Computer Aided Design

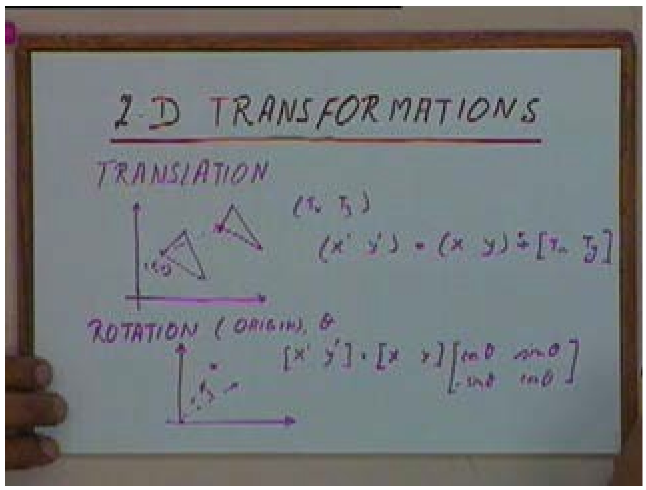
Prof. Anoop Chawla
Department of Mechanical Engineering
Indian Institute of Technology, Delhi

Lecture No. # 08

2 D Transformations

Last time we had introduced this topic of two dimensional transformations, we will go into details of this topic today.

(Refer Slide Time: 00:01:14 min)

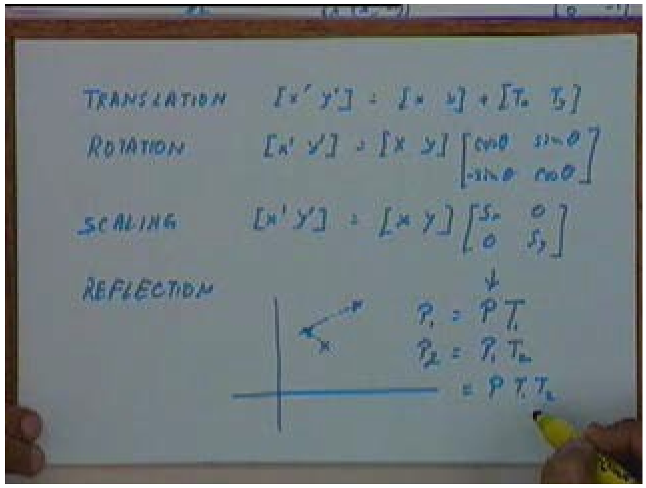


The first two dimensional transformation is a operation of translation. So if you have any point xy and that is translated by a vector given by T_x T_y then, we can say that the transform point (x' y') = (x y) + [T_x T_y]. So x'=x+ T_x and y'= y+ T_y. This simple translation operation in which all entities will get translated by a uniform vector. The shape etcetera of the entities will be retained. The other operation that we had seen last time was rotation and we said that in rotation about the origin by an angle theta, any point xy will get transformed to a point in this manner and we had said that [x' y'] = [x y] * $\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$

So x'= [x y] * $\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$ which x'= x *cosθ –y *sinθ and y' = x* sinθ +y* cosθ. This will be the transform point x' y'.

.

(Refer Slide Time: 00:03:18 min)

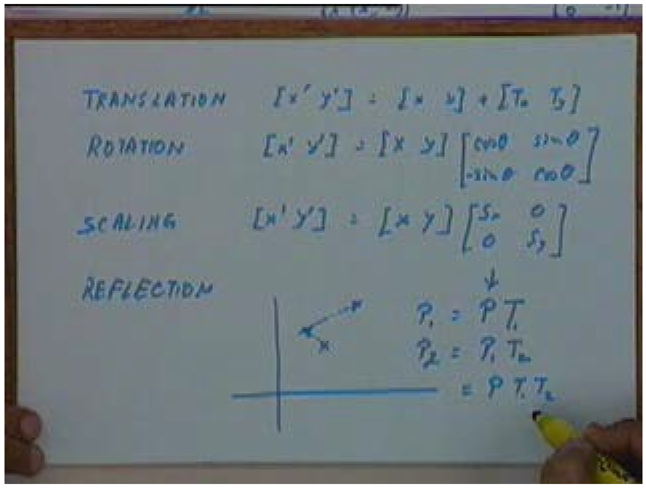


The third transformation that we had seen last time was the scaling operation and in scaling we had said that [x' y'] = [x y]* $\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$. The x coordinate will get multiplied by S_x and y coordinate will get multiplied by S_y, this is a scaling operation. So if you have a figure like this and scale it by a factor of 2 it will become something like this. All the x coordinates will get multiplied by 2, the y coordinates will also get multiplied by two of all the points. This is the scaling operation. Now let's see reflection.

If you have any arbitrary point xy and we reflect it about the y axis, this point will be x', y' which will be nothing but – x, y. So when you are reflecting about the y axis, the transformation can be written as [x' y'] = [x y]* $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$. Similarly if you are reflecting about the x axis, this point will go somewhere down at this location and the coordinates would then become x , -y and this matrix will become $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. Similarly if you reflect about the origin that means we take this radial line proceeded in this direction and we get a point in that direction which is -x , -y and for reflecting about the origin the transformation matrix will be $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$

So when you are reflecting about the y axis, the x axis will become, the x coordinate will change its sign and when you reflect about the x axis the y coordinate will change the sign and when you reflect about the origin both the coordinates will change the sign. Now if we compare these matrices with this scaling operations, we notice that for reflecting about the y axis it is nothing but the scaling operation when S_x =-1 and S_y = 1. Similarly the other two reflections can also be captured as scaling operations. So reflection is nothing but a specific case of scaling. Whenever you want to reflect a point or an entity about either x axis y axis or about the origin that can always be obtained by scaling by having a correct values of S_x and S_y either 1 or both of them will be - 1.

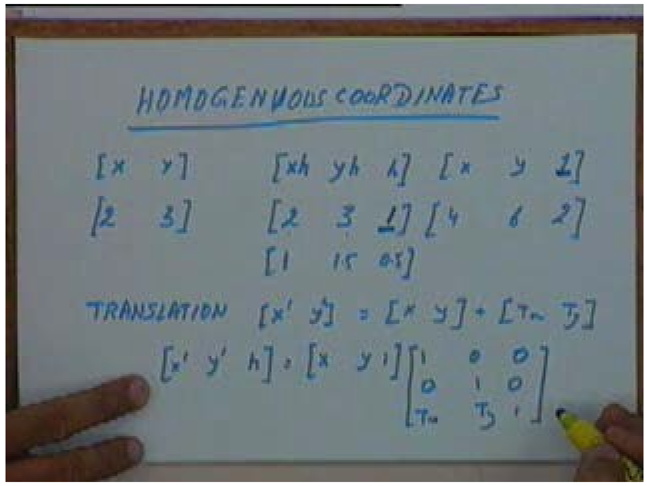
(Refer Slide Time: 00:07:23 min)



So for the four operations that we have seen we can summarize in this manner. Translation is captured by [x’ y’] = [x y] * [T_x T_y]. Rotation is captured by [x’ y’] = [x y] * $\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$ and scaling is captured by the equations [x’ y’] = [x y]* $\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$ and of course reflection is a specific case of scaling. Now in these three basic operations if you notice translation is captured by the addition of two matrices while both rotation as well as scaling are captured by multiplication of two matrices and multiplication is definitely more convenient because let’s say if you take any point in this coordinate system, we first want to rotate it about the origin we get this point then maybe you want to translate it and go the third point. We can capture that as a sequence of rotation or sequence of transformation sorry. We will get point P_1 will be equal to the point P multiplied by some transformation matrix T_1 where this T_1 corresponds to this rotation.

Now this point let’s say is to be scaled. If this point is to be scaled we will say P_2 will be equal to P_1 multiplied by some other transformation matrix T_2 which is nothing but P * T_1 T_2. So if all our matrix, all our transformations are captured as matrix multiplications the transformations can be very easily, multiple transformations can be very easily captured. Essentially with this same we will define what are called as homogeneous coordinates.

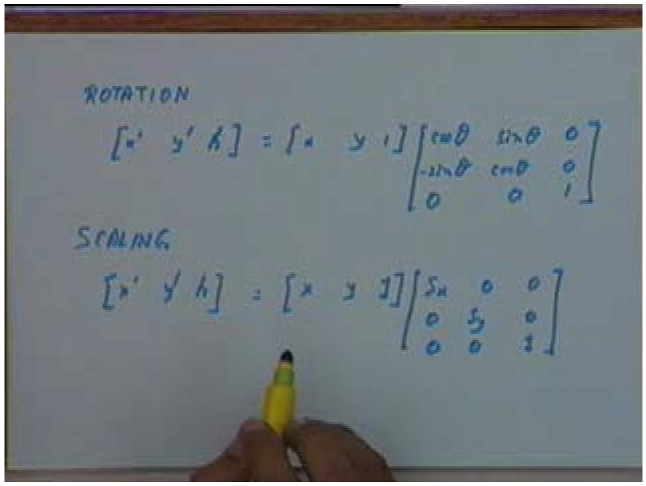
(Refer Slide Time: 00:10:03 min)



In homogeneous coordinates every point [x y] = [x*h y*h h] . So the point xy instead of being written as a tuple will now be written as consisting of three numbers. So if you have a point let's say [2 3], this point can be written as [2 3 1], it can also be written as [4 6 2], it can also be written as [1 1.5 0.5]. All these represent the same point. Basically what we will do is whatever be the value of this homogeneous coordinate, we will divide both of these by that value and then when this value is equal to one these two will give us the exact x and y values. So any point xy can always be represented as [x y 1], so [x y 1] is one of the homogeneous coordinate representation of the point x y because we have basically added one more coordinate to the two dimensional point and the advantage is, well to start with we were talking of translation.

Earlier we had written [x' y'] = [x y] * [T_x T_y]. Now [x' y' h] = [x y 1] * $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$. This will be equal to x y 1 and now since our points consists of three coordinates, our transformation matrix will also be a 3 by 3 matrix and for translation we will get $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$ and we say this is h. So x'=[x y h]* $\begin{bmatrix} 1 \\ 0 \\ T_x \end{bmatrix}$ which is nothing but x + T_x, y'=[x y h]* $\begin{bmatrix} 0 \\ 1 \\ T_y \end{bmatrix}$ which is nothing but y + T_y and the homogeneous coordinate h = =[x y 1]* $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ and we will get 1. This way even translation can be captured easily as matrix multiplication.

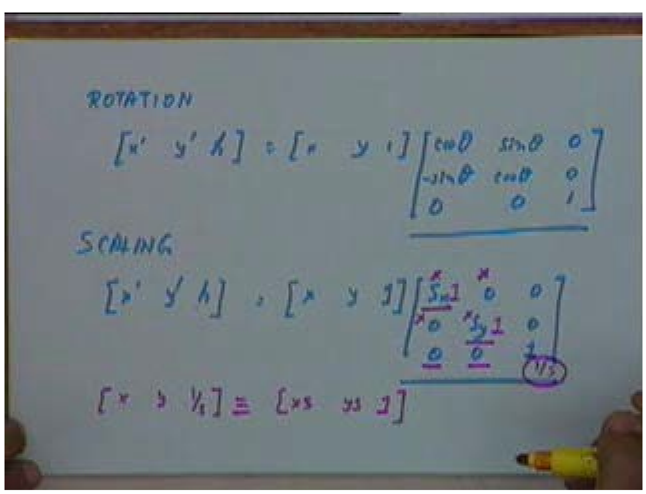
(Refer Slide Time: 00:13:53 min)



The other two operations, we were talking about rotation [x 'y' h] = [x y 1]* $\begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

The basic matrix will remain the same. In other places we will just add zero's and one's. This way x prime will be equal to this row vector multiplied by this column vector which gives us the same equation as we had earlier. Similarly y prime will be this row vector multiplied by this column vector, it will again give us the same equation as earlier and the homogeneous coordinate will just give one. So rotation can also be captured in a similar manner. Then scaling, that can also be captured in a similar manner and we have to scale by an amount of S_x, this row vector will get multiplied by this column vector and x'= x* S_x and y'= y* S_y , the homogeneous coordinate h will again be equal to 1.

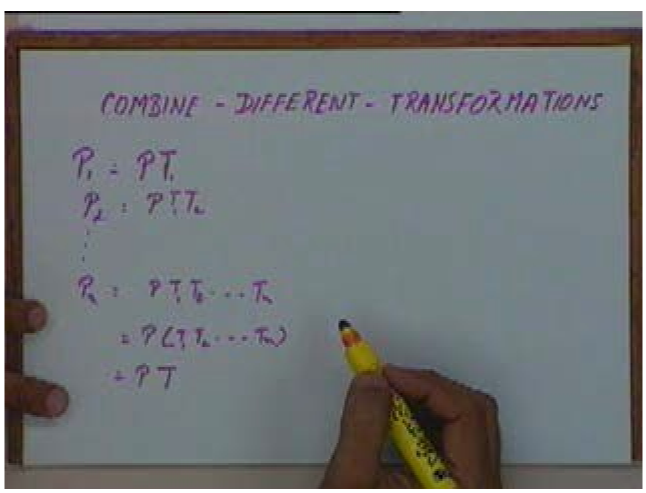
(Refer Slide Time: 00:15:51 min)



So this way by using homogeneous coordinates, we first want to capture translation by this matrix multiplication operation and similarly even rotation and scaling are captured as matrix multiplication operations. We are keeping h equal to 1. Yes. [Conversation between Student and Professor - Not audible ((00:16:05 min))] see if we divide h by 1 over S_x we will get uniform scaling in the both the directions. Again. Multiplying by 1, see I will just come to that in a minute. The reason why we have added homogeneous coordinates is, the first primary reason is that translation should be captured as matrix multiplication. If we don't add the homogeneous coordinate, translation will not be captured as a matrix multiplication. As a result of that we will not be able to combine different kinds of transformation by a single matrix, we will just see how that is to be done. The second thing that you are saying is that instead of this one, if I change this to some value let's say 1 over S, if I change it to 1 over S and these for the timing being I will let's say both equal to 1.

Now we will get uniform scaling in the x and the y direction, not different scaling we will get uniform. Yeah, you won't get different scaling, you will get uniform scaling in the x and the y direction. This is 1, this is 1 and this is 1 over S this point will become [x y 1/s] = [x*s y*s 1]. So if you want uniform scaling in both the directions, we can just give a 1 over S factor in the bottom right corner of the transmission matrix. If you want non uniform scaling then we need to give one factor over here and one factor over here. If you want to translate a point then we need to add some value here and some value here. If you need to rotate a point about the origin then you will have some value cos theta here, sin theta here minus sin theta here and cos theta here. This is when you want to rotate by an angle of theta counter clockwise about the origin.

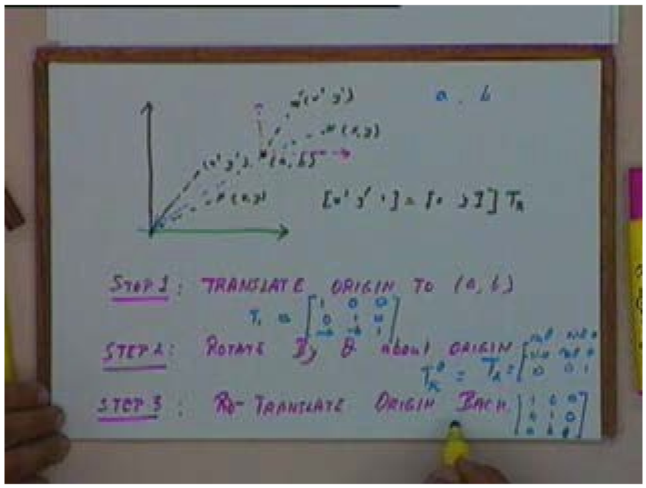
(Refer Slide Time: 00:18:42 min)



Now I had mentioned that if you want to combine different operation, different transformations you can take any point P_1= P* T_1. We can take this point P_1 transformed by another matrix and we will get P_2 =P* T_1 T_2 and we continue this operation, we will finally get P_n =P* T_1 T_2……………… T_n.

This means that for transforming any point P, you can take each of these individual transformations, multiply them together or you can write it like this. I can get one combined transformation matrix for all these transformations. This way we are able to do purely because all the transformations are represented in the form of matrix multiplications. We just see one specific example of where this kind of thing is very useful.

(Refer Slide Time: 00:20:29 min)

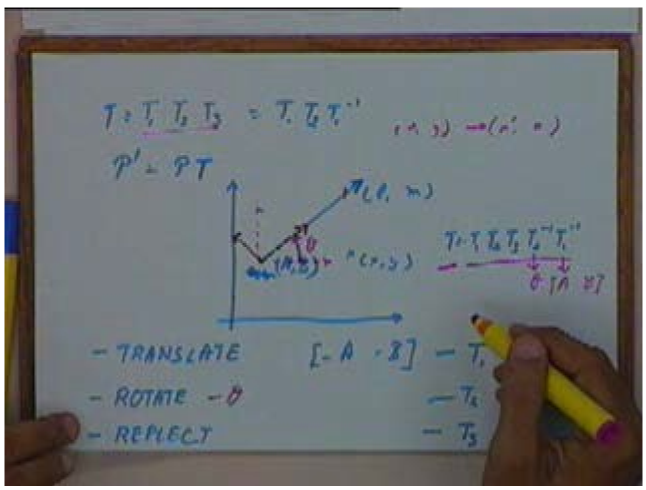


So far when we were taking of rotation, we were rotating a point about the origin. You have taken a point x y and you are rotating it about the origin by an angle of theta. Instead of that if I have a point xy and I want to rotate it about any arbitrary point ab, how do I find out the values of x prime and y prime? I know that if I am rotating about the origin I know what is the transformation matrix meant for that. If this is the case I can easily get [x' y' 1] = [x y 1] *T_R. But if I have to rotate about the point ab, how do I find the transformation matrix for this case? What we will do is we will translate the axis such that ab becomes the origin, this origin should come to this location. We will then rotate the point x y to x' y' and then we will translate this axis back to this position. So our step 1 would be to translate origin to a b, step 2 would be to rotate by theta about origin and step 3 would be to translate origin back to its original position.

So for rotating a point about any arbitrary point that can be done as a sequence of these three steps. And how do we translate the origin to the point ab? What will be the transformation matrix for that? Anyone? Translate all the points by –a, - b. We will translate all the points by minus a minus b not by ab. So transformation matrix for this will be $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -a & -b & 1 \end{bmatrix}$. This is the first transformation matrix, $T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -a & -b & 1 \end{bmatrix}$. Is that okay? Is this clear? So why we were using - a and - b over here? No. see what we want is that this point ab should now have the coordinate of 0 0. So to a and b, I have to add minus a and minus b that means what I am effectively doing is this point is being translated back to the origin by this amount.

So the translation vector is - a - b not just ab. So when I am translating the origin to ab that is the origin should become this point that means coordinates of ab should now become 0 0 that is a negative translation by ab or translation by - a - b. So the first step is we translate the origin to ab using this transformation matrix then you need to rotate by theta about the origin. What is the transformation matrix for that? The same as a one for rotation, $\begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$. For rotation about the origin we use the same matrix as that is meant for rotation by an angle theta. Then translating the origin back, how do we translate the origin back? Again? The inverse of T_1 or that could be ab. this would be $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & 1 \end{bmatrix}$ and I want to translate the origin back. Again this point should retain the coordinate of ab. the points which are 0 0 should now retain the coordinate of a b, so we will add ab to all the coordinates.

(Refer Slide Time: 00:26:49 min)



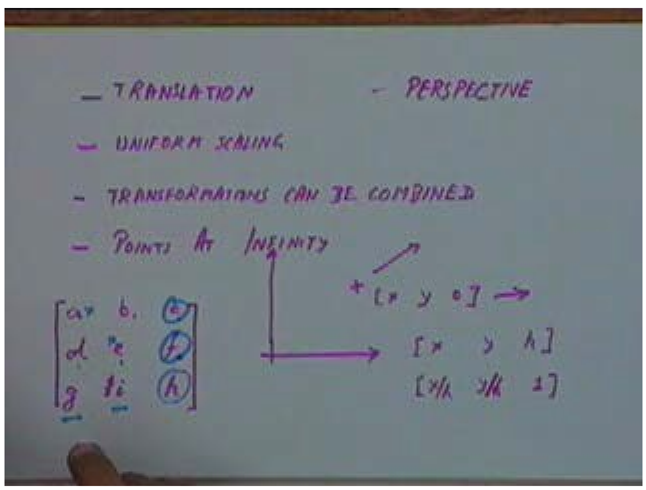
So if you combine these three or combine transmission matrix will be the product of the three transformation matrices T= T_1 T_2 T_3 or we can also write this as T_1 T_2 T_1^{-1}. And now if you want to rotate any arbitrary point x y about the point a b, all that we need to do is P'= P* T. Is that okay? So this way we can combine any sets of transformations into a single transformation.

Similarly if we want to you have a point, arbitrary point xy and you want to reflect it about an inclined axis, how do we reflect the point about the x of the y axis. We use the scaling operations. Now we want to reflect it about any arbitrary axis. Let's say the axis is passing through the point ab and I have got direction cosines of (l, m). How do we reflect the point xy about this axis? Anyone? [Conversation between Student and Professor - Not audible ((00:28:36 min))]. So first translate, we translate the origin to this point so that this ab now becomes the origin. So we translating by a vector of [-A –B] then we rotate, for this you will get some transformation matrix T_1. Then we rotate the axis so that this direction coincides with the let's say the x axis that means in the first step my axis has become like this. In the second step I rotate my axis so that my axis becomes like this.

That is rotate, this will give us the transmission matrix T_2 then I will reflect about the x axis that will give us the transformation matrix T_3 and then I will do the reverse of these two. So the combined transformation will become T will be $T_1\ T_2\ T_3\ T_2^{-1}\ T_1^{-1}$. The sequence is important; I cannot take the inverses in the opposite order. I am doing T_1 then T_2 so the inverses have to be taken T_2 inverse first then T_1 inverse. Any questions on this part? These individual transformation matrices, it will be able to write them on your own. Transformation matrix for translation, for rotation, for reflection and the reverse of these two transformations, only one thing you should be careful about is in step two. When we are rotating the axis, let's say this angle with an angle theta, we will have to write the transformation matrix for minus theta because what we want is that this direction should not become the x axis.

So a point on this axis will be a point on the x axis. So this point which is earlier xy should now become that is from x', 0 that will be obtained when I am rotating actually by an angle of minus theta. Similarly translation we have done minus a - b, so rotation will be by minus theta. Reflection, in T_2^{-1} here we will have rotation by theta and translation by a b. So we have basically seen how we can combine different operations and get a combined transformation matrix for a series of operations.

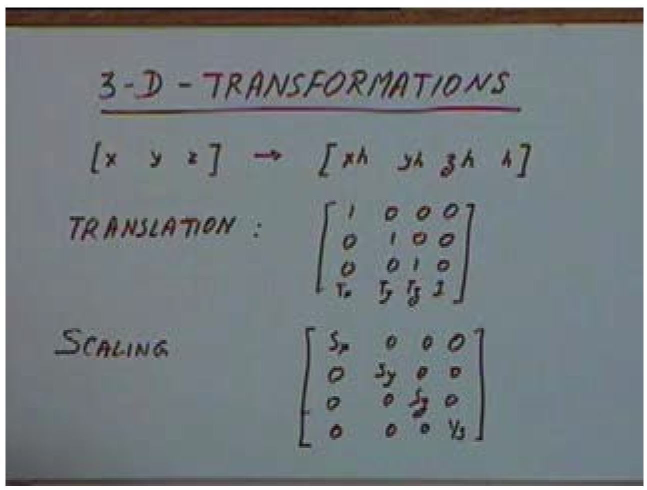
(Refer Slide Time: 00:33:05 min)



The basic advantage that we have of using homogeneous coordinates, the first advantage was that transmission can be captured as matrix multiplication. The second advantage that we have seen is uniform scaling can be captured by one single parameter. The third we have seen is that the transformations can be combined. One more advantage of using homogeneous coordinates is that if we want to represent the point at infinity let's say infinity in this direction, you can take any point here xy and [x y 0] will be a point at infinity in that direction because we have said that a homogeneous coordinate[x y h] actually corresponds to [x/h, y/ h 1]. So this h is equal to 0, so this point represents a point at infinity in the direction of x y. So another advantage of using homogeneous coordinate is points at infinity can be easily captured.

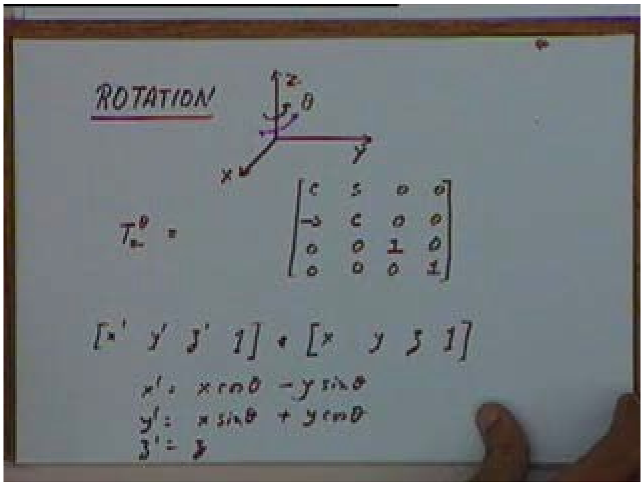
Now in the homogeneous transformation matrix, you have got 9 terms, I will just write them as $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$. Of these 9 we have seen that this is a homogeneous coordinate, it gives uniform scaling, these four are giving us rotation, these two in addition to rotation they also give us scaling and these two are giving us translation. So far these two where retained at zero for all transformation, we have not tried to modify these two. We will see later on that these two are used for capturing perspective transformations. You know what is a perspective view? No, and you will see that later.
If you want to capture the perspective view of any object then we will be using these two coordinates to capture the perspective transformation. So another advantage is that we will see of these homogeneous coordinates later on will be perspective transformations can easily be captured, a perspective view can easily be obtained by such transformations. So these are some of the advantages of using homogeneous coordinates. Any question on this part? Then I will just explain this idea to transformation in three dimensions.

(Refer Slide Time: 00:37:01 min)



So if you have any 3 D point, if you have any 3 D point which is x y z again we will directly go into homogeneous coordinate, [x y z] = [x*h y*h z*h h]. This would be, a 3 D point would be represented as a four tuple where h is the, fourth coordinate will be the homogeneous one. Translation, I don't know you need to define translation again. The translation matrix will be a 4 by 4 matrix now. So we want to translate any point by T_x T_y T_z, in the bottom row we have T_x T_y and T_z in one place $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$. Scaling will get $\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0_z & 1/s \end{bmatrix}$. 1/ S is for uniform scaling S_x S_y and S_z of a non-uniform scaling. Now let's come to rotation.

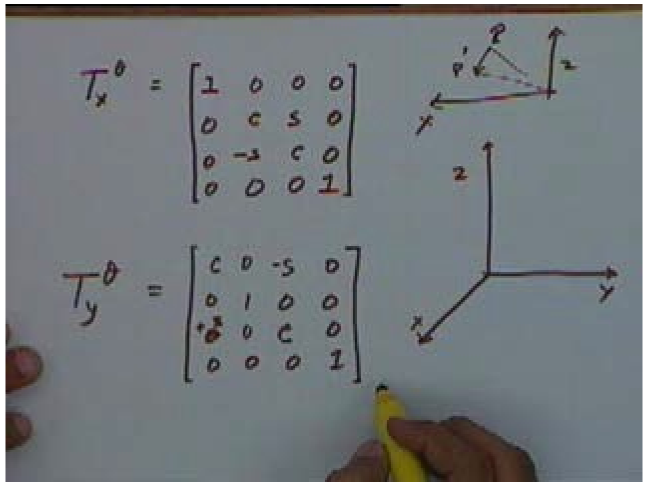
(Refer Slide Time: 00:39:07 min)



In the two dimensional case we were rotating about the point which was the origin. In a three dimensional case we can’t rotate about a point, we have to rotate about an axis. So let’s take the first case. We want to rotate about the z axis by an angle theta counter clockwise as you look from the top. For that what will be your transformation matrix? x’ y’ z’ prime, these are the three points. What will be the value of x prime? Our initial point was [x y z 1] x’= x *cosθ –y *sinθ and y’ = x* sinθ +y* cosθ and z’= z. The value of z prime will not change, it is rotating about the z axis.

Any arbitrary point let’s say here and you are rotating it like this, the z value will remain the same. So the transformation matrix for this case would be $\begin{bmatrix} c & s & 0 & 0 \\ -s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, c stands for cos theta and s stands for sin theta. So this will be the transformation matrix for rotating about the z axis. We can write it as T_z by an angle theta. This is by transformation matrix for this one.

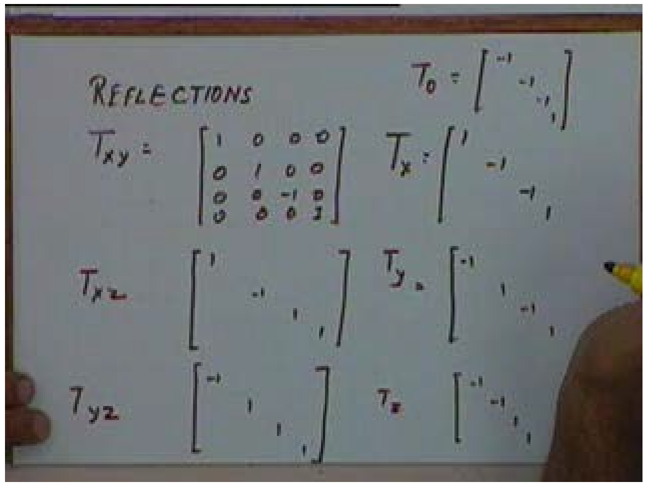
(Refer Slide Time: 00:41:57 min)



Now if we want to rotate about the x axis, what will be the transformation matrix for rotation about the x axis by an angle theta? The x coordinate has to remain unchanged, this will be $\begin{bmatrix} 1 & s & 0 & 0 \\ 0 & c & s & 0 \\ 0 & -s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. These four values we have to fill up, what will they be? cos sin minus sin cos. Is that okay? I have directly written the result, we got this. We want to rotate the, about theta sorry about the y axis, the y coordinate will remain unchanged. These four coordinates these four values, we will have to fill up. $\begin{bmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ What will these values be? Anyone? c s minus s c. Is that correct? Minus s c will be [Conversation between Student and Professor - Not audible (00:43:43 min)] no, just think of the figure x y z. So on this xz plane I am drawing the xz plane now, this is x, this is z. I am rotating by an angle of theta. This is my point P, this will be my point P prime, the z value of this point is going to decrease. [Conversation between Student and Professor - Not audible ((00:44:37 min))] so this will be plus, this will be minus. You can verify it from this figure, the z value is decreasing, the x value is increasing. So the x value will be c' or cosθ + z sinθ, the y value will remain unchanged.

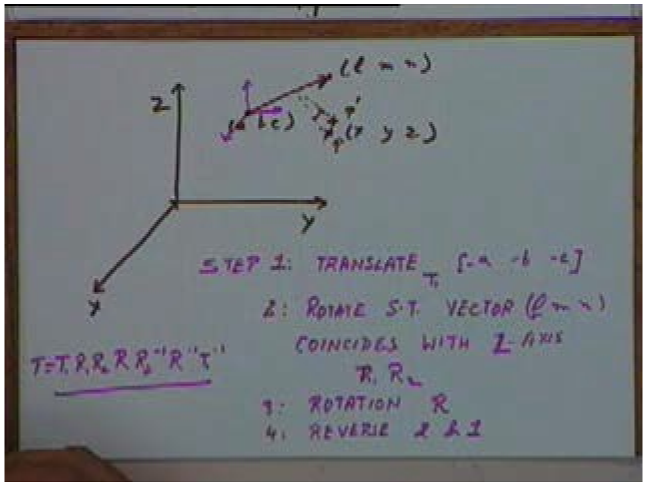
So when you want to rotate about the y axis by an angle theta this will be your transformation matrix. So the basic transformations we have seen, our basic transformations were translation, scaling then rotation about the x axis sorry about the z axis and about the x and the y axis. Now from these basic operations let's try and get some of the other operations.

(Refer Slide Time: 00:46:01 min)



The first is reflections. If you want to reflect about the xy plane, what will happen? The z value will become negative $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, the rest will remain the same. Similarly if you want to reflect about the xz plane, the y value will become negative so for that $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, all the others will be 0. If you want to reflect about the yz plane, you have $\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, all the others will be 0. If you want to reflect about an axis so if you want to reflect about the x axis, what will happen? Both y and z values will. Both y and z values will become will change. Similarly with respect to the y axis, x and z values will change. If you want to reflect about the z axis, x and y values will change. If you want to reflect about the origin, reflecting about the origin all the values will change. So reflections are very easily captured as scaling operations.

(Refer Slide Time: 00:48:21 min)



Now let's again talk of rotation, a rotation about any arbitrary point or an about an arbitrary axis. So we want to rotate about an axis like this. This is let's say the point a b c and this vector has got direction cosines of l m n. How do we rotate about this arbitrary vector? We have any arbitrary point x y z, this is my x axis, this is my y axis, this is my z axis. I want to rotate this point x y z about this vector, so we will get something like this again by an angle of theta. This will be transformed to some point over here, this is P this will be transformed to P'. How do we find out the coordinates of P', what will be sequence of steps involved. Anyone? We translate so that this becomes the origin, translate by [-a –b –c]. So now my axis would look something like this. What is the next step? We rotate so that rotate such that vector l m n coincides with one axis, any axis for sake of argument let us say this will be our z axis. This will take two rotations. This will take two rotations, what will be the two rotations? We will first rotate so that this vector comes into the xz plane then we will rotate so that x coincides with the z.

So this will be one matrix T_1, this will be two rotations let's say R_1 and R_2 then we will carry out the rotation, that let's say is R and then we will say reverse of steps, reverse 2 and 1. Whatever transformations we have obtained in step 2, we will do the opposite of that. Whatever transformation is obtained in step 1, we will do apposite of that. The combined transformation will come out to be $T_1 R_1 R_2 R R_2^{-1} R_1^{-1} T_1^{-1}$. So if we want to rotate a point x y z about any arbitrary vector that would be done by a sequence of these 7 steps.

In the next class what we will see is how these two rotations are done, what will be angle then about which axis these two rotations will be done. We will see that in detail in the next class. [Conversation between Student and Professor - Not audible ((00:52:17 min))] again. Same thing we will put this as T_2 and this will become T_2 inverse. You take this as let's say T_2 and then this as T_2 inverse. You can do that but once you have multiplied the three matrices, finding out the inverse is going to be a difficult job. Four by four. four by four matrices you have to find out the inverse by inverse ((00:52:50 min) by the matrix inverse algorithm. So each of these individuals, the inverse is just a rotation by an opposite angle of minus theta. So this T_2 inverse can be found out easily by matrix multiplication little easier that is I will prefer to do this kind of thing. Nevertheless, you can multiply them and then find out the inverse, you will still get the same result. Any other questions?