is what a simple polygon is. Here you see the example. So this is an example of a simple polygon. It could be convex, it could be non convex. So the restriction we are posing here is we should not have a line crossing, just the way it is happening here $V_1$ $V_2$ $V_3$ $V_4$ and there is a crossing of the line and there should not be any hole here. So given that I consider a simple polygon and the reason I am interested in a polygon because they are area primitives where I can define primitives filling area. One thing which we were talking about is that can we extend the notion of clipping for the line to the polygon. If you just consider the polygon to be defined as collection of lines each of the edge being aligned then all I need to do is process the line of the polygon one by one and do the line clipping. Let us see whether we can extend these ideas.

(Refer Slide Time: 3:30)



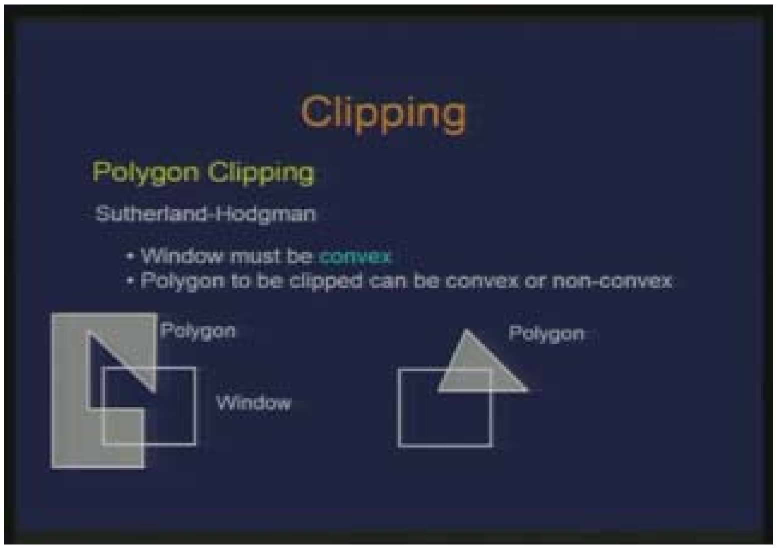
## Clipping

**Polygon Clipping**

Polygon : Area primitive

So just to further illustrate the difference between convex and non convex that we have earlier also seen there is this convex polygon where if I join two points within the region the line is inside that region that is what I mean by convex polygon. Whereas a non convex or concave sometimes referred as if I join two points which are inside the polygon the line joining these two points may not be inside the polygon. This is non convex. Now let us try to see what is the restriction for the time being we want to impose on the given polygon we are interested in to clip and the window against which we are going to perform this clipping. As far as the given polygon is concerned we are taking a simple polygon, convex or non convex whereas there is a restriction to the window which we want to use for clipping.

(Refer Slide Time: 4:52)



## Clipping

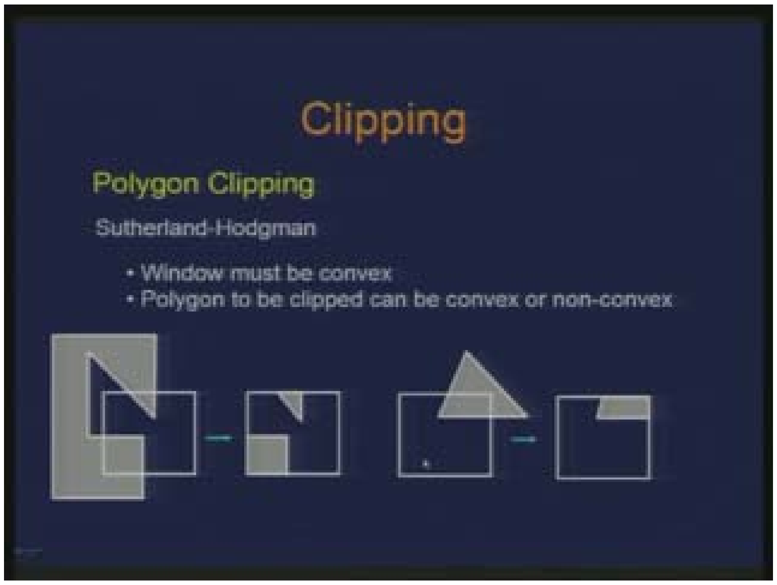
**Polygon Clipping**

Sutherland-Hodgman

- Window must be convex
- Polygon to be clipped can be convex or non-convex

So we would actually work on a window which is convex. So, we have this restriction then the window must be convex but the polygon could be convex or not. Here is the example, we have this window, just for simplicity I have consider here the window to be just a rectangular region which is convex and here you see a polygon which is non convex whereas here I have the polygon which is the triangle which is a convex problem. So what are we trying to do as a process of clipping is, we are interested in getting this portion of the polygon and this portion of the polygon inside the window and reject or discard the rest. Similarly I am interested in getting this part of the polygon to be declared inside the window and chop of the rest. That is what I mean by clipping polygon against this way.

(Refer Slide Time: 6:10)


## Clipping

### Polygon Clipping

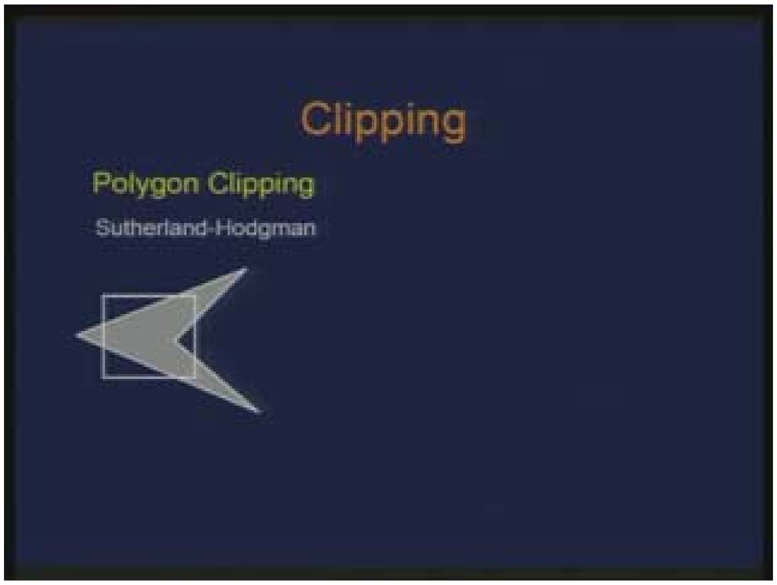
Sutherland-Hodgman

- Window must be convex
- Polygon to be clipped can be convex or non-convex

That is basically what is shown here. After the process of clipping I get this as the result for this polygon. Similarly for the second case where I considered triangle to be clipped the result is this. Now looking back to what we have done in the case of line clipping can you suggest an algorithm to do this?

Remember one thing, for instance Cyrus Beck Algorithm or Liang Barsky Algorithm basically got an assumption that it is a convex window. The non convexity of the window was sort of a special case which was also treated in a similar way so we always had the clipping against the convex region. So, what this convexity gives you is some sort of a notion of each edge of the convex window to be a sort of a clipper deciding whether it is inside or on one side of that edge or not. That is what I mean by saying a clipper is. May be we can apply a similar concept where the edge of the window acts like a clipper and then we perform that operation of clipping considering each edge of the window. Therefore each edge acts as a clipper.

(Refer Slide Time: 8:36)

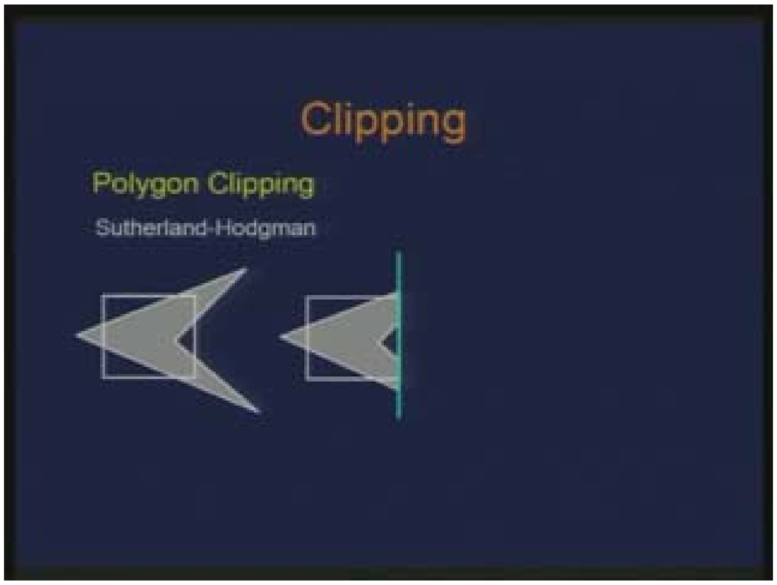

## Clipping

Polygon Clipping

Sutherland-Hodgman

I consider a non convex polygon to be clipped against this rectangular window. This is an example. Now what I do is I consider one of the edges of the window to be a clipper. Clipper means I apply the clipping against that edge and I have this notion defined to the edge that which is inside and which is outside in a very similar fashion as we have seen in the case of a line.

(Refer Slide Time: 9:13)

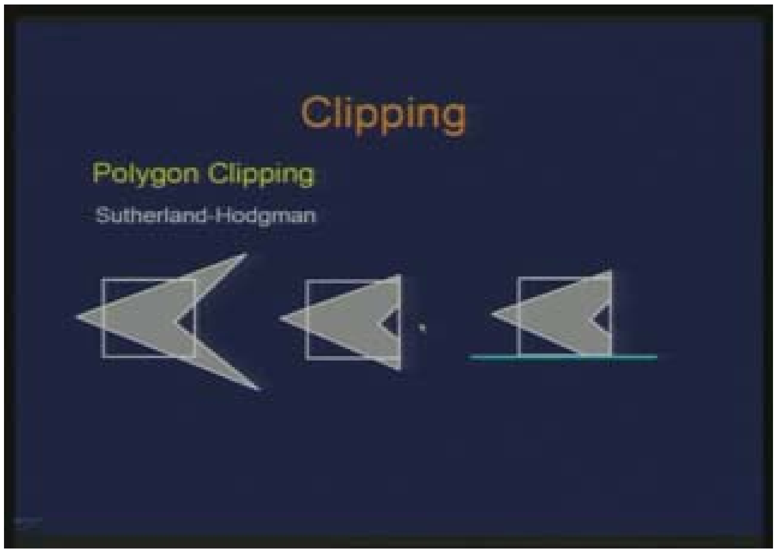

## Clipping

Polygon Clipping

Sutherland-Hodgman

So I define this clipper which is the right edge. Now what I do is I basically take this polygon use this as my clipper and discard whatever is outside this. So this portion is the outside portion of the clipper and this portion is the inside portion of the clipper. So basically I chop of, just slice it by this clipper the polygon. And this is what I get as a result. This is my resulting polygon after having clipped against this edge. Now I can apply this operation in succession for all edges from the window.

(Refer Slide Time: 10:19)



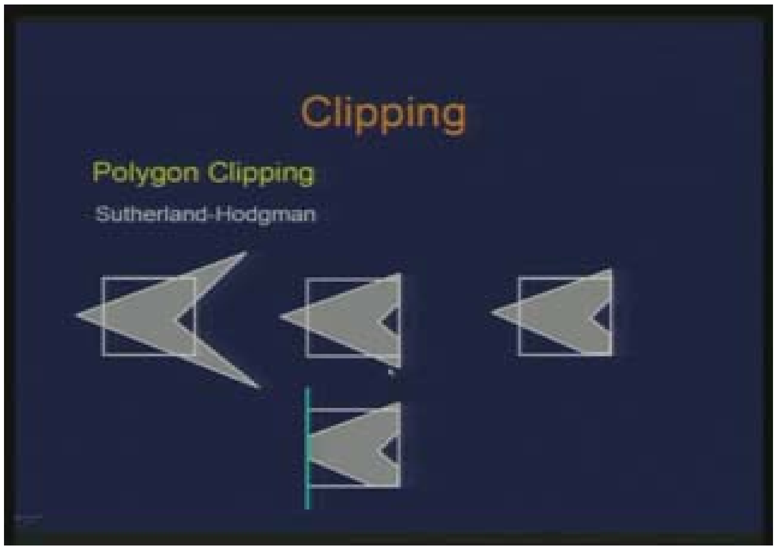
## Clipping

Polygon Clipping

Sutherland-Hodgman

I take that as an input whatever was given after the first clipper and apply against this edge now. So this chops of this part of the polygon and so on.

(Refer Slide Time: 10:36)



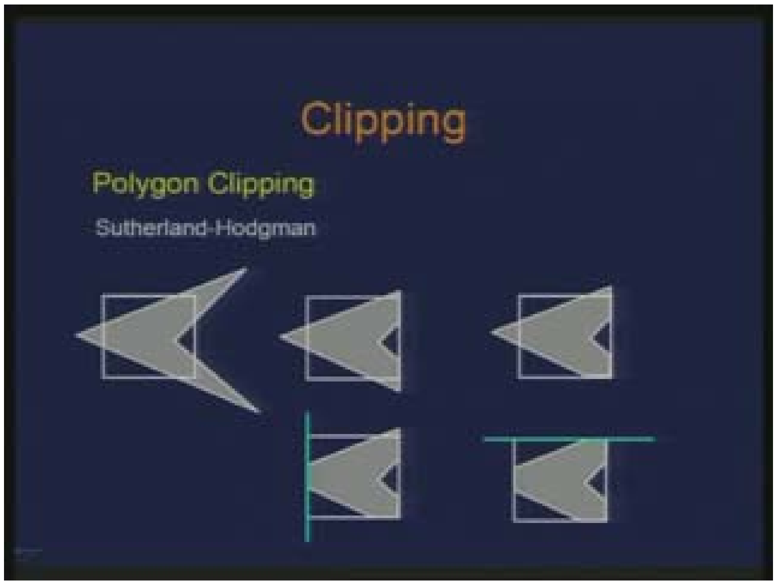
## Clipping

Polygon Clipping

Sutherland-Hodgman

That chops of the left part of the polygon and this is the top edge.

(Refer Slide Time: 10:43)



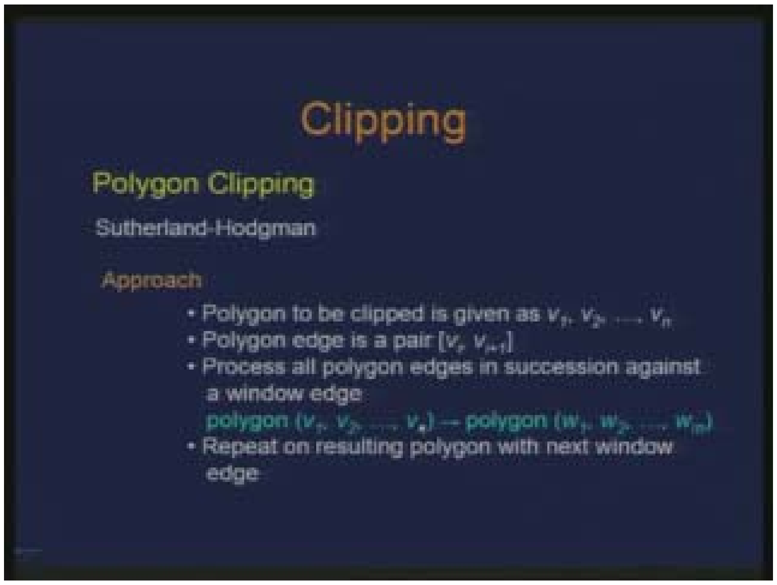
## Clipping

### Polygon Clipping

Sutherland-Hodgman

Considering each edge of the window as a clipper at the end of this full operation I get this polygon which is the clipped polygon that I need.

(Refer Slide Time: 11:11)



## Clipping

### Polygon Clipping

Sutherland-Hodgman

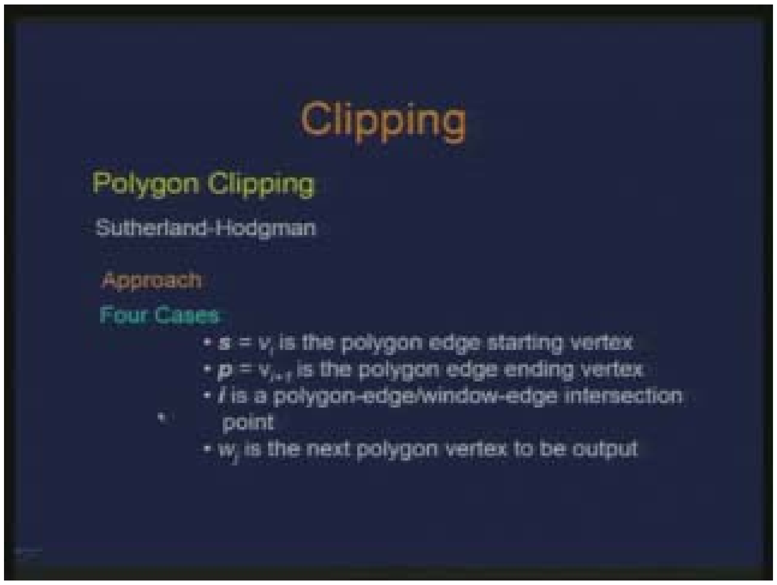
Approach

- Polygon to be clipped is given as $v_1, v_2, \ldots, v_n$
- Polygon edge is a pair $[v_i, v_{i+1}]$
- Process all polygon edges in succession against a window edge
  polygon $(v_1, v_2, \ldots, v_n) \rightarrow$ polygon $(w_1, w_2, \ldots, w_m)$
- Repeat on resulting polygon with next window edge

Approach: Here is what I am trying to do as an algorithm. I consider polygon to be clipped given as series of points, ordered points taken in one particular order as $V_1$ $V_2$ to $V_n$ and a polygon edge is basically a pair defined between $V_i$ $V_{i \text{ plus } 1}$ and since polygon is a closed figure I need to consider the last as to be $V_n$ $V_1$ so there is a rap around from this sequence I have as points.

Now process all the polygon edges in succession against a window edge. So what it does basically is, if I consider $V_1$ $V_2$…….$V_n$ as the input polygon and the respective pairs of these $V_i$'s defining the edges what I get is a new set of points defined by $W_1$ $W_2$ to $W_m$ which gives me the new polygon. Then I repeat this process against the next window edge and when I have exhausted all the window edges whatever I get is my resulting polygon. So what are we basically doing here is we are trying to output certain number of points as a process of clipping each edge of the polygon against a window edge. In fact what we are doing is basically create a new set of points which are referred here as $W_s$, they are just the new set of points. So, if we try to see, actually there happens four cases and before I talk about those four cases which would output 0. or 1. or 2.

(Refer Slide Time: 13:43)



## Clipping

Polygon Clipping

Sutherland-Hodgman

Approach

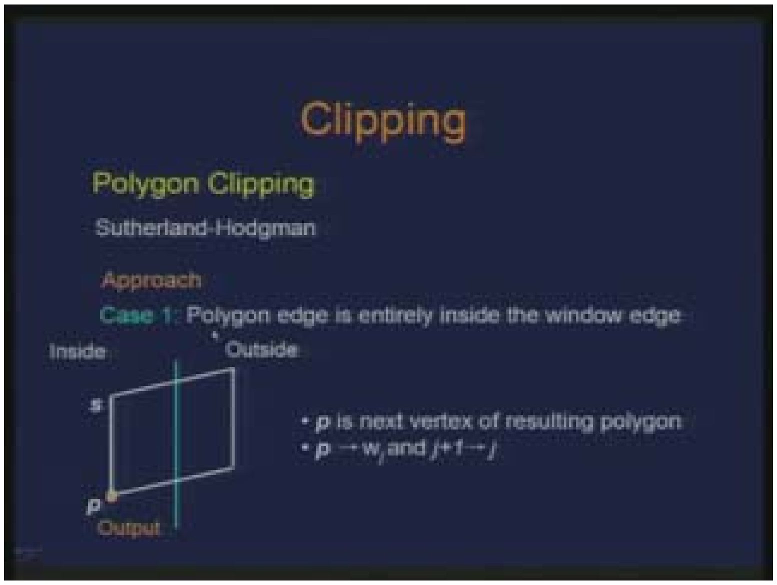
Four Cases

- $s = v_i$ is the polygon edge starting vertex
- $p = v_{i+1}$ is the polygon edge ending vertex
- $i$ is a polygon-edge/window-edge intersection point
- $w_j$ is the next polygon vertex to be output

After performing a clipping of an edge of the polygon against the window edge actually happened to recover by four cases. So, if I do a case analysis of those four cases I would know what points are going to be given as the output which will then be included in my series of points of $W_s$ to define a new polygon.

Therefore I consider s which is given as $V_i$ to be the starting vertex of the polygon edge. And this s could have been previously analyzed, this could have been analyzed in the previous iteration but for the current edge I am considering s as $V_i$. And p is the next point for the edge pair $_{Vi}$ plus 1 which is the ending point or ending vertex of the intersection. now I define i which may result from the polygon edge and window edge intersection and there could be an intersection of this polygon edge with the window edge and as a result I may get point of intersection which I define as i. So this i should not be confused with the subscript i here and this is a point of intersection. Then what I am going to have is this point $w_j$ which I will declare as the output from the case analysis I am going to perform. And at the end the collection of these Ws will form the polygon of my interest. Now let us try to see each of these four cases which would in turn give me either no output zero point or one output or two outputs that is two points for the output.

(Refer Slide Time: 16:22)


## Clipping

Polygon Clipping

Sutherland-Hodgman

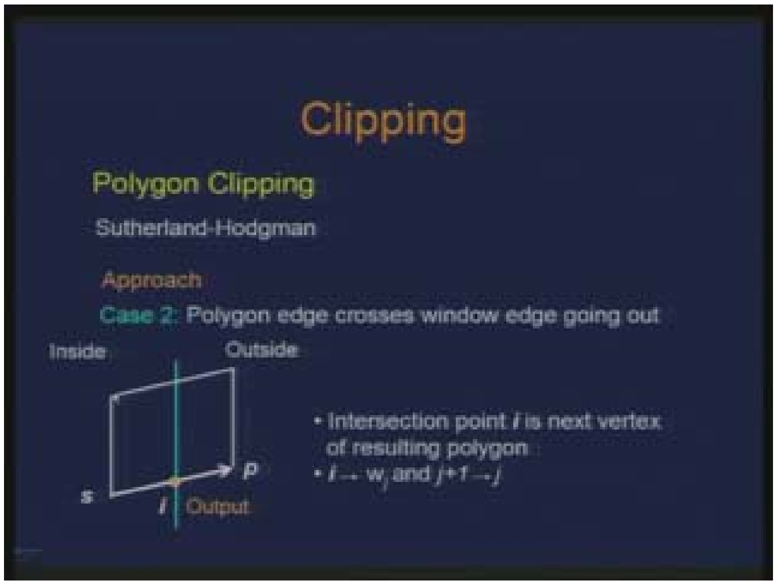
Approach

Case 1: Polygon edge is entirely inside the window edge

- $p$ is next vertex of resulting polygon
- $p \rightarrow w_j$ and $j+1 \rightarrow j$

Case one: the polygon is basically as a cyclic kind of a shape. It is a closed shape. Let us say I have a case where the polygon edge is entirely inside the window edge. So this is my window edge drawn in blue and I have this notion of what is inside and what is outside with respect to this window edge and now considering this edge which is given as S P that is my polygon edge in question, we are having S as the start point and P as the end point and this edge could actually be related to the rest of the polygon in some fashion. Now what we are going to look at is just analyze these two end points S and P for this particular polygon edge. So, if I observe that the polygon edge is entirely on inside on the side of the window edge which is the situation then I output P so P is the next vertex of the resulting polygon meaning I assign P to the $W_j$ and I increment j in the series of the points which I am going to use for defining my output polygon. In this case I have only one output given as P when both these points are inside the window edge.

(Refer Slide Time: 18:28)



## Clipping

Polygon Clipping

Sutherland-Hodgman

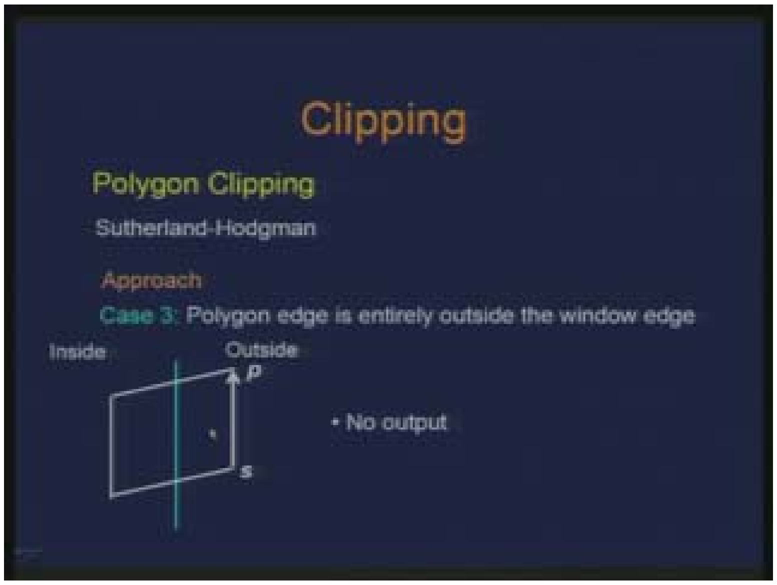
Approach

Case 2: Polygon edge crosses window edge going out

- Intersection point *i* is next vertex of resulting polygon
- $i \rightarrow w_j$ and $j+1 \rightarrow j$

The next case is, when I see that polygon edge crosses window edge going out that is going from inside to outside. So this is the scenario I have this edge S P which is going from inside to outside. So clearly when I cross the window edge it is going to have an intersection with the window edge i find out this point of intersection i and that is what my output is. I output i therefore I assign this i to the $W_j$ and I increment j. So in this case also I get one output for the point which is i.

(Refer Slide Time: 19:38)



## Clipping

Polygon Clipping

Sutherland-Hodgman

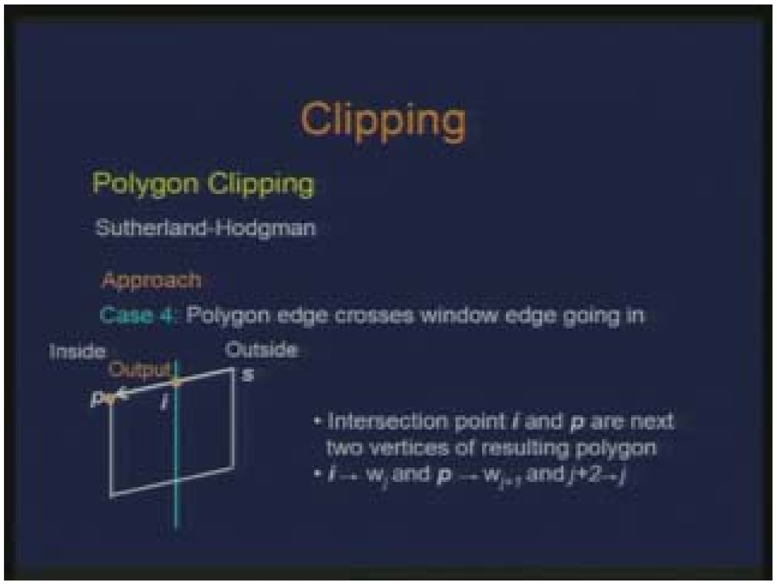
Approach

Case 3: Polygon edge is entirely outside the window edge

- No output

Now the next case is the polygon edge is entirely outside the window edge so S and P happen to be outside the window edge. Then I give no output that means no point is given as output for the polygon. So the last case left is this when the polygon edge crosses the window edge going in. That means I have this edge from outside to inside. That means S is outside the window edge and P is inside the window edge.

(Refer Slide Time: 20:07)



## Clipping

### Polygon Clipping

Sutherland-Hodgman

Approach

Case 4: Polygon edge crosses window edge going in

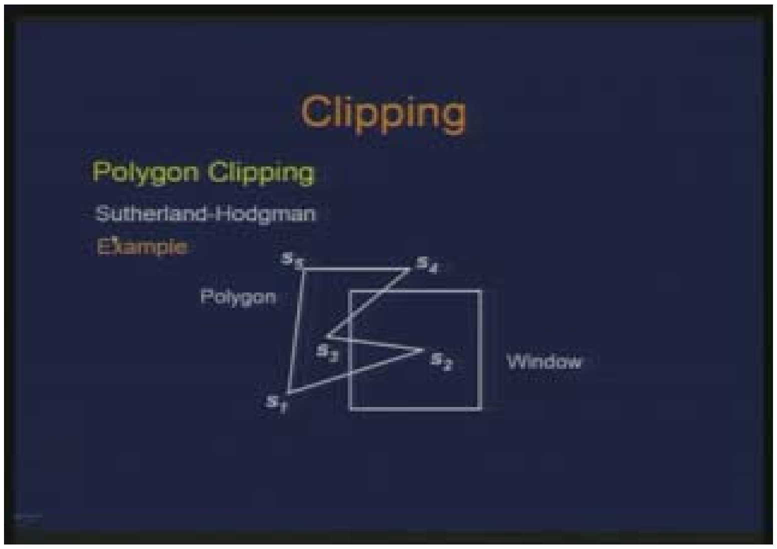
Inside
Outside
Output
p
i
s

- Intersection point *i* and *p* are next two vertices of resulting polygon
- $i \rightarrow w_j$ and $p \rightarrow w_{j+1}$ and $j+2 \rightarrow j$

Therefore clearly I have a point of intersection of this edge with the window edge which is given by i again. Then in this case I would output this i and P. These are two outputs now I have. I output i and P as the two vertices of the polygon. That basically means assigning i to $W_j$, P to $W_j$ plus 1 and j is incremented by 2.

Example: When I make this polygon as a collection of my vertices Wis then how do you I give the sequence of these points. I always have a definition of my polygon I go in an order basically. It is not just a collection of sorted points. If I go in an order of these points I get a polygon. This algorithm is called as Sutherland Hodgman algorithm where we consider the window to be convex and polygon to be any simple polygon.

(Refer Slide Time: 22:23)



## Clipping

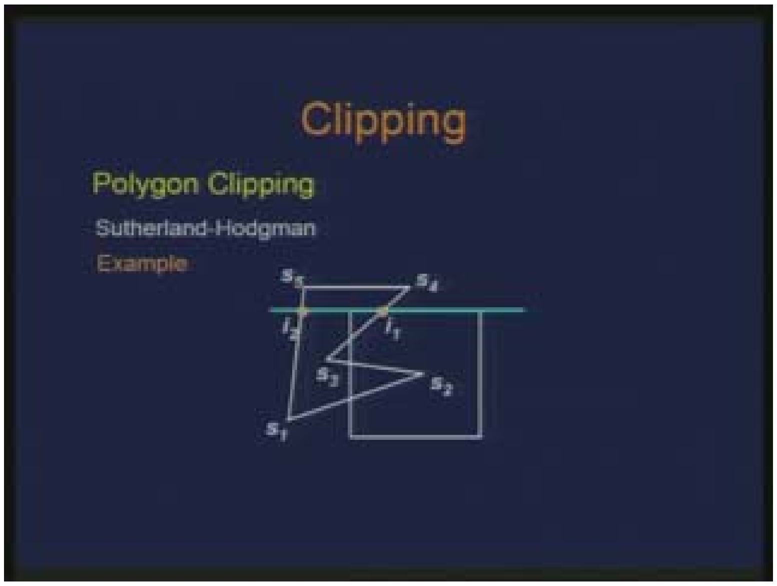
### Polygon Clipping

Sutherland-Hodgman

Example

Example: here is an example where I consider a non convex polygon $S_1$ $S_2$ $S_3$ $S_4$ $S_5$ these are the vertices of the polygon. And again I consider the window to be a rectangular region. Remember this has no restriction to be rectangular it could be any convex region because the way we are performing the operation of inside outside does not really restrict us to have a rectangular region as long as we have the region to be convex. Let us say $S_1$ is the point to start that is the first point I have in that definition of my polygon and all we are trying to do here is do this clipping for each of the window edges and we analyze these four cases and we output the points appropriately to be considered as the polygon for the next iteration. Given this as an input polygon let us see how to proceed.

(Refer Slide Time: 23:44)



## Clipping

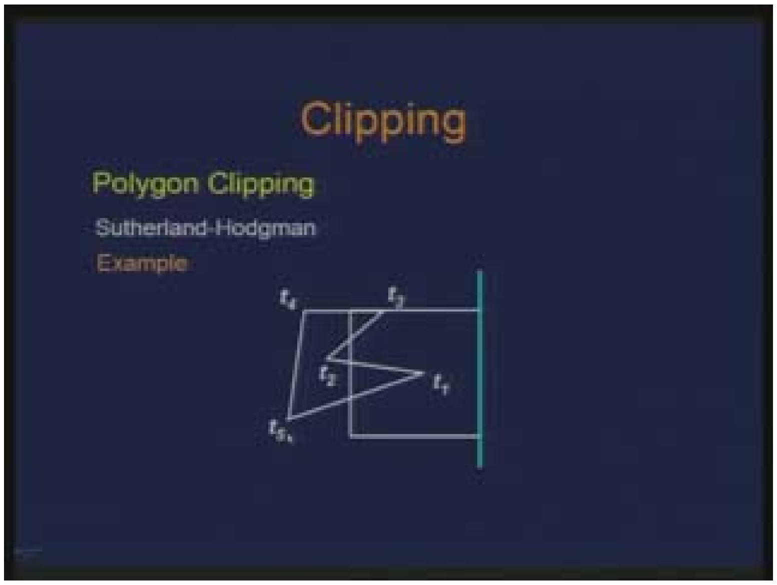
### Polygon Clipping

Sutherland-Hodgman

Example

This is my first clipper the top edge of the window, this is my first clipping. What will happen is that if I consider from $S_1$ to $S_2$ I go from outside to totally inside because I am not considering with respect to the window I am basically considering with respect to the edge. Therefore the outside inside operation is with respect to the window edge. I go from $S_1$ to $S_2$ which are both inside. So when I have inside to inside I output t which will be $S_2$ then I go from $S_2$ to $S_3$ again both happened to be inside the window edge then I output $S_3$, I go from $S_3$ to $S_4$ I go from inside to outside and I find a point of intersection which is $i_1$ so I output i1, when I go from $S_4$ to $S_5$ both happened to be outside the window edge I output nothing and then I go from $S_5$ to $S_1$ I find the point of intersection is $i_2$ so I output $i_2$ and $S_1$.

(Refer Slide Time: 25:23)



## Clipping

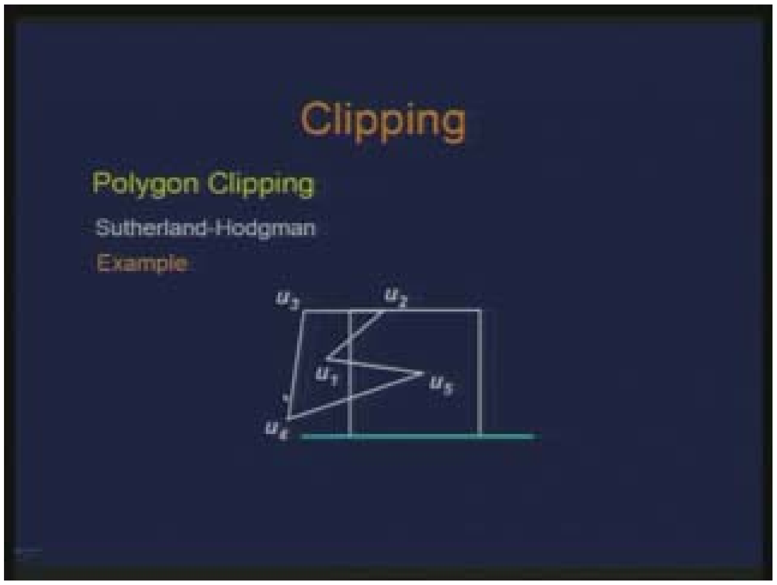
Polygon Clipping

Sutherland-Hodgman

Example

Therefore what I get as the resulting polygon is this. I just change the S to t now and basically change the suffixes. This becomes $t_1$ $t_2$ $t_3$ $t_4$ $t_5$. So this becomes my polygon for the next window edge to be considered. Now what happens is with respect to this window edge all the points are inside. So if I start from $t_1$ to $t_2$ I will output $t_2$, $t_2$ to $t_3$ I will output $t_3$ and so on.

(Refer Slide Time: 26:07)



## Clipping

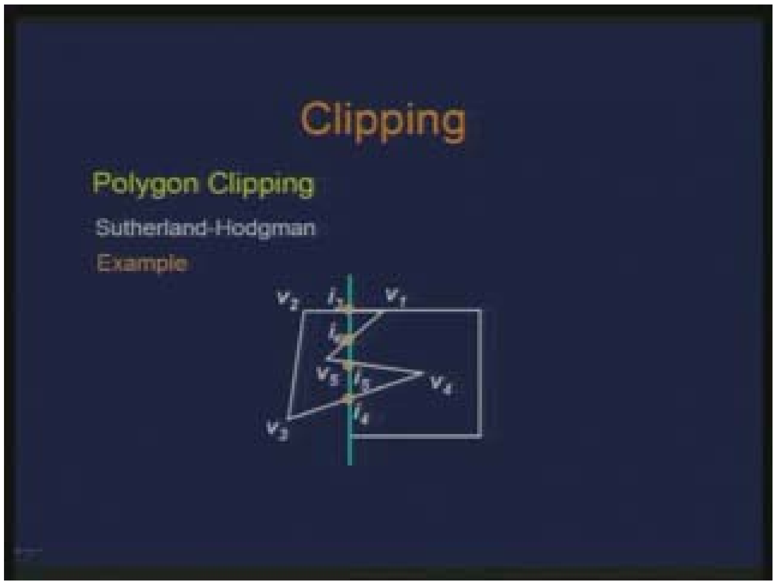
Polygon Clipping

Sutherland-Hodgman

Example

Therefore that is what I will get as a sequence $u_1$ so again I change from t to u and change the respective suffixes. Therefore it is $u_1$ to $u_2$, $u_3$ and $u_4$ and $u_5$. Again if I consider this window edge all these points are inside therefore nothing really changes with respect to these vertices except the numbering or enumeration that changes. So when I say $u_1$ to $u_2$ then $u_2$ will be given as the output so I will start from here.

(Refer Slide Time: 26:49)



## Clipping

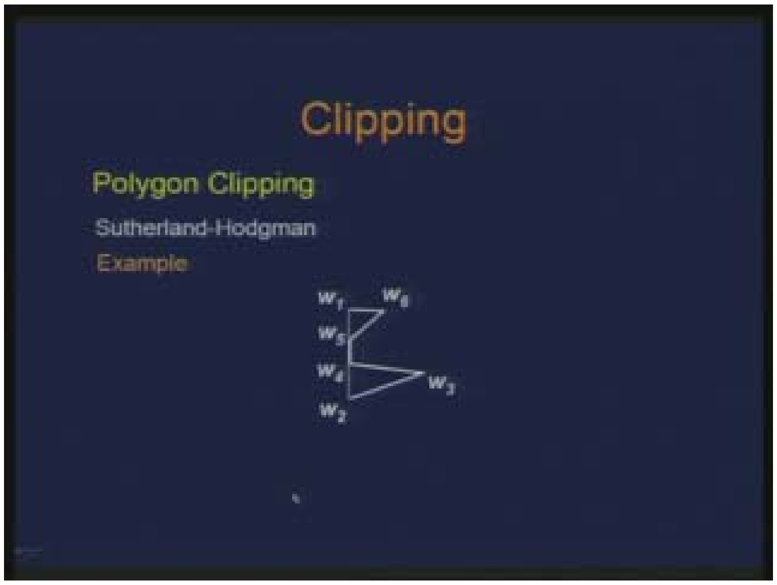
Polygon Clipping

Sutherland-Hodgman

Example

Now I get the output as $V_1$ $V_2$ $V_3$ $V_4$ $V_5$. Now this is my last window edge. I start from $V_1$ to $V_2$ which is from inside to outside, I find the point of intersection $i_3$ so when I go from inside to outside I basically output $i_3$, when I go from $V_2$ to $V_3$ both are outside I output nothing, I go from $V_3$ to $V_4$ which is from outside to inside I find this point of intersection as $i_4$ and I output $i_4$ and $V_4$ and when I go from $V_4$ to $V_5$ again I go from inside to outside, I find this point of intersection $i_5$ and I output $i_5$, when I go from $V_5$ to $V_1$ I go from outside to inside find the point of intersection as i6 and I output $i_6$ and $V_1$ and that completes the entire set of points. So, at the end of this I get my polygon as this $W_1$ $W_2$ $W_3$ $W_4$ $W_5$ $W_6$ and that is my resulting clipped polygon.

(Refer Slide Time: 28:22)


Clipping

Polygon Clipping

Sutherland-Hodgman

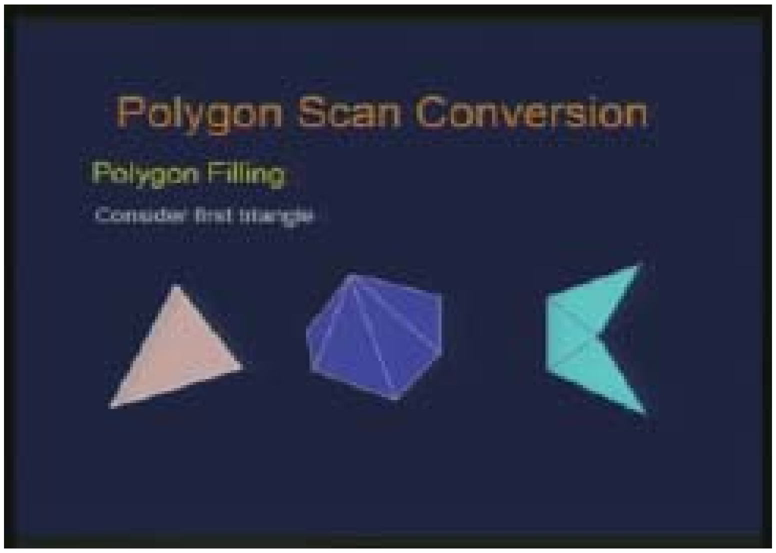
Example

The algorithm is fairly straight forward. And the idea which we actually used in the case of line clipping is being reused. Now one may say that this is not really a good polygon for various reasons because if I consider one single polygon where I have these two points laying on the same line it is sort of degenerative. Particularly it can cause problem when I do the shading of the polygon. So there is also a modification to this algorithm wherein instead of having output given as this as a single polygon sequence of W points they have the notion of fragmenting the polygons wherein actually you get this as one polygon and this as another polygon so there is a fragmentation. But that fragmentation actually requires much more book keeping in order being able to define these fragments right.

We will look at that algorithm when we talk about hidden surface elimination because that is in conjunction with hidden surface elimination and the algorithm is given by Weiler and Athreton to modify this Sutherland and Hodgman algorithm where you do not have degenerate polygons.

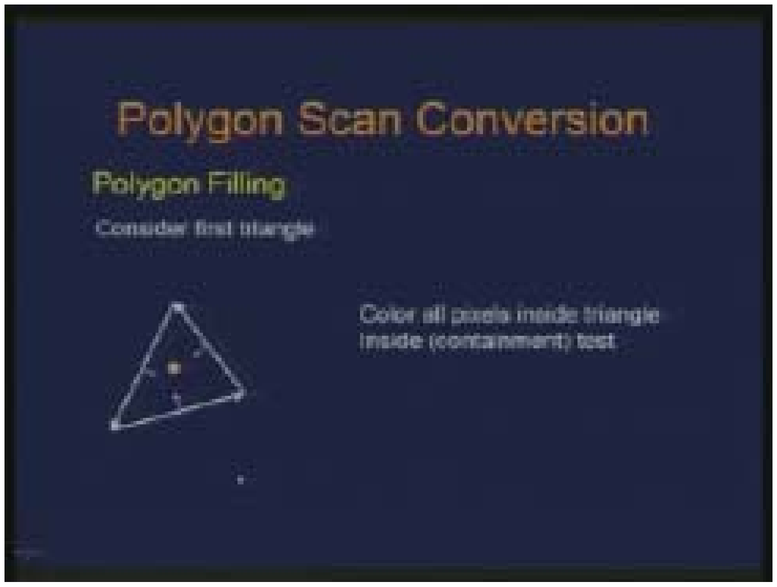
Now that since we have been talking about polygon let us try to see how we draw these polygons. We have basically looked at drawing algorithms for lines, drawing algorithm for circle, ellipse but we have not really looked at drawing of a polygon. one may say that polygon drawing is basically a line drawing because if I just define a polygon to be just a set of connected points through these edges then all I am saying is that consider each edge and do a plot of each edge using line drawing. But I then do not get the feeling of that polygon to be an area because I need to fill that with some color or with some other display attributes. So you would like to see some sort of a drawing algorithm where I can fill these polygons with some color. Just a collection of line drawing doing the polygon edges for drawing is actually also known as some sort of a wire frame display of the polygon. Here we are trying to address the polygon display using filling. This is known as the scan conversion of the polygon. Each of the pixels of the polygon is basically getting converted into some value.

(Refer Slide Time: 32:32)



Polygon scan conversion is basically for drawing polygons using filling. If I consider a simplest polygon which is triangular, here you see the examples of filled polygons, this is a filled triangle. Now if I am able to design an algorithm for filling a triangle then I can actually use that even for that other polygons because I can decompose a polygon into set of triangles irrespective whether the polygon is convex or non convex as shown here then I can just replicate my triangle filling algorithm if I am interested only in the filling algorithm for a triangle.

(Refer Slide Time: 33:37)



## Polygon Scan Conversion

Polygon Filling

Consider first triangle

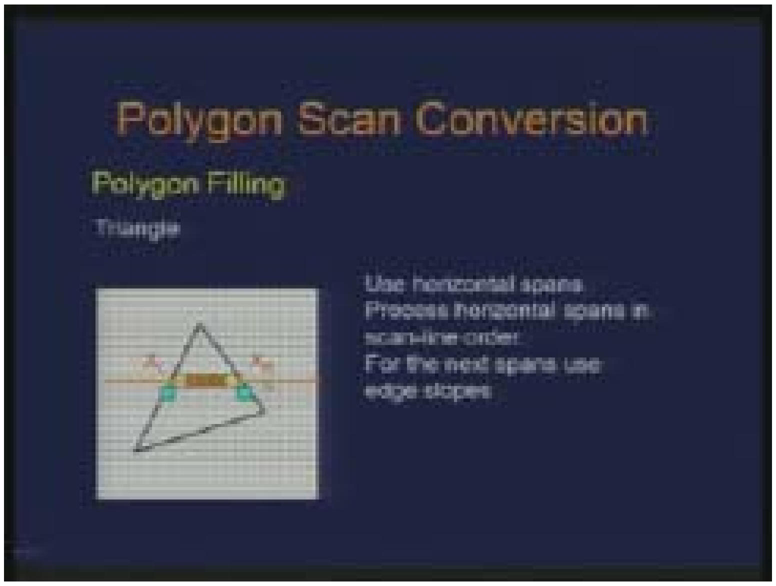
Color all pixels inside triangle

Inside (containment) test

Here is the pixel which is inside the triangle and if I figure out that this pixel is inside then I can assign the desired attributes of color or any other display attribute to this pixel. So the question I am basically asking is, whether this pixel is inside that triangle or not? In other words it is a matter of doing a containment test of points or the pixels. So whatever is contained in the triangle I just give the necessary attributes of this plane.

Now the question of whether a point is inside or outside a triangle can be well taken care of by what we have seen earlier. If I define the normal to these edges of the triangle then it is a matter of inside outside of these edges and if I get the answer that with respect to all the edges it turns out to be inside then I know that the point is inside the triangle. Therefore I can give the color to that pixel and if I find that point it is outside the triangle I just discard it. I cannot start with all pixels in this screen. You basically bound your area by considering the minmax of the points of the triangle so create a virtual window around it and do the scan conversion of that window.

(Refer Slide Time: 36:06)


## Polygon Scan Conversion

Polygon Filling

Triangle

Use horizontal spans

Process horizontal spans in scan-line order

For the next spans use edge slopes

Basically we are looking at some sort of a extend which is defined around that triangle and we sort of also exploit the fact that scans are horizontal so we go one scan line by one scan line. We sort of exploit this property which is in some sense a coherence along the scan line because if I find that the two extremes of that scan line happen to be inside the triangle then all the points which are spanned with that scan line can be colored. in turn it means that if I find out an extent here which I define as xl for a particular scan line, here when I refer to scan line it is a basically a horizontal line with respect to the window which I am trying to scan convert around this triangle.

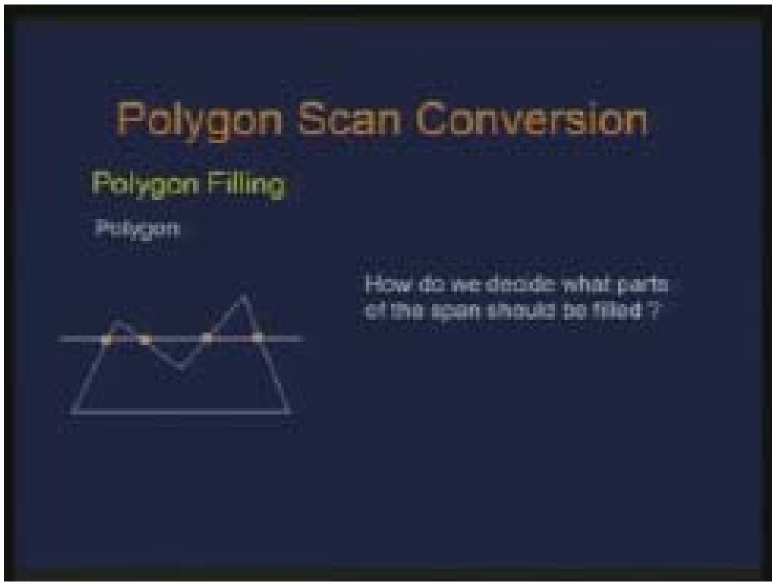
So, for this particular scan line I find out these xl and xr which could be just the intersection of this scan line with respect to this edge and this edge and then once I find out this xl and xr then all the points here can be colored. This is particularly useful when I just need to assign one color to all the points.

Another thing is that I can cover one scan line like this. So the next question is I go to the next scan line, next scan line means I go vertically down. From here I come to this scan line. One may say that find out the point of intersection explicitly with respect to this edge and the other edge and do the same thing but that could turn out to be an expensive thing. Therefore what we can do is we can exploit the knowledge of the edge slopes. The edge slope can actually give me the next point here. Remember we have actually used that knowledge while doing line drawing Bresenham’s or midpoint algorithm.

Therefore I actually get the next point for the next scan line using the slope of this edge. Similarly I can get this point using the slope of this edge. And once I have these two extremes the rest is done. The only thing is that this particular point I will have to change the edge. Here I was considering this edge, now it is basically between this edge and this edge. But that is also simple. All I am doing is going scan by scan finding out the two extremes in the horizontal span and fill them and for the next horizontal span I use the information from edge slopes.

How do we get the first scan line? That is the initial computation you need to do. The best thing is, you get a window around this which is like finding out the minmax of this area which gives you the window so you start at the top of the window. One thing for the general polygon is to use this triangle filling algorithm by decomposing that polygon into various triangles and apply to each of those triangles. But can we design an algorithm for a general or simple polygon or convex or non convex? The question which we are trying to answer is always the containment.

(Refer Slide Time: 41:29)



## Polygon Scan Conversion

### Polygon Filling

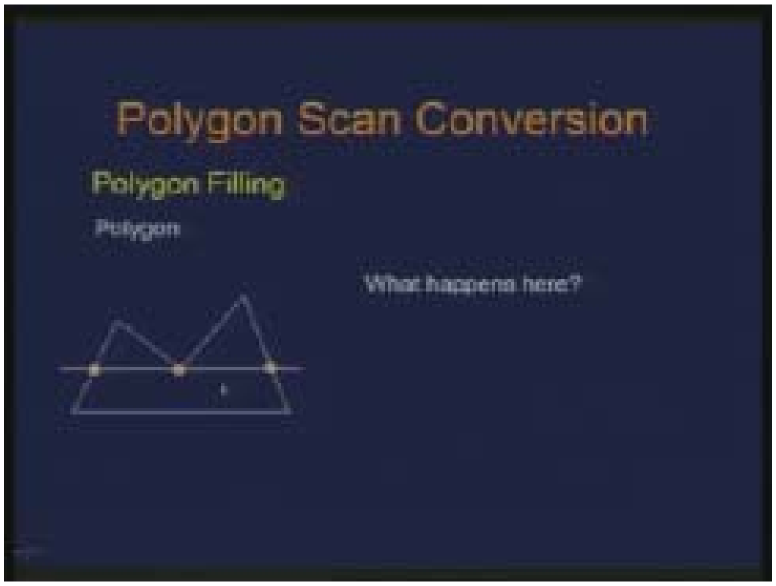
Polygon

How do we decide what parts of the span should be filled ?

Let me pose the problem here: Now I have this polygon which is a non convex polygon and this is the current scan line in question and all I am trying to answer is what are the pixels intercepted by this scan line or inside this polygon. The moment I answer that question I have my filling algorithm. Now given this scenario can I devise some algorithm saying that what are the points which are inside the polygon and what are the points outside the polygon? Let us say I am able to find these points of intersection. These points of intersection of the scan line with the edges are known to me.

Therefore what we observe here is that this part is to be filled, this part is not to be filled, and this part is to be filled. Basically it is actually a parity check. What we are saying is that, if I find the segments as we usually call them, this is one segment, this is another segment, this is another segment and so on. So if I do a parity check then all I am saying is that if the segment is odd fill it, if the segment is even do not fill it. This principle even holds when I have a convex filling because I am going to get only one that is odd.

(Refer Slide Time: 44:26)


## Polygon Scan Conversion

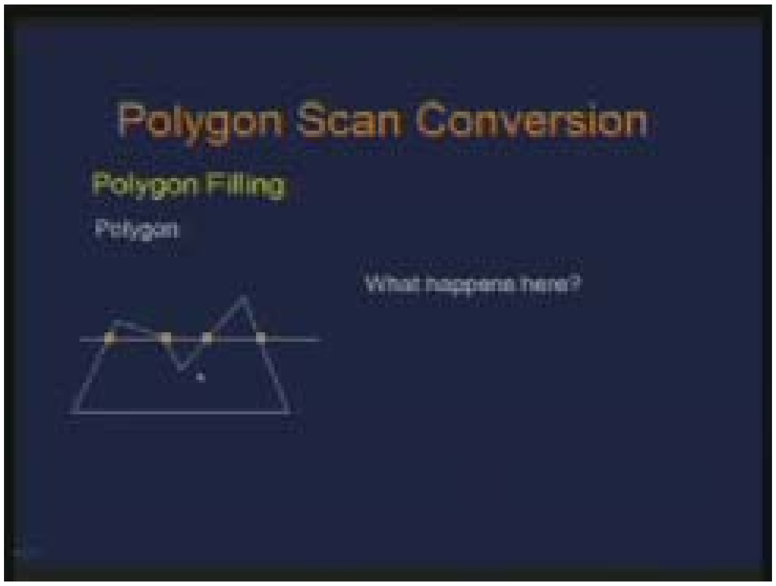
Polygon Filling

Polygon

What happens here?

But does it work always? What happens here? If I just apply the same parity check then I am going to display only this part and not this.

(Refer Slide Time: 44:58)


## Polygon Scan Conversion

Polygon Filling

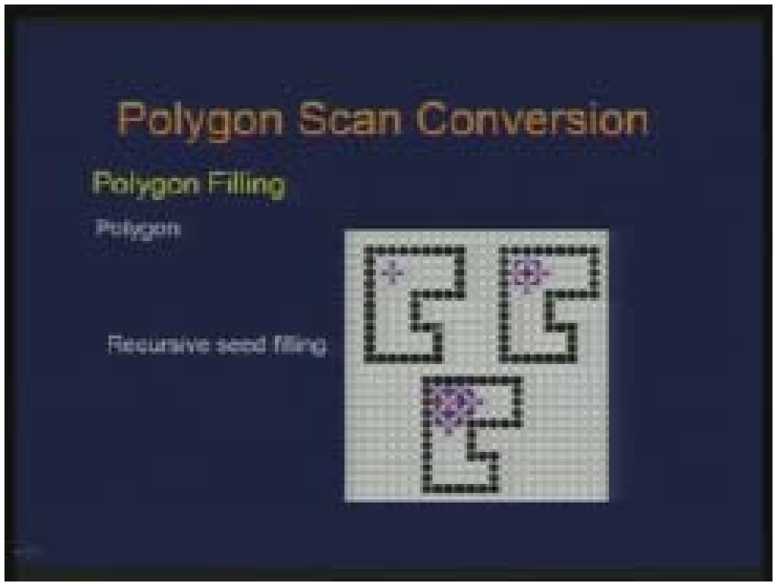
Polygon

What happens here?

Now what happens here? I go back to this and I actually give you a modification to the algorithm that in such situations you duplicate this point. Duplicate this point means I have basically virtually added a segment there. Then what will happen is this will become one, this is second right there but I do not draw that and this becomes third and I draw that. So what I am suggesting is that whenever you have an intersection with a vertex for the polygon you duplicate it. But the question is, does it work always? What happens here? This is duplicated I do not plot at this point but I plot this, this is not inside, it does not work. There has to be some topological issue here. It is not just that I duplicate the vertex. So, one way I can look at is that if the vertex in question is either a minimum or a maximum vertex of the incident edges on that point then I duplicate. Here this vertex is a minimum vertex, minimum in terms of when I say y of that is minimum with respect to the two instant edges at this point that is this and this. So here I duplicate.

Here what happens is that there are this edge and this edge so for this case it is the minimum and for this case it is the maximum therefore I do not duplicate it. So when it intersects to a vertex of the polygon I need to do this extra checking. But the rest of it is just the same. I just label these segments to be drawn or not to be drawn. Now there is another way to look at the filling problem. What are we trying to do basically when we fill the polygon is, we basically look at the pixels which are inside a polygon and if I can in some way grow that point using the neighborhood information around that point and ask that question to the neighbors and propagate that then also I will do the filling. The propagation is basically constrained by the boundary edges of the region.

(Refer Slide Time: 49:07)



Polygon Scan Conversion

Polygon Filling

Polygon

Recursive seed filling

Therefore here is an example where I have this polygon and the black points are showing the boundary of the polygon. I get this seed which is plotted here as the pink point. So once I establish that this point is inside then I can consider the neighborhood of this point so neighborhood could be in different ways, here I am talking about the left and the neighbor the bottom and the top neighbor or the low and the up neighbor. Therefore you call this as four neighbors of the point and you can also consider eight neighbors if you consider the diagonal pixels around. So all I am trying to do is get these neighbors and ask this question whether it is inside or outside and then in turn get the neighbors for these points and just keep growing. In image processing you must have seen a similar thing.

It is a pixel aggregation or a region growing which is used for segmentation because there what we are doing is this boundary is actually given by certain features of the image or the segment and within that boundary everything is homogenous. So those pixel attributes within that segment are very similar. All I am trying to do there is find out the connected pixel to the seed and keep growing that region till I hit the boundary. Again we can actually use certain information from the pixel itself.

For instance, if I find out this pixel is inside the explicit check which I will do for the inside outside I may actually restrict to only part of the boundaries not all the edges of the boundary. So I can actually add more information to the point knowing that where this point is. So computationally it may be more costly than what we have done earlier. But this shows you a different way of thinking. You are actually doing a filling but in a different thought.