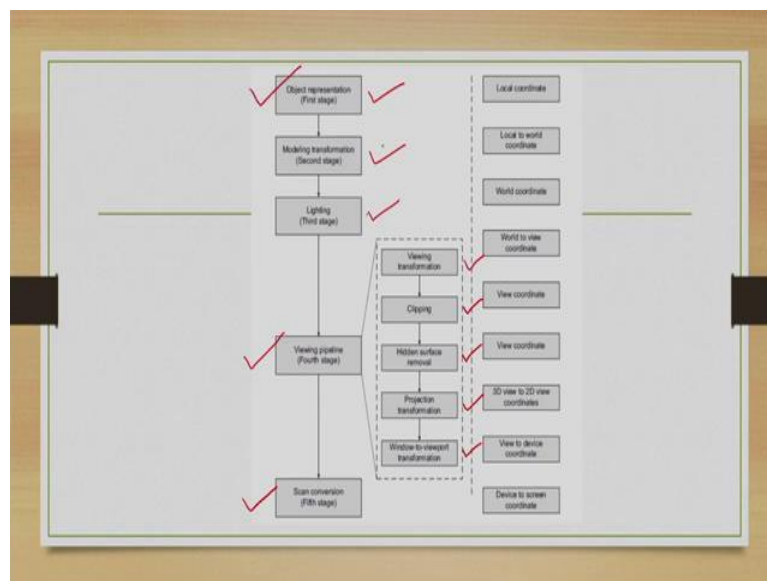


Computer Graphics
Professor. Samit Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati
Lecture No. 11
Matrix representation and composition of transformations

Hello and welcome to lecture number 11 in the course Computer graphics. We are discussing different stages of the graphics pipeline. Before we go into today's topic, let us again quickly recap the stages and where we are currently.

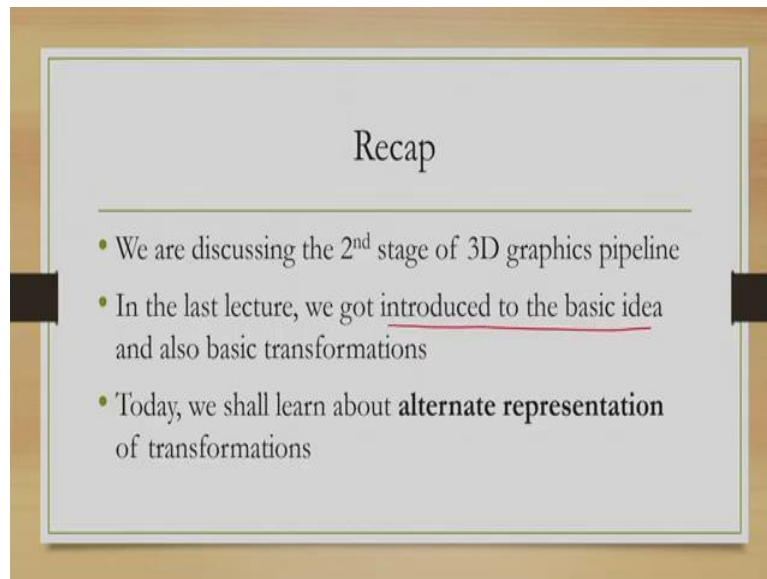
(Refer Slide Time: 00:50)



So, there are 5 stages, first stage is object representation, second stage is modelling transformation, third stage is lighting or colouring, fourth stage is viewing pipeline which itself consists of few sub stages, 5 sub stages mainly; viewing transformation, clipping, hidden surface removal, projects and transformation and window to viewport transformation, and the last stage is scan conversion.

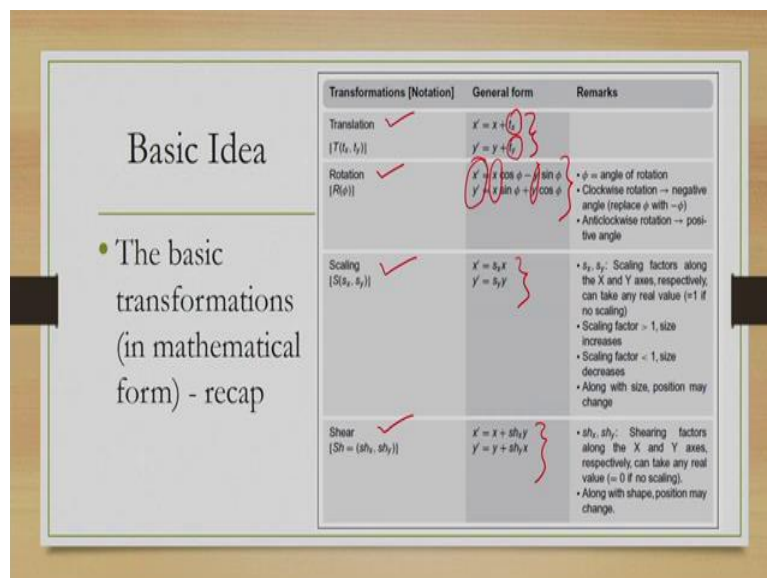
So, what we do in this stage is, the first stage we define objects, in the second stage, we put them together to construct a scene in world coordinate and then in the subsequent stages, we process those till we perform rendering on the actual computer screen. We have already discussed the first stage object representation, currently we are discussing the second stage that is modelling transformation.

(Refer Slide Time: 1:56)



So, in the last lecture, we got introduced to the basic idea what we mean by modelling transformation and we also introduced 4 basic transformations using which we perform any type of modelling transformation. Now, today, we are going to learn about representation. So, in the previous lecture we talked about how to represent the transformations, today we will learn about an alternative way of representing those transformations.

(Refer Slide Time: 02:32)



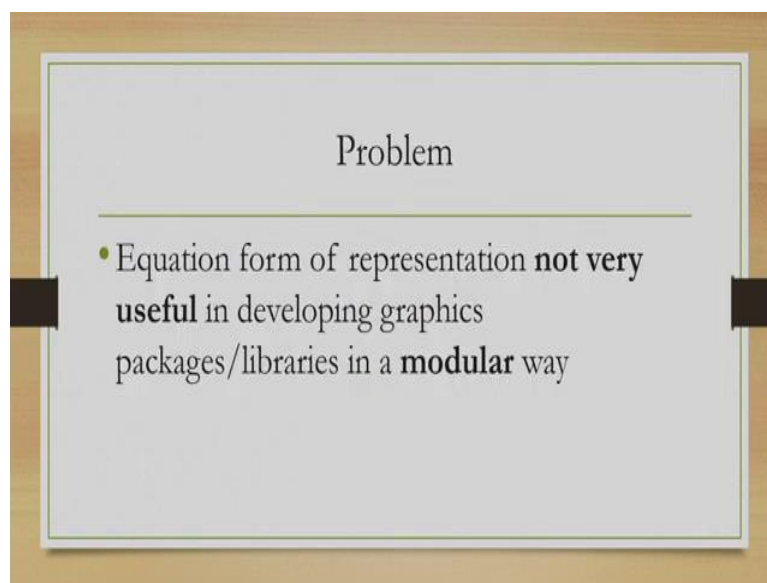
What we have seen in the previous lecture how we can represent the transformation. If you may recollect, there are 4 basic transformations; translation, rotation, scaling and shearing. Using these 4 basic transformations, we can perform any geometric transformation on any

object, either applying any one of these 4 transformations or applying these transformations in sequence one after another multiple times and so on.

And we discussed these transformations in terms of equations. For translation, we discussed the relationship between the original point and the transformed point that is point after translation as shown in these two equations. For rotation, similarly, we established the relationship between the original point and the transformed point using these two equations.

Same was the case with scaling, again two equations; one each for the two coordinates and shear. In these equations starting with translation, we used some parameters t_x , t_y or the amount of translations along x and y direction. Similarly, ϕ is the angle of rotation in these rotation equations, s_x , s_y are the scaling factors along the x and y directions respectively. And sh_x , sh_y are the shearing factors along the x and y directions respectively.

(Refer Slide Time: 04:55)



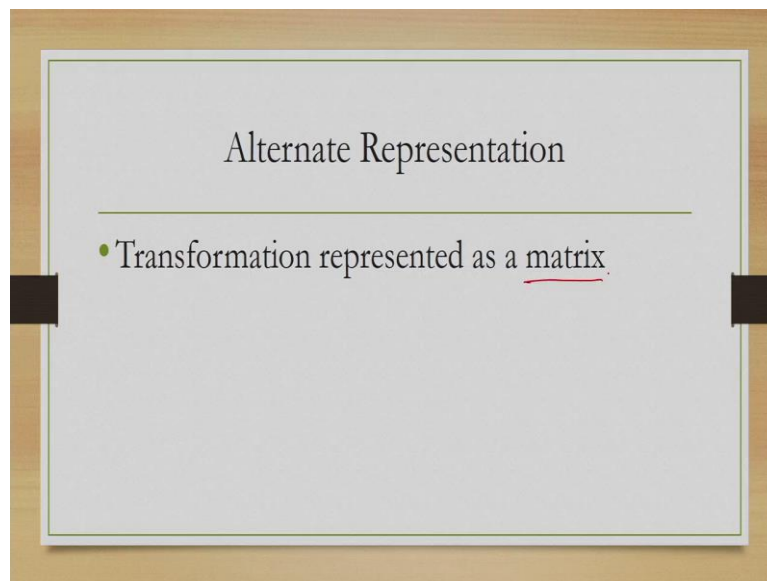
So, as we have shown we can actually use these equations to represent the transformations. Now, as we have discussed in introductory lectures, there are graphics packages, there are graphics libraries that are developed to actually make the life of a developer easier so that a developer need not always implement the individual components of a graphics pipeline to develop a product. In order to build a package or in order to develop library functions, what we need, we need modularity, we need standard way of defining inputs and outputs for each function.

Unfortunately, the equation based representations of transformations do not support such modularity. So, when we are trying to represent transformations using equations, it is difficult

to modularize the overall pipeline in terms of standardized input and output and standardized functions, because in subsequent stages of the pipeline, we will see other transformations and each of those transformations will have different equations represented in different forms and formats.

So, then it will be very difficult to actually combine these different stages and implement a package or a library, where the user will not be bothered about the internal working of the package.

(Refer Slide Time: 06:50)



To maintain this modularity, equation based representations are not suitable. We require some alternative representation and one such alternative representation, which supports our need for a modularized modular based system development is matrix representation. So, we can actually represent transformations in the form of matrices. And later on, we will see that other stages of the pipeline can also be implemented by representing basic operations in the form of matrices.

So, there will be some synergy between different stages and it will be easier to implement those stages in the form of predefined packages, functions or libraries.

(Refer Slide Time: 07:48)

Example

- We can represent scaling transformation as

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \quad 2 \times 2$$

So, how these matrices look like let us take, for example, the scaling transformation. Now, if we want to represent scaling in the form of matrices, what we will do? We will create a matrix in this form, a 2×2 matrix and the scaling factors will be positioned along the diagonal as shown here.

(Refer Slide Time: 08:20)

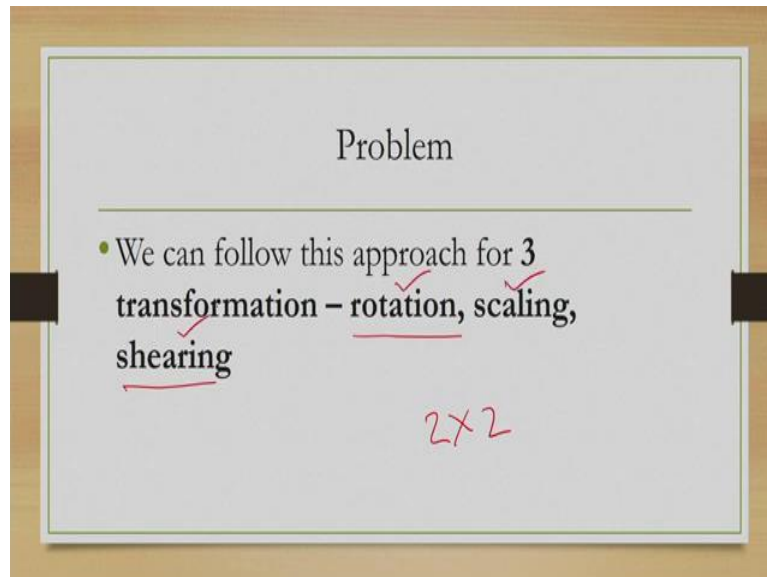
How it Works

- Given a point $P(x, y)$, we can represent it as a (column) vector $P = \begin{bmatrix} x \\ y \end{bmatrix}$ ✓
- Multiply transformation matrix with column vector together (i.e., $P' = S \cdot P$) to get the new points

Now, how to apply this transformation then? Suppose we are given a point $P(x, y)$ and we want to transform it by scaling. So, what we will do? We can represent this point as a column vector shown here and then multiply that transformation matrix with the column vector. This is the dot product of the matrices to get the new points. So essentially, we need to have matrix

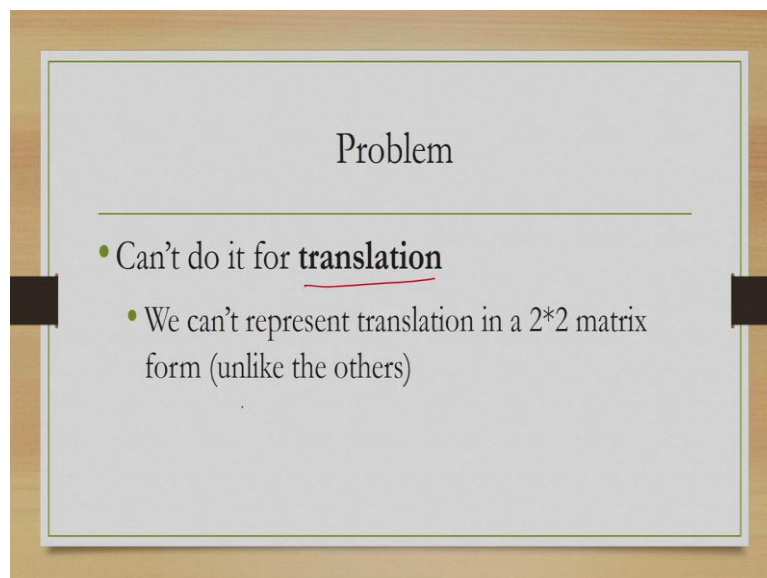
multiplication. And this form of representing the operations of transformation actually is what makes it easier to implement in a modular way.

(Refer Slide Time: 9:20)



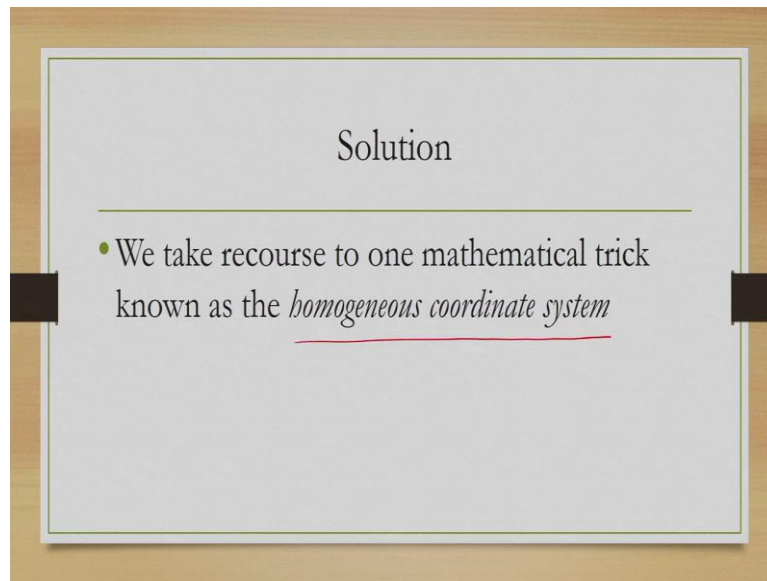
So, we have represented scaling in terms of 2×2 matrix. We can do the same with rotation, we can have a 2×2 matrix for representing rotation transformation, as well as shearing. So then, we can have 2×2 matrices for the 3 operations; rotation, scaling and shearing.

(Refer Slide Time: 9:50)



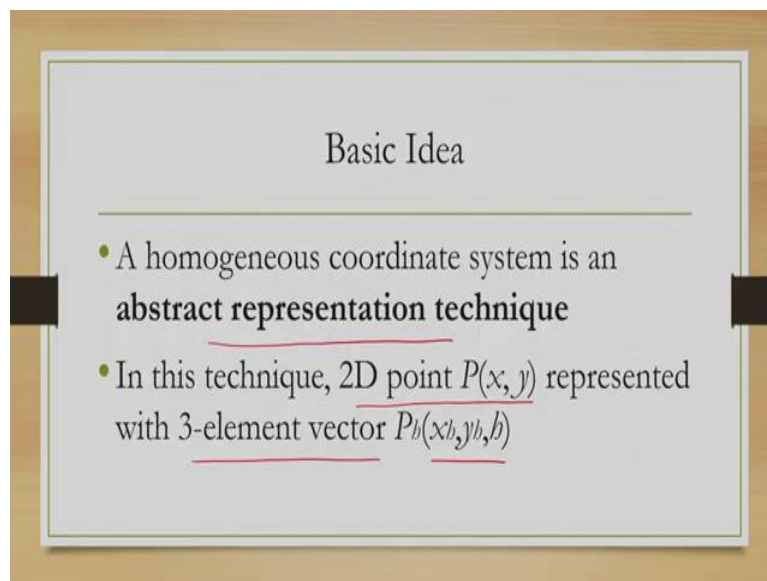
Unfortunately, 2×2 matrices would not serve our purpose. Because, no matter how much we try, we will not be able to implement or represent the translation transformation using a 2×2 matrix unlike the other 3 basic transformations that is not possible.

(Refer Slide Time: 10:33)



So, in order to avoid this problem, in order to address this issue, we go for another type of matrix representation, which is called representation in a homogeneous coordinate system. Now, what this homogeneous coordinate system based matrices representation refers to?

(Refer Slide Time: 10:48)



So, essentially it is an abstract representation technique that means, this coordinate system actually does not exist in the physical sense, it is purely mathematical, purely abstract. So, there may be physically a 2 dimensional point which we transform to a 3 dimensional abstract coordinate system called homogeneous coordinate system. So, each 2D point represented by these 2 coordinates x and y can be represented with a 3 element vector as

shown here, each of these elements correspond to the coordinates in the homogeneous coordinate system.

So, we are transforming a 2D point into a 3D space in this case, the 3D space is the abstract homogeneous coordinate space and each point is represented with a 3 element vector.

(Refer Slide Time: 11:56)

Basic Idea

- Relationship between $P(x, y)$ and $P_h(x_h, y_h, h)$

$$x = \frac{x_h}{h}, y = \frac{y_h}{h}$$

- h called homogeneous factor – can take any non-zero value

So, what is the relationship between these 2 representations? So, we have a 2D point represented by its 2 coordinates x and y . And now, we have transformed it or we are presenting the same point in a 3 dimensional space called homogeneous coordinate system where we are representing the same point with 3 coordinate values x_h , y_h and h . So, what are the relationships between these quantities?

Now, the original coordinate x is equals to the x coordinate in the homogeneous coordinate system divided by h , which is the third coordinate value and original coordinate y is equals to the y coordinate in the homogeneous coordinate system divided by again the h , h is called homogeneous factor and it is important to note that it can take any nonzero value, it must be nonzero value.

(Refer Slide Time: 13:05)

Basic Idea

- Any point of the form $P_h(x_h, y_h, 0)$ is considered to be at infinity
- Point $(0,0,0)$ **not allowed** $\frac{0}{0}$

There are a few more things we should note here, since, we are considering h to be homogeneous factor. So, if h is 0, then we consider that point to be at infinity in the homogeneous coordinate system, and there is no concept of origin since 0×0 is not defined so, we usually do not allow the origin point where everything is 0. So, these two things we should remember while dealing with homogeneous coordinate system, first thing is if h becomes 0, then we consider that point to be at infinity and there is no concept of origin in the homogeneous coordinate system.

(Refer Slide Time: 14:05)

Converting GTs to HCS

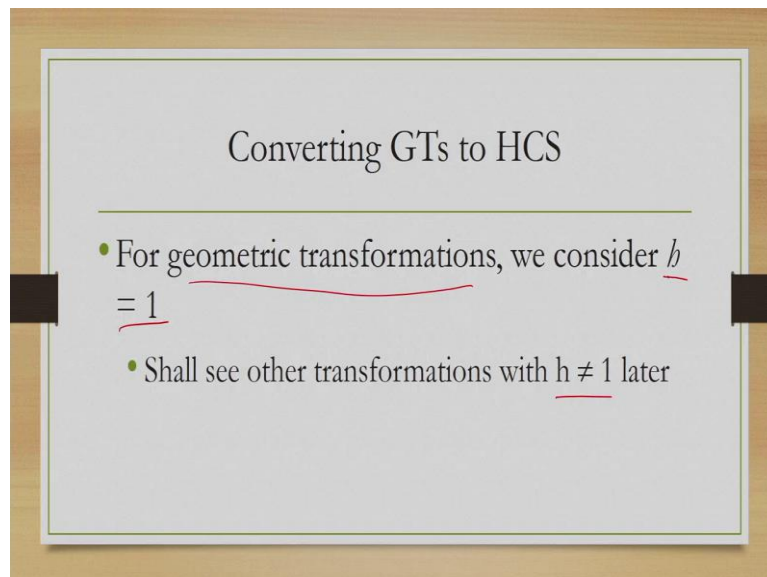
- Earlier 2×2 matrices transform to 3×3 matrices (in general, any $N \times N$ transformation matrix is converted to $(N + 1) \times (N + 1)$ matrix)

Now, let us try to understand how we can convert this geometric transformation matrices into the matrices in the homogeneous coordinate system. So, earlier we had this 2 by 2 matrices

representing the 3 basic transformation out of 4; rotation, scaling and shearing. As we have already mentioned, so this 2×2 matrices will transform to 3×3 matrices in the homogeneous coordinate system. In fact, in general if there is an $N \times N$ matrix transformation matrices, it is converted to $(N+1) \times (N+1)$ matrices.

Now, if we represent a transformation matrix, a 2D transformation matrix using a 3×3 matrix, then we will be able to represent translation as well so our earlier problem will be resolved, earlier we were unable to represent translation using a 2×2 matrix, although you are able to represent the other 3 basic transformations. Now, with homogeneous representation, we will be able to avoid that, we will be able to represent all the 4 basic transformation using 3×3 matrices.

(Refer Slide Time: 15:36)

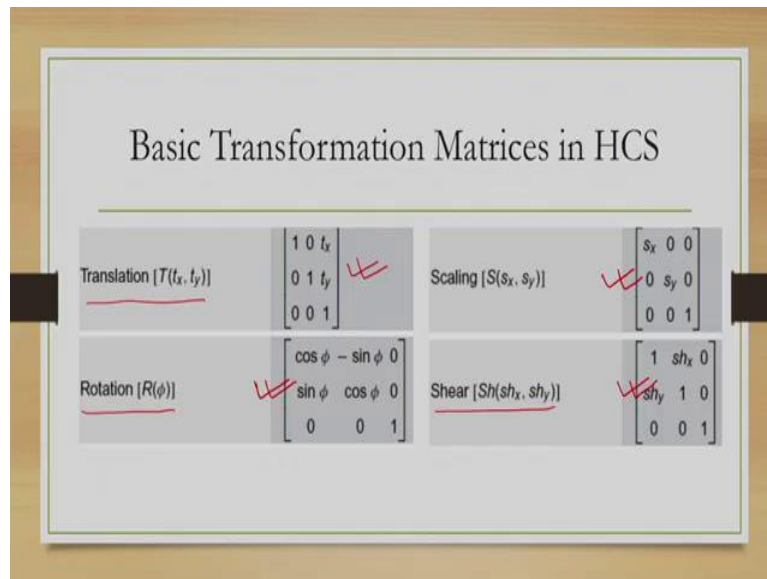


Converting GTs to HCS

- For geometric transformations, we consider $\underline{h = 1}$
- Shall see other transformations with $\underline{h \neq 1}$ later

Another thing we should keep in mind is that, when we are talking about geometric transformations, we always consider h to be 1. So, h value will always be 1. However, there are other transformations that we will encounter in our subsequent lectures, where h is not equal to 1.

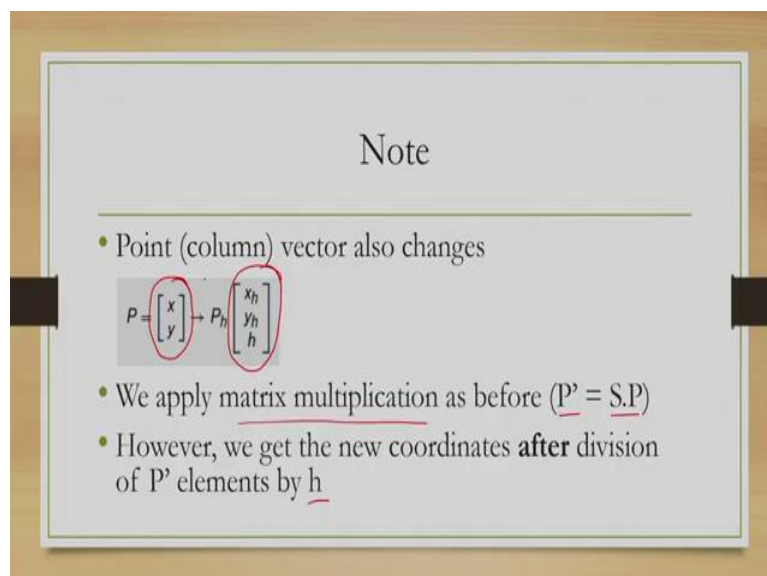
(Refer Slide Time 16:03)



Now, let us see how the basic transformations are represented using homogeneous coordinate matrices. So, translation we can represent using this matrices, rotation we can represent using these matrices where phi is the angle of rotation, scaling can be represented using this matrices and finally, shear can be represented using these matrices. Now, in case of scaling, s_x, s_y represents the scaling factors along x and y direction, in case of shearing sh_x and sh_y represent the shearing factors along x and y direction respectively.

So, here you can see that we managed to the present all transformations, all basic transformations in the form of matrices, although we have to use 3×3 matrices to represent 2 dimensional transformations.

(Refer Slide Time: 17:12)

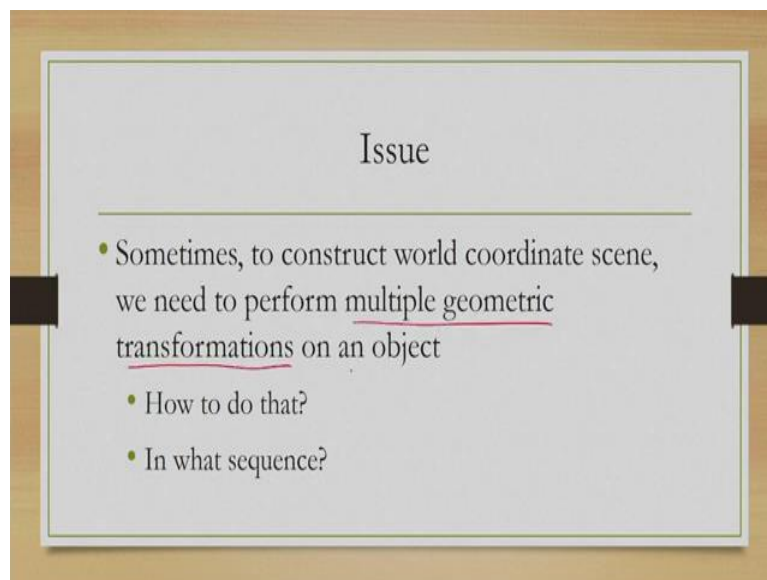


Since, we are using homogeneous coordinate system, so, our point representation also changes. So, earlier we had this representation for each point, now we will be representing each point using a 3 element column vector and the other operations remain the same with minor modification. So, first we apply the matrix multiplication as before to get the new point that is $P' = S.P$. But, after this, what we need to do is divide whatever we got in P' , the x and y values by h to get the original value, this is the general rule for getting back the actual points.

But, in case of geometric transformation as we have already mentioned, h is always 1. So, it really does not matter. But, other transformations will see in subsequent lectures where it matters very much. So far what we have discussed is what are the basic transformations, and how we can represent those transformations, and also how we can use those to transform a point which is by performing a matrix multiplication.

Now, let us try to understand the process of composition of Transformation. When we require composition? If we have to perform transformations that involve more than one basic transformation, then we need to combine them together. Now, the question is how to combine and in which sequence?

(Refer Slide Time: 19:10)

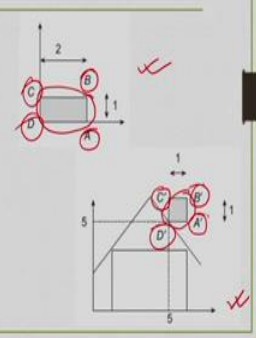


So, when we are performing multiple geometric transformations to construct world coordinates scene, we need to address the issues of how we perform these multiple transformations together and what should be the sequence of transformations to be followed.

(Refer Slide Time: 19:34)

Example

- An object (rectangle) ABCD defined in local coordinate - used to define chimney of the house (in world coordinate)
- Transforming ABCD to A'B'C'D' not possible with single basic transformation - **we need two: scaling and translation**
- How to calculate new vertices?



Let us try to understand this in terms of an example. Here in this figure, look at the top figure here. We see one object denoted by the vertices ABCD with its dimension is given. Now, the bottom figure shows a world coordinate scene in which the same object is placed here which is used to define a chimney let us assume of the house. Now, here you can see that the original vertex A got transformed to A', B got transformed to B', C got transformed to C' and D got transformed to D'. And also, the dimension changed.

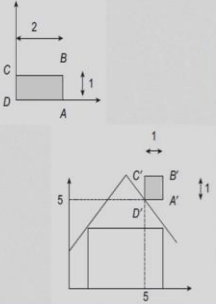
So, earlier dimension actually got reduced along the x direction, although the dimension along the y direction remained the same. So, two things happened here as you can note in this figure, first of all its dimension changed and secondly its position changed. So, earlier it was having one vertex as origin, now it is placed at a different point.

So, two transformations are required; one is scaling and the other one is translation, scaling to reduce the size, translation to reposition it in the world coordinates scene. This much we can understand from the figure, but how to actually apply these transformations that is the question we want to answer so that we get the new vertices.

(Refer Slide Time: 21:42)

Procedure

- Multiply current vertices with the transformation matrix – basic procedure
- However, transformation matrix here is composition of two matrices - scaling matrix and the translation matrix

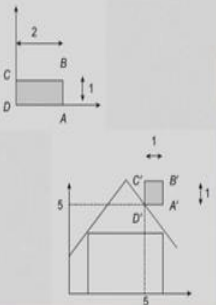


What we know? We know that to get the new vertices we need to multiply the current vertices with a transformation matrix. But, here it is not a basic transformation matrix, it is a composition of two basic transformation matrices and we need to perform that how to do that, how to combine the two matrices?

(Refer Slide Time: 22:13)

Procedure

- First Step – determine basic matrices
 - Object is halved in length while the height remains the same
 - Scaling matrix $S = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$



Let us go step by step. First step, we need to determine the basic matrices that means determine the amount of translation and determine the scaling factors. Note that the object is halved in length while the height is the same that means along the x direction it halved but along y direction it remained the same. So, the scaling matrix would be s_x should be half and s_y will be 1 as shown in this transformation matrix for scaling.

(Refer Slide Time: 22:57)

Procedure

- First Step – determine basic matrices
- Vertex $D(0, 0)$ has now positioned at $D'(5, 5)$ - 5 unit displacements along both horizontal and vertical directions
- Translation matrix $T = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix}$ $t_x = 5$
 $t_y = 5$

Now translation, the second basic transformation that we require. Now, here the vertex D was the origin as you can see here, where it got transferred to? To D' . Now, what is the vertex position of the transformed point that is $(5, 5)$. So, origin got repositioned to $(5, 5)$ that is essentially 5 unit displacement along both horizontal and vertical directions. So, then t_x equal to 5 and t_y equal to 5, so if we use these values in the transformation matrix for translation, then we will get this matrix in this current case. So, earlier we obtained the scaling matrix now, we obtained the translation matrix but our question remains how to combine them?

(Refer Slide Time: 24:15)

Procedure

- Second Step – Obtain composite matrix
- Multiply basic matrices in sequence
- We follow the right-to-left rule in forming sequence

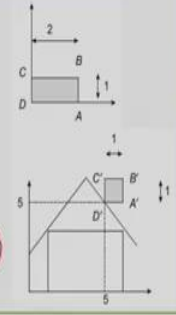
That is the second step composition of the matrices or obtain the composite matrix. What we need to do is to multiply the basic matrices in sequence and this sequencing is very important,

we follow the right to left sequence that is a rule we follow to form the sequence. Now, what this rule tells us?

(Refer Slide Time: 24:50)

Procedure

- Second Step – Obtain composite matrix
 - First transformation applied on object is the rightmost in sequence
 - Next transformation is placed on the left
 - We continue in this way till the last transformation



$T_n \dots T_3 T_2 T_1$

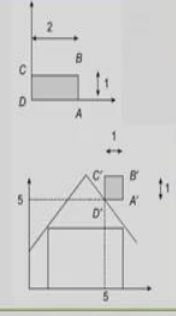
First transformation applied on object is the right most in the sequence, next transformation is lists on the left of this earlier transformation and so on, till we reach the last transformation. So, if we apply the first transformation say T_1 on the object then it should be placed at the rightmost. Now, suppose we require another transformation which is 2, then T_2 will come on the left side of T_1 , if there is one more transformation need to be applied say T_3 then it comes left of T_2 and so on till we reach the final transformation say T_n .

This is the right to left rule; first transformation applied on the object is on the rightmost side followed by other transformations in sequence till the leftmost point where we place the last transformation applied on the object.

(Refer Slide Time: 26:04)

Procedure

- Second Step – Obtain composite matrix

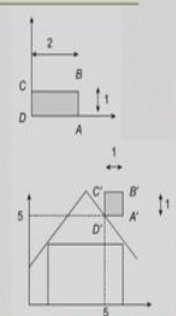
$$M = TS = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & 5 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix}$$


So, in our case, we can form it in this way, first transformation to be applied is scaling followed by translation. So, right to left rule means first S, and on its left side will be T so these two will have multiplication as shown by these 2 matrices and the result will be this matrix. So, this is our composite matrix for that particular transformation.

(Refer Slide Time: 26:39)

Procedure

- Third Step – obtain new coordinate position
 - Multiply current vertices with composite matrix



Once we get the composite matrix after multiplying the current matrices with the composite matrix, we will get the new points.

(Refer Slide Time: 26:55)

Procedure

- Third Step – obtain new coordinate position
- Multiply current vertices with composite matrix

$$A' = MA = \begin{bmatrix} 0.5 & 0.5 \\ 0 & 1.5 \\ 0 & 0.1 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \\ 1 \end{bmatrix}$$

$$B' = MB = \begin{bmatrix} 0.5 & 0.5 \\ 0 & 1.5 \\ 0 & 0.1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \\ 1 \end{bmatrix}$$

$$C' = MC = \begin{bmatrix} 0.5 & 0.5 \\ 0 & 1.5 \\ 0 & 0.1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \\ 1 \end{bmatrix}$$

$$D' = MD = \begin{bmatrix} 0.5 & 0.5 \\ 0 & 1.5 \\ 0 & 0.1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \\ 1 \end{bmatrix}$$

So in our case, this step will lead us to the points as shown here, A' can be derived by multiplying this composite matrix with the corresponding vertex in homogeneous coordinate system to get this final vertex in homogeneous coordinate system and that is true for B' , C' and D' .

(Refer Slide Time: 27:31)

Procedure

- Third Step – obtain new coordinate position
 - Results in homogeneous coordinates – to get Cartesian coordinates, we divide by h (1 for geometric transformations)

$$A' = (6/1, 5/1) = (6, 5),$$

$$B' = (6/1, 6/1) = (6, 6),$$

$$C' = (5/1, 6/1) = (5, 6),$$

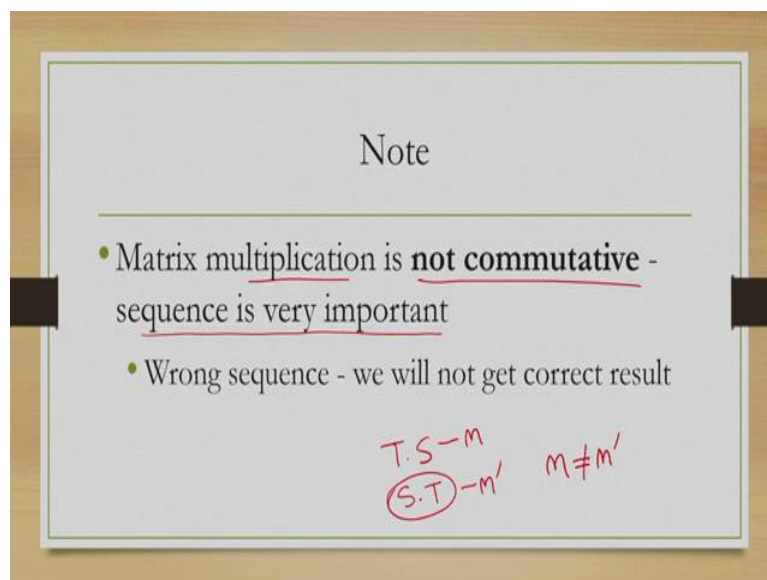
$$D' = (5/1, 5/1) = (5, 5)$$

Now, the last stage of course, is to transform from the homogeneous representation to the actual representation that we do by dividing the x and y values by the homogeneous factor h . Now h in our case, that is the case where we are concerned about geometric transformation, it is 1. So, our final transform points or vertices should be obtained in this way, A' we will get by dividing the x and y values by the homogeneous factors, and similarly for B' , C' , and D' .

So, what we did? We first identified the basic transformations. This was followed by forming the sequence in right to left manner that is we put the transformation that is to be applied on the object at first as the rightmost transformation, then the next transformation to be applied on the object as the transformation left to the earlier transformation and so, on.

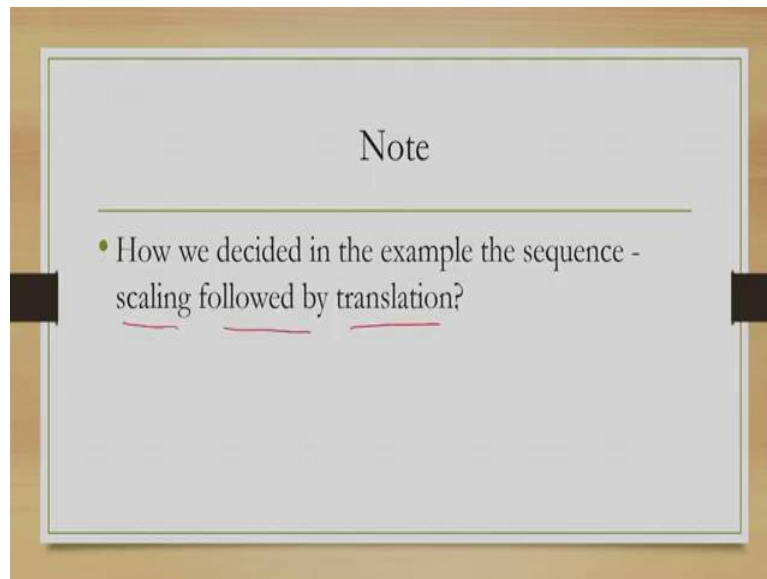
Then we multiply these basic transformation matrices to get the composite transformation matrix. Then, we multiplied the points with this composite transformation matrix to get the transform points in homogeneous coordinate system. Finally, we divided the x and y values of this homogeneous coordinate representation by the homogeneous factor to get back the original transformed point.

(Refer Slide Time: 29:32)



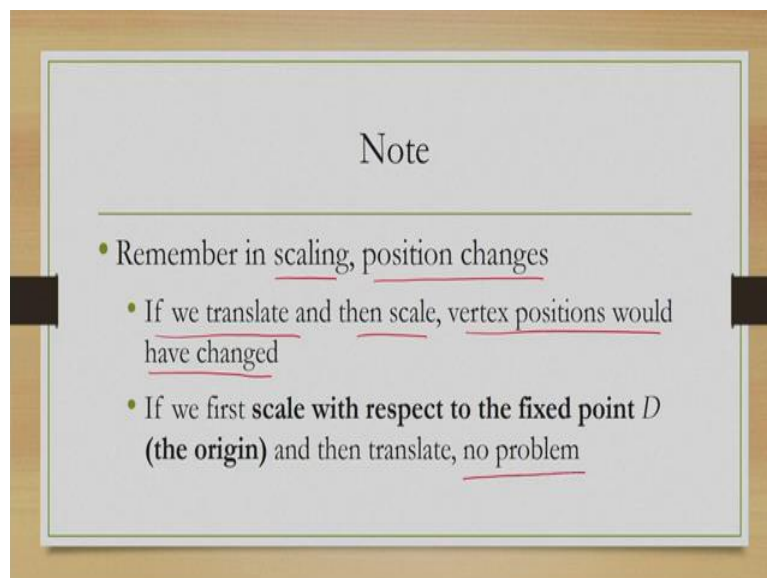
We must remember here that matrix multiplication is not commutative. So, the formation of the sequence is very important. So, earlier we did translation multiplied by scaling following the right to left rule. Now, if we have done it in the other way that is scaling followed by translation, it will lead to a different matrix whereas this gave us M , and since matrix multiplication is not commutative, so we cannot say $M=M'$ so actual $M \neq M'$. So, if we do not create the sequence properly, then our result will be wrong, we may not get the right transformation matrices.

(Refer Slide Time: 30:28)



So, how to decide which sequence to follow. So, earlier we simply said that first we will apply scaling and then we will follow translation, on the basis of what we made that decision. Let us try to understand the example again where we made the decision that scaling should be followed by translation. So, what was there in the example that indicated that this would be the sequence?

(Refer Slide Time: 31:06)

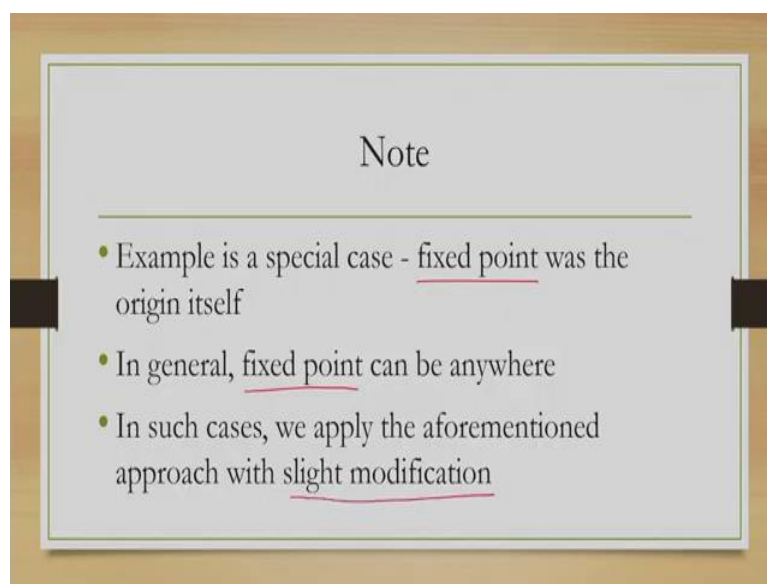


When we discussed scaling, we mentioned one thing that is during scaling the position of the object changes. Now, if we translate first and then scale, then the vertex position might have changed because scaling may lead to change in position. However, if we scale first and then translate, then anyway we are going to reposition it at the right place where we want it. So,

there is no possibility of further position change. So, clearly in this case, we first apply scaling and the associated changes that take place is fine that is followed by translation. If we do that in that sequence, then we do not get any problem so that was the logic behind going for this sequence.

And in general, we follow this logic where if we require multiple basic transformations to be applied, so we keep translation at the end, the last transformation because scaling and shearing are likely to change the position so with translation we try to compensate with that so typically we follow this rule of thumb.

(Refer Slide Time: 32:37)



Now, one thing should be noted here, when we applied scaling, we actually applied it with respect to the origin. So, origin is the fixed point in the example. However, that is not necessarily true. We can have any fixed point located at any coordinate in a coordinate system. So, in such cases, what we do? We apply the approach that we have seen earlier in the example, but with slight modification. So, our approach when we are considering fixed point which is not the origin is slightly different, let us see how it is different.

(Refer Slide Time: 33:33)

Scaling (Arbitrary Fixed Point)

- Suppose we want to scale with respect to fixed point $F(x, y)$
- To determine transformation matrix, we assume sequence of steps

Suppose there is a fixed point F and we want to scale with respect to this fixed point. Now, this is not origin, this is situated at any arbitrary location. Now, to determine the transformation sequence, we assume a sequence of steps. So, if the scaling was with respect to origin then we do not require anything else we simply scale, but if it is not with respect to origin, if it is with respect to some other fixed point which is not the origin then scaling itself involves a sequence of steps, just to perform scaling.

(Refer Slide Time: 34:15)

Sequence

$t_x = -x$
 $t_y = -y$

- Fixed point translated to origin ($-x$ and $-y$ units of displacements in the horizontal and vertical directions, respectively)
- Scaling performed with respect to origin
- Fixed point translated back to its original place

$t_x = x$
 $t_y = y$

What is that sequence? So, first we translate the fixed point to origin that means, we make the translation amount as such T_x is $-x$ and T_y is $-y$; that is the first transformation. Then we perform scaling with respect to origin, this is important. So, our scaling matrix is defined with

respect to origin. So, we first brought or in a conceptual way brought the fixed point to origin then perform scaling and then the fixed point is translated back to its original place, now T_x becomes x and t_y becomes y , reverse translation.

(Refer Slide Time: 35:16)

Sequence

- Composite matrix

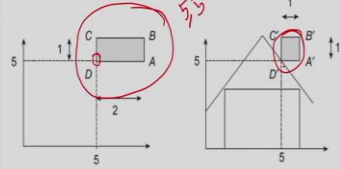
$$M = \underbrace{T(tx = x, ty = y)}_{T_2} \cdot \underbrace{S(sx, sy)}_S \cdot \underbrace{T(tx = -x, ty = -y)}_{T_1}$$

So, how to form the sequence? Will follow the same right to left rule, first translation is the rightmost transformation that is bringing the fixed point to origin, this is followed by scaling so that is the second transformation that is followed by reverse translation that is bringing the point to the original point again that is the leftmost transformation. So, our composite matrix will be a multiplication of these these matrices; T, S and T, let us call it T_1 and T_2 . We multiply to get the composite matrices representing scaling with respect to any point other than origin.

(Refer Slide Time: 36:17)

Sequence

- Example - a modification of earlier example (local coordinate position changed)



And in the same way we can actually perform other basic transformations with respect to any fixed point other than origin. This is one example, which shows the procedure that we just mentioned that is now suppose this original object was defined not with one vertex at origin, but here where we have new vertices and the new point with respect to which the scaling takes place is at T which is (5, 5), and the same object is placed here after scaling and translation. So in this case, translation is not required because it was already at that point and only scaling took place.

(Refer Slide Time: 37:25)

Sequence

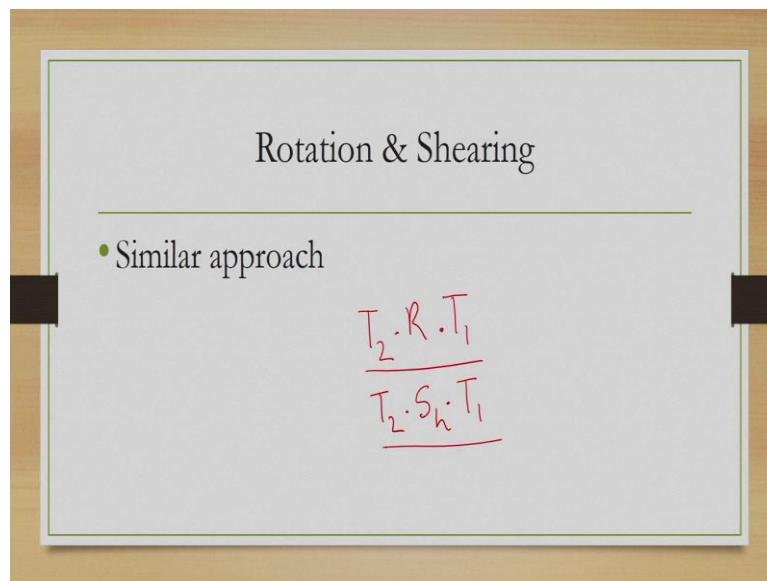
- Now, scaling w.r.t fixed $D(5, 5)$
- Transformation matrix

$$M =$$
$$\underline{T(tx = 5, ty = 5)} \underline{S(sx = 0.5, sy = 1)} \underline{T(tx = -5, ty = -5)}$$

So, if we apply the previous approach that we outlined. So, here we are performing scaling with respect to these fixed point D, and the transformation matrix, the composite

transformation matrix can be found by multiplying these 3 matrices. So, first we translate this fixed point origin so T_x will be -5, T_y will be -5. Then we perform scaling with respect to origin along the x axis that is s_x will be 1/2, s_y will be 1. And then we translate back this point to the original position that is $T_x=5$, $T_y=5$ that is the composite matrix. So, once we get this composite matrix for scaling we apply it to the points to get the transformed points.

(Refer Slide Time: 38:21)



And as I said, we can follow a similar approach with respect to rotation and shearing by first transforming the fixed point with respect to which rotation are shearing had to be performed to the origin then performing the corresponding operation and then translating it back to the original location. So, for rotation, first we will have one translation. This is followed by rotation with respect to origin. This is followed by this will be followed by translating back to the original fixed point location.

For shearing same approach, translation to origin followed by shearing with respect to origin followed by translating back to the original fixed point location. So, this is the composite matrix form for performing any of the basic operation with respect to a fixed point that is not origin.

(Refer Slide Time: 39:35)

General Approach w.r.t. Arbitrary Fixed Point -
Summary

- Composition of basic transformations - same procedure for all (rotation/scaling/shearing)
 - ✓ 1. Translate fixed point to origin R_m
 - ✓ 2. Perform transformation (rotation/scaling/shearing) ✓
 - ✓ 3. Translate fixed point back to its original place ✓

So, to recap, if we are performing the basic operation with respect to origin, then we do not require to do anything else, we simply apply the basic transformation matrix. However, if we are performing the operation with respect to a point which is not the origin, then we perform a composite transformation which involves 3 basic transformations; first one is translation translate the fixed point to origin, second one is the actual transformation that is either scaling, rotation or shearing and the third one is translating back the fixed point to its original place.

And we perform it in this right to left manner so this is the right most, then this will be one on the left of this, the second transformation and the third one will be on the left of this second transformation. So if we put the sequence, first come 1, this will be followed by 2, this will be followed by 3.

(Refer Slide Time 41:07)

Example

- When more than one basic transformation is applied w.r.t. arbitrary fixed point - we apply same process
 - First translate fixed point to origin
 - Perform basic transformations (in sequence)
 - Translate back to original position

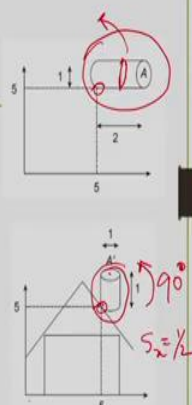
For a better understanding let us go through one more example, which will illustrate the idea further. Now, let us assume we require more than one transformations, so we will apply the same process which we already outlined.

(Refer Slide Time: 41:24)

Example

- **Step 1:** Obtain composite matrix
 - Translate the fixed point (5,5) to origin
 - Scale by 1/2 in X-direction
 - Rotate anticlockwise by 90 degrees
 - Translate the fixed point back to (5,5)

$T(5,5) \cdot R(90^\circ) \cdot S(1/2) \cdot T(-5, -5)$



Consider this object, what are the transformations required to put this object as a chimney in proportion here, as you can see that we need to rotate this object here. So, earlier the surface now becomes here so it is a rotation in counter-clockwise direction positive rotation by 90 degree, and the size also reduces by half along the x direction. So s_x should be 1/2, but all these basic operations took place with respect to this fixed point. So, then how to get the composite matrix?

So, we first translate the fixed point to origin so that is $T(-5, -5)$, then we scale to make it $1/2$ so then S half 1 , along y axis there is no change so, we will keep it 1 . So, then we get objects like this, then we rotate it to get this final one. So, rotate by 90 degree but these 2 operations we performed with respect to origin after translating the fixed point to origin. So, now we have to translate it back so another translation $(5, 5)$. So these matrices together when multiplied will give us the composite matrix.

(Refer Slide Time: 43:29)

Example

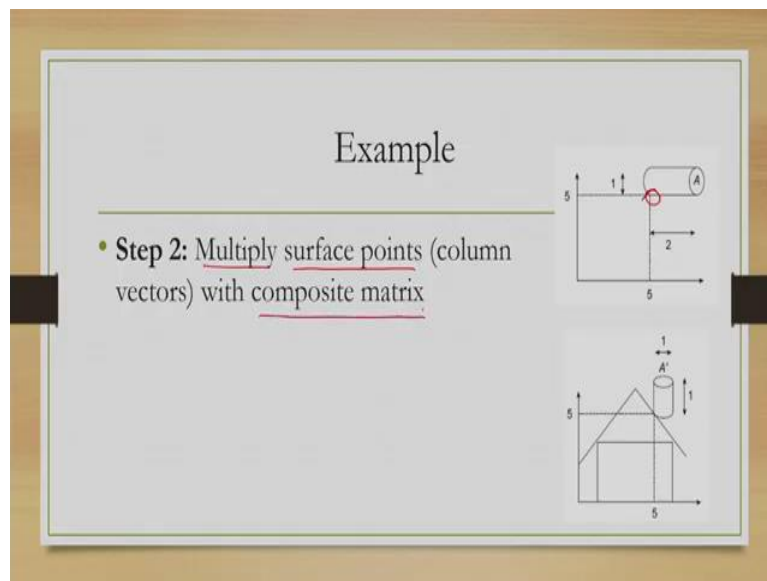
- Step 1: Obtain composite matrix

$$T(t_x = 5, t_y = 5)R(90^\circ)S(s_x = 0.5, s_y = 1)T(t_x = -5, t_y = -5)$$

$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -10 & 10 \\ 0.5 & 0 & 2.5 \\ 0 & 0 & 1 \end{bmatrix}$$

So, it will look something like this. So, if we replace this notations with actual matrices then we will get these four matrices and when we multiply we will get the composite matrix which will look like this. So, this is our way to get a composite matrix when we are trying to perform multiple basic operations with respect to a point which is not the origin.

(Refer Slide Time: 44:08)



And after getting the composite matrix we will follow the same steps that is we will multiply the surface points say these points suppose or any other surface point with the composite matrix to get the transformed point, and that brings us to the end of this discussion. So, before we end it, let us try to recap what we have learned today.

First, we discussed about an alternative representation for basic transformations that is the homogeneous coordinate systems where we represent a 2D point using a 3D coordinate system. And as we have seen, it makes life easier for building modular graphics packages or libraries. So, using this homogeneous form, we can represent all 4 basic transformations using 3 by 3 matrices.

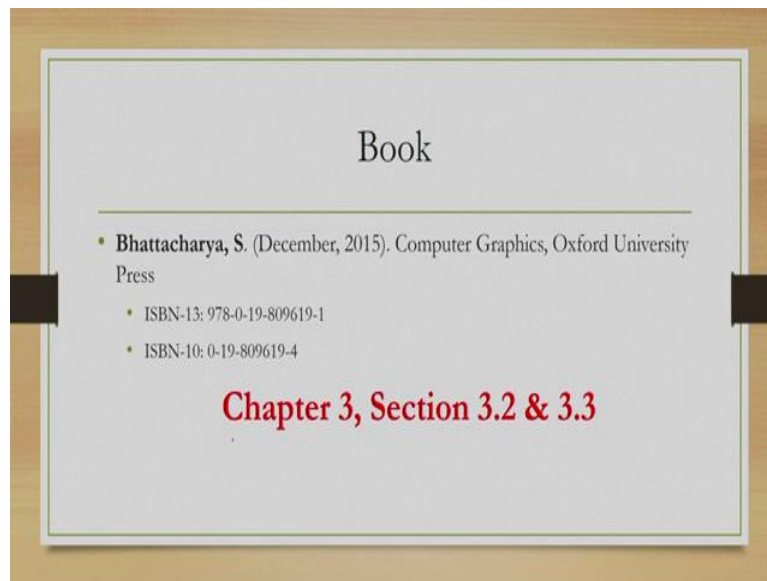
Then what we learned is to form a composite matrix following the right to left rule so first matrix that we apply on the objects should be the right most, next matrix that we apply should be the left to the right most matrix and so on till the last transformation. And we multiply all these matrices together to get the composite matrices. Once we get the composite matrix, we multiply it with the points to get the transformed points in homogeneous coordinate system.

Finally, we divide this x and y values in the homogeneous system by the homogeneous factor to get back the original points. We also learned about how to perform the basic transformations with respect to any point that is not origin. The earlier notations were meant to be performed with respect to origin so when we are given a fixed point and we are supposed to perform the basic transformation with respect to that fixed point, which is not the origin, then we follow a composite matrix approach there we first translate the fixed point to

origin, perform the required transformations basic transformations with respect to origin and translate the point back to its original location.

Following the same right to left rule, we get the composite matrix to represent the basic transformation with respect to any arbitrary point. So far, whatever we have discussed are related to 2D transformations. In the next lecture, we will learn about transformations in 3D.

(Refer Slide Time 47:19)



The topic that I covered today can be found in this book, chapter 3, section 3.2 and 3.3. You may go through these chapters and sections to learn more about these topics. We will meet again in the next lecture. Till then thank you and goodbye.