Computer Aided Design

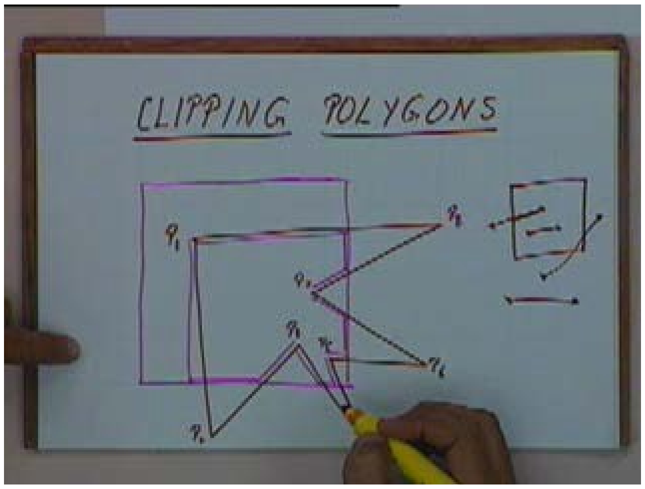
Prof. Anoop Chalwa
Department of Mechanical Engineering
Indian Institute of Technology, Delhi

Lecture No. # 07

Clipping of Polygons

Today we will be talking of algorithms for clipping polygons. In the last class we had seen methods of clipping straight lines and curves as a sequence of straight lines.

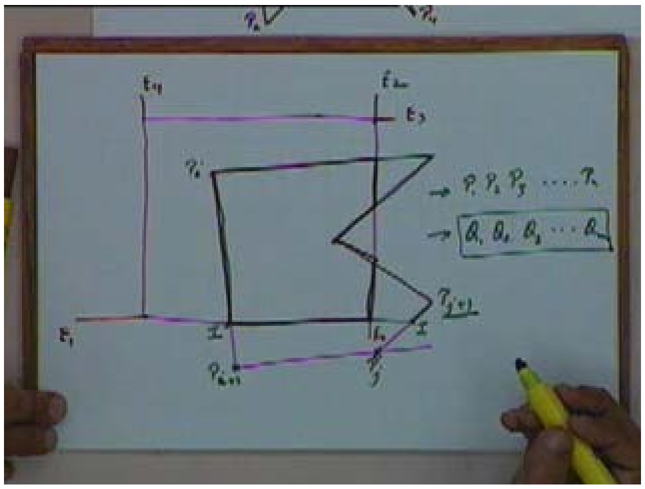
(Refer Slide Time: 00:01:15 min)



We had seen that if we have a clipping window and a straight line crossing that window. How we can clip it at the clipping edge and get the portion which is inside the window? The line could be either like this or like this or totally inside or total invisible. We had seen different cases and we had seen methods of deciding what portion the lines lie inside the polygon and what portion lies outside. Today we will see a related problem that is how to clip a polygon.
Now a polygon is a sequence of edges but we cannot directly use the same method because if we use the same method, this polygon will be clipped into a set of line segments which would be this line, this line segment, this portion, this portion and so on. It won't be clipped into a closed region. Initially the polygon was a closed region once we do the clipping, we still want to retain that this is a polygon and we want to retain this as a closed region. The reason is simple that if we want to do any polygon filling or if this polygon is a solid area, by solid area I mean a region which is filled inside either by hatching or by all the pixels inside being by polygon filling algorithms. If it is solid area or we want to process this polygon later for further clipping or for other applications. We want to retain this as a closed area. If, it needed to be a closed area then finally this polygon should be clipped into a sequence of edges including these edges only then it will become a closed polygon.

For a clipping a polygon and then finally getting a closed polygon from it. We will be, basic technique would be if this is a clipping window and we have a polygon crossing it like this. As I said finally we want this clipped polygon which is. Okay. We clip this polygon first with respect to one edge I think this is edge one then with respect to the second edge then with respect to the third edge and so on. We clip it with respect to each of these four edges separately.

(Refer Slide Time: 00:03:25 min)

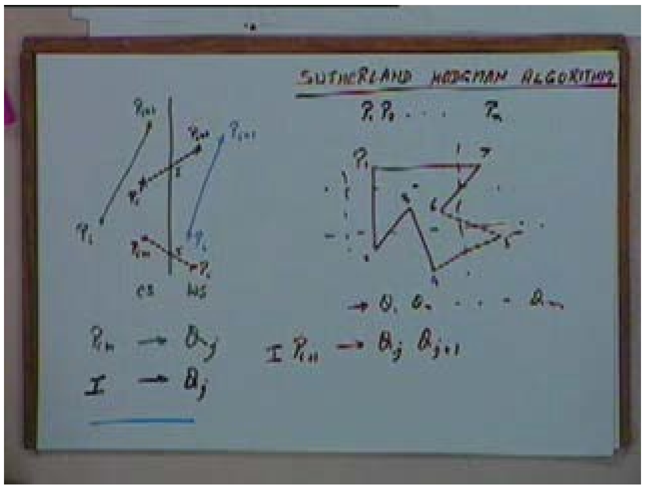


When you clip it with respect to the first edge E_1, we will get a polygon which would look like this. Then we will clip it with respect to the edge E_2 and this would be clipped on to this portion and then we will clip with respect to edge E_3 and E_4 and when you are clipping with respect to any single edge, you have to keep in mind when initially we input a sequence of vertices like this and the output should also be a sequence of vertices something like this. So, a sequence of vertices would represent that this is a closed region or a closed polygon. The input was a closed polygon, the output should also be a closed polygon that is a prime thing that has to be kept in mind. The way you do it is if you have any pair of points let's say P_i and P_{i+1} where one point is inside the clipping region, the other point is outside. Then the intersection of the line P_i P_{i+1} with respect to the clipping edge has to be included in the output list. So this point which is the intersection I, this will be included in the output list.

On the other hand if you have a case like this where this is let's say P_j and this is P_{j+1}. Mind you right now I am clipping with respect to the clipping edge E_1, so P_j is outside and P_{j+1} is inside. In this case this intersection point I has to be included in the output list and P_{j+1} has to be included in the output because P_{j+1} will also be inside the clipping region because the clipping edge is they are even. While in this case P_i P_{i+1} where P_i is inside and P_{i+1} is outside, P_{i+1} will not be included in the output list, only P_i and the intersection point I would be included. On the other hand if you have a case like this, this point and this point, both the points are outside. In that case we don't have to bother, neither of these two points would be included in the output list. If we have both the points on the correct side or within the clipping region, for example in this case P_i

and this intersection point I when the clipping edge is via E_2 then obviously both these points will be retained in the output list. Is that okay? So corresponding to that we will get four cases.

(Refer Slide Time: 00:08:11 min)

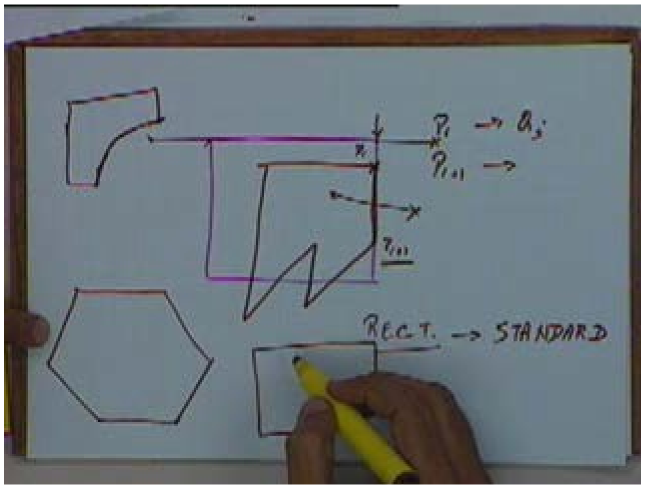


Let's say this is the clipping edge and this let's say the correct side of the clipping edge and this is the wrong side of the clipping edge. We have a P_i and P_{i+1}, this is one simple case. The second case is when both points are on the wrong side of the clipping edge. One case is a case like this and the fourth case is a case like this. In both the cases this is the intersection point and this is the intersection point. My input was P_1 P_2 till P_n and we will start from the point P_1 and keep processing each of these points one by one.

Let's say we have already decided about the point P_i and we want to decide about the point P_{i+1}. In the first case P_{i+1} would be included in the output list Q, the appropriate number will be included as let's say Q_j. In this case P_i is already included in the output list, so the intersection point I would be included as Q_j. Okay. In this case where P_i and P_{i+1} both are on the wrong side, no action would be taken and point P_{i+1} will be ignored and in this case we have already processed the point P_i which was on the wrong side so that time you have ignored the point P_i but now I as well as P_{i+1} both these points have to be included in the output list. So I and P_{i+1} would be included as Q_j and Q_{j+1}. So this is say, if you have a polygon we will start with the first point P_1, go through all the points, clip this with respect to one clipping edge and get a list of points Q_1 Q_2 so on till Q_m. then take this list, clip it with respect to second clipping edge, get another output then clip that with respect to the third clipping edge and then with respect to the fourth clipping edge. This process will be repeated four times once with respect to each edge and this algorithm is referred to as Sutherland Hodgman algorithm. Any question with respect to this Sutherland Hodgman algorithm?

In all the clipping algorithms one very important thing is finding out this intersection point. It seems to be a very trivial exercise. You don't want the intersection between this straight lines but maybe we can take it as an exercise and try to write a code which will find out the intersection point between two straight lines.it is not as trivial as it sounds. Well you we have to take care of cases when the two lines are parallel. Both intersection point are in decimals [Conversation between Student and Professor - Not audible (Refer Slide Time: 00:13:34 min))] Again. Yeah, we will draw. See the effect of that is not going to be critical as well. As I told this line is going to be displayed as a sequence of pixels, so you want to cut the line only at a pixel which is displayed. So whatever the intersection point you find out, you round it off that would be a point to be displayed, so there will be no problem. Yeah question from that side. Student: It is more being parallel because if it is parallel then one point cannot be inside and the other point outside. Think of a case like this.

(Refer Slide Time: 00:14:22 min)



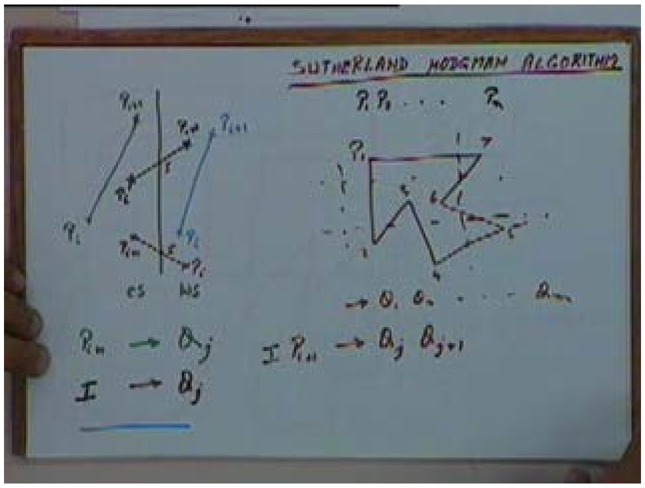
This is your clipping window and your clipping polygon is something like this. What happens in a case like this? This is your clipping polygon. Yeah, but when you are trying to find out the intersection point at that time you shouldn't get overflows. You shouldn't get a division by zero, if you trying to find out the intersection point between these two lines. Between, this as the clipping edge and this as the edge. Sir, in this case also you couldn't find the intersection. How do you decide that you will find the intersection or not, even at this point which is your point P_i. Now you got a point P_{i+1}. How do you decide whether you want to find the intersection or not find the intersection? One is inside one is outside. P_i you have already processed and you have included P_i in the output list, now you got a point P_{i+1}.
You know the coordinates of P_i and you know the coordinates of P_{i+1}. Now you have to decide whether to find the intersection or not to find the intersection. So you look at P_{i+1}, P_i you have already included in the output list which is let's say Q_j. Now P_{i+1} you have to decide whether this has to be included or not. Now this point what you are saying is it will be always treated as a point inside and you will not try to find the intersection at all. That might not happen. In some cases you might get at this point is just outside because of errors in deciding out, the difference is going to be very small. See in the worst case it is at the line you mean that case it will be decided as inside. If it is just outside the line. Student: it maybe integers the points are even in integers.

We can have simply a check for it or separate check for it. You have to have separate check for it. No. both have same x coordinates and coincides with one of the clipping edge keep the intensities. What kind of check that you can think of but you have to have some kind of a check so that such cases are taken care of. In fact in all these problems whether it is clipping or whether it is an even polygon filling or any algorithm, you will find that such cases always give a lot of problem. You have to take care of such let's say seemingly trivial cases and that is what makes the code sometimes quite complex. In fact if you try to write a clipping algorithm, you will find situation in such cases come up and we have to explicitly take care of such situations.

In fact even in the line clipping, simple line clipping case where with respect to this clipping window you want to clip this line. What happens if this line is like this and you want to clip it against let's say this edge that's our top edge. If you try to find out the intersection points, you will get absurd results and mind you intersection point has to be found out efficiently. Efficiency is of time concern, so there are number of methods for finding out the intersection points which are available. What I suggest is you can go through the book, you will find some method for finding out the intersection points. Okay. But in all those method, you have to be carefully about such cases. If you try to find out the intersection point of this edge or this line with a top clipping edge, as I said you can possibly get integer overflows division error. So all such cases have to be taken care of. Sir (Refer Slide Time: 00:18:58 min, not audible). I am just coming to that.

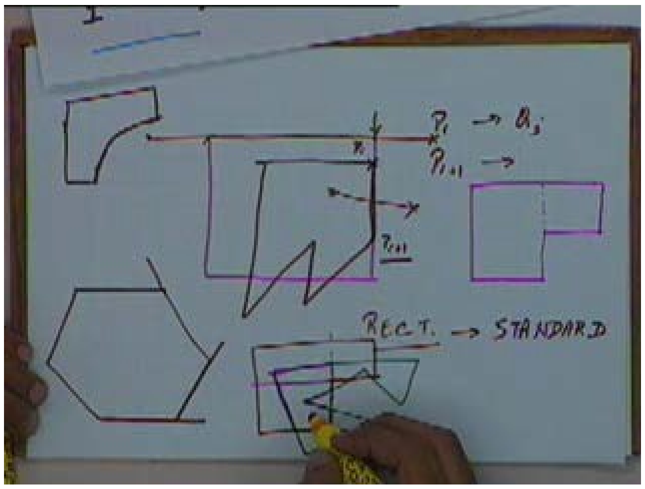
So far we have been talking of rectangular windows. Now a window need not always be rectangular, whenever we have a rectangular clipping window it is referred to as a standard clipping window. A clipping window can be of any arbitrary shape. Let's say my clipping window can be a polygon like this or it can be even a concave polygon like this. The algorithm that we have discussed so far, we have been talking of a standard clipping window. Strictly speaking yes but that will make the clipping much more complex. Typically a curved edges can always be split up as a sequence of edges. So even if we have a curve edge in a clipping window like this, we can talk of it as a polygonal clipping window. Now this is a convex clipping window, this is a concave clipping window. The (Refer Slide Time: 00:20:20 min) algorithm which are suitable for convex boundaries only.

(Refer Slide Time: 00:20:30 min)



For example Sutherland Hodgman algorithm, we are clipping this polygon first with respect to one edge then with respect to the second and so on.

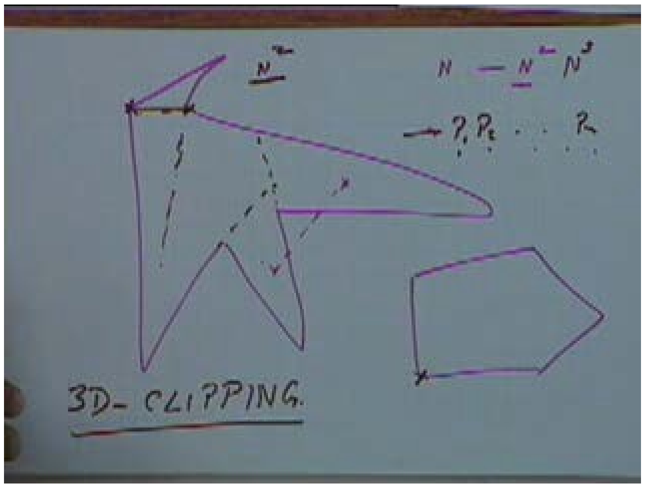
(Refer Slide Time: 00:20:43 min)



If we follow this algorithm for a convex clipping polygon like this, the first case with respect to this edge then with respect to this edge then with respect to this edge and so on. We will still get the same result, we will still get the correct result but if we follow with respect to this concave clipping volume, clipping area we will not get correct results. The reason is quit straight forward. Let’s say if I have a polygon like this and I first clip with respect to this edge and I will take a portion which would look something like this, this way, this way.

Then we clip with respect to this edge. When I am clipping with respect to this edge, the polygon will get changed to this. Then I clip with respect to this edge and my polygon will get changed to this. Now obviously this one is the correct result. So if you have any concavity in a clipping region then we need different algorithms or different ways of handling this. So if we have a convex clipping algorithm oh sorry, a convex clipping region, the Sutherland Hodgman algorithm is good enough but if we concave clipping areas then we need to modify this algorithm or we need some other ways of handling the clipping. We don't go into details of a concave clipping area, I will just very briefly tell you how it can be handled, the details maybe you can look up elsewhere. let's say if we have a concave clipping area like this, what we have to do is we will have to split it up into a set of convex areas and set of convex areas would be something like this.
So we will get one clipping area like this and the second clipping area like this. Then we will clip the polygon with respect to both the areas separately and the result of clipping this polygon will not be one polygon but two separate polygon, one which is inside this polygon and other which is inside this polygon. So, if we have any concave clipping area, we will have to split that into a set of convex areas. Again you can think of algorithms for doing that. How do we decide this concave? How do decide whether it is concave? How do you define a concave area? Student: sir how do you decide it for a machine. What is the algorithm because for determining its concavity. For determining the concavity.

(Refer Slide Time: 00:24:04 min)



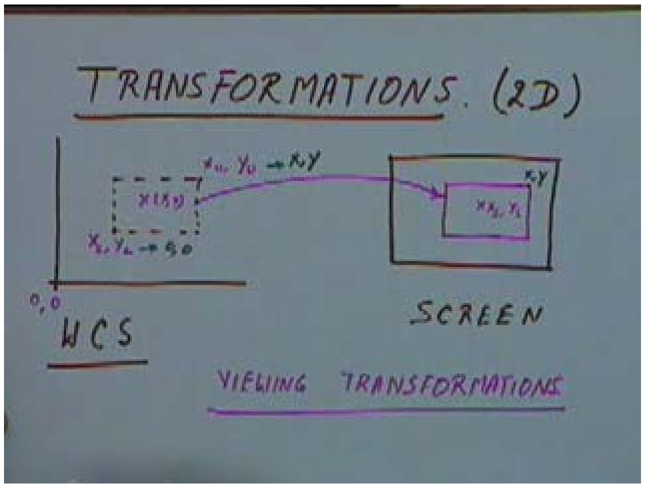
If you have any polygon like this or you have a polygon like this, you want to decide whether the particular polygon is convex or concave. How do you define a concavity, anyone? Student: always greater than… degree. One at a time, one at a time please. (Refer Slide Time: 00:24:27 min))] A line joining at any two point will remain inside there totally then it is convex. How do you check it algorithmically? You draw line between all the vertices. So let's say if you have n vertices, you have to draw lines between all the n vertices that means you have to draw an N square lines. Student: between two adjacent vertices, two adjacent vertices will always be inside, alternate vertices even then you will have a N square edges, so you have to check for N square line where they are totally inside. Yeah ((np)) ((00:25:17 min, not audible)) 2 that is order of N square. Any other way? Anyone, it's quite simple. Even then you will have about N square by 2 edges. Between alternate vertices. (Refer Slide Time: 00:26:07 min, not audible) I will explain to you this point.

Let's say I am taking this point, I will check it with all the other points if all the lines are inside but it's still a concave polygon that won't work. For alternate vertices see, let's say I am taking this edge. I have to decide whether this edge is completely inside the polygon or not, how do I decide that? Student: sir one point within this, just you check one point within this. Not one point. I would decide whether the complete edge is inside the polygon or not. How much is the total time required to check that it is not order of n because each edge deciding whether even one point is inside or outside. Student: that would mean, that was a order of N type. So total time required for checking whether it is a concave polygon will become N square again. In this case N square is the number of edges, the time required would become N cube. This is still N square time we are using, even simpler than that.

How do I decide whether this, whether any polygon is convex at a particular point or not? If the integer angle is less than 180, so we keep finding out the interior angle at all the end points. We have given a sequence of P_1 P_2 till P_n, we can find out the interior angle at each of these points. At any point you find out the interior angle is greater than 180 that mean it has to be a concave polygon. So, this way we can decide whether a particular polygon is concave but then let's say if we have clipping polygon which is shaped like this, we have to split it into a sequence of or into a set of convex polygons. We will have to do something like this and may be something like this 1 2 3 4, we get 4 convex polygons for this case. How do we do that? You can again think of algorithms, we don't go into that right now.

Student: Sir how do you make sure that both are these are giving the angle which we need, it might give that ?? Think of that, think of that. What he is basically saying is when you are finding out the angle here, you can find out the angle but how do you know which is interior which is exterior? You can find out the angle between two lines and how do we know which is angle is interior which angle is exterior? Give it some thought, its straight forward. The basic point is that if we have a concave clipping area then our clipping algorithm becomes much more complex. And other thing in related to clipping which will come back to later on is what is referred to as 3 D clipping. A 3- dimensional clipping that means if you have 3 dimensional lines or 3 dimensional entities and you want to clip them with respect to a clipping volume not an area, how do we do that?
So what are the situations when this comes and how it is tackled, we will come back to it later on when we talk of three dimensional, three D graphics. But any question that you have on 2 D clipping either polygonal clipping or clipping of lines segments or curves, we will take it up right now. Then we introduce another topic and that is transformations.

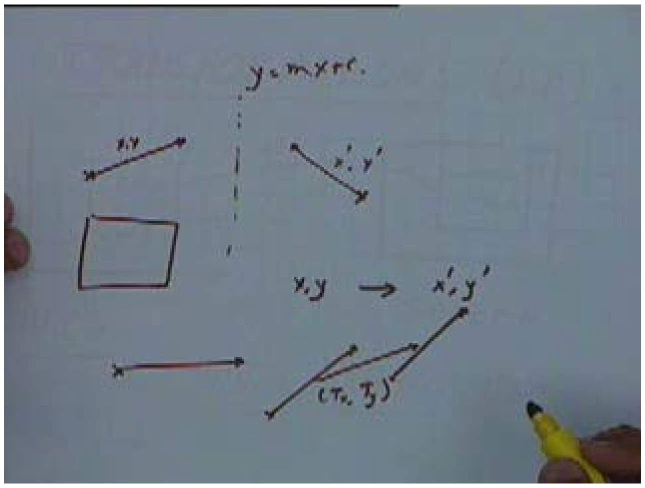
(Refer Slide Time: 00:30:59 min)



And right now again we will be talking of 2 D transformations. We will go onto 3 D very soon. Now what do you mean by transformations? We have already seen that if we have the world coordinate system, WCS, I am using as a short form of a world coordinate system. In this world coordinate system we can take a finite window and then this portion of the world coordinates is transformed on to the screen coordinate and on this screen coordinates. We take a view port and this portion of the window is then mapped on to the view port, after necessary clipping and so on. If you have any point here, let's say a point (x,y) that will be transformed onto the point here, that would be let's say (x_S, y_S). So a particular point here will get mapped on to a point (x_S, y_S.) Now this is what is referred to as a viewing transformation.

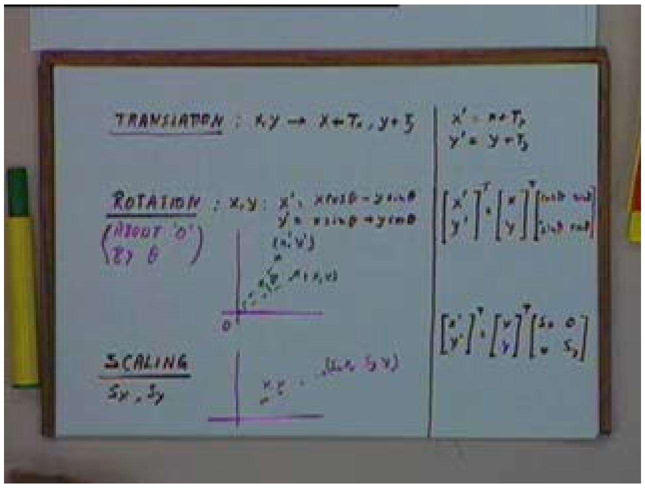
In viewing transformation the point in the world coordinate system is transformed on to the point on the view port in the screen coordinate system and the transformations involved in that are referred to as viewing transformations. Typically let's say in the world coordinate system this is my 0 0 and this point is my x lower, y lower and this point is x upper and y upper. When you transform it on to the screen coordinate system, this plane x_L y_L should be transformed into a 0 0. So this point will become a 0 0 and if this upper corner is xy then this will get transformed into xy. So this is one case where points in one coordinate system are transformed on to a second set of points.

9
(Refer Slide Time: 00:34:27 min)



There can be other situations where let’s say if we have some entity, either a line segment or a rectangle or any such entity and you want to either rotate it or deflect it about a particular axis. If you reflect this about this axis, about this plane let’s say this line segment will get transformed into a line segment like this. A point on this let’s say a point here xy will get transformed on to a point x prime y prime. If I know the equation of this line let’s say y is equal to mx plus c, how do I find x prime y prime for a given xy? The point xy has been transformed on to the point x prime y prime. So this is another case where transformations are important. Transformation need not just be reflection, they can be either a reflection or a rotation or scaling or any such case. We will see each of these transformations one by one and see we will see how it can be done. Student: sir with the help of matrices? Yes, with the help of matrices.

(Refer Slide Time: 00:35:58 min)



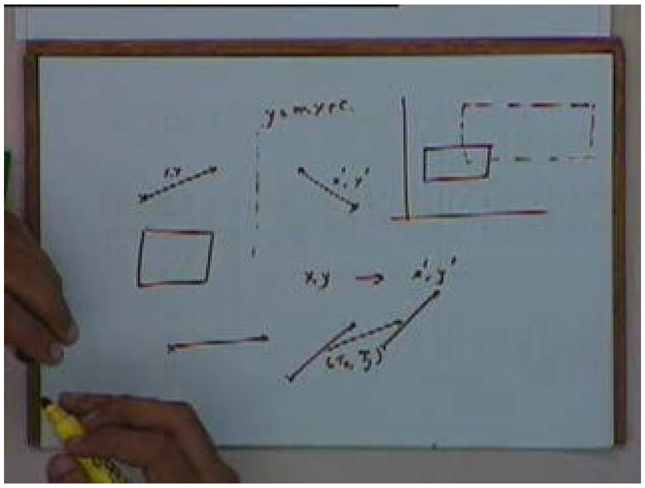
The first simple case is a translation. How do we define translation? A point xy will be transformed on to let's say x + T_x and y + T_y. So if I write it say mathematically, this will become x' = x + T_x, y'= y + T_y. This is simple translation. Any point is translated by a vector. If we have a line segment like this, this will be translated by a vector on to a location like this. This is translation by a vector given by T_x T_y and that can be captured simply be addition of T_x and T_y in the x and the y coordinates. Then we talk of rotation, we have a point xy and we want to rotate about origin, about the origin o and by an angle theta. This is my origin o and we have a point here, we want to rotate this point about the origin by an angle of theta. Theta is defined to be positive in the counterclockwise direction. This is a point xy, this is the point x', y'. What will be the value of x', y'? Anyone? How much will be the x'? x'=x cos θ - y sin θ. x cos θ - y sin θ and y'? Student: y'=x sin θ + y cos θ. x sin θ + y cos θ. Is that okay? Anyone having any doubt about this?

Now this can be written in the matrix form like this. I am writing x', y' as a row vector x', y' this is equal to the row vector xy multiplied by this 2 by 2 matrix. So $[x,y]^T$ multiplied by cos θ - sin θ will give us this first equation, $[x,y]^T$ multiplied by sin θ + cos θ will give us the second equation. So these two equations can be written in the matrix form like this and this transformation will capture a rotation about the origin by an angle of θ.

The third case is scaling and how do you define scaling? If you have a point xy and we want to scale it by let's say s_x in the x direction and s_y in the y direction. this xy will get transformed to a point which will be given as s_x multiplied by x comma s_y multiplied by y. the x coordinate will get multiplied by s_x, the y coordinates will get multiplied by s_y and in terms of matrix rotation, we will say x', y' will be equal to xy multiplied by $\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$

11

(Refer Slide Time: 00:42:00 min)



So what would happen in a case like this is if we have a rectangle and we scale it then this will may be get modified to something like this. The scaling of it by a factor of 2 in the x direction and a factor of 2 in the y direction, this rectangle will get modified to a location like this. This is basically magnifying by a factor of s_x and s_y in the x and y directions. So this is how scaling can be captured. Now by using a combination of these three transformations, we can get any other type of transformation by using combination of translation rotation and scaling, we will will be able to get deflection, we will be able to get rotation about an arbitrary point. We can rotate we need not always rotate about origin. We can rotate it about an arbitrary point here. But using a combination of these three. You will be able to get any arbitrary transformation. We will see how we can combine the transformation and how by combining different transformations, we can get arbitrarily transformation. We will see that in the next class.

12