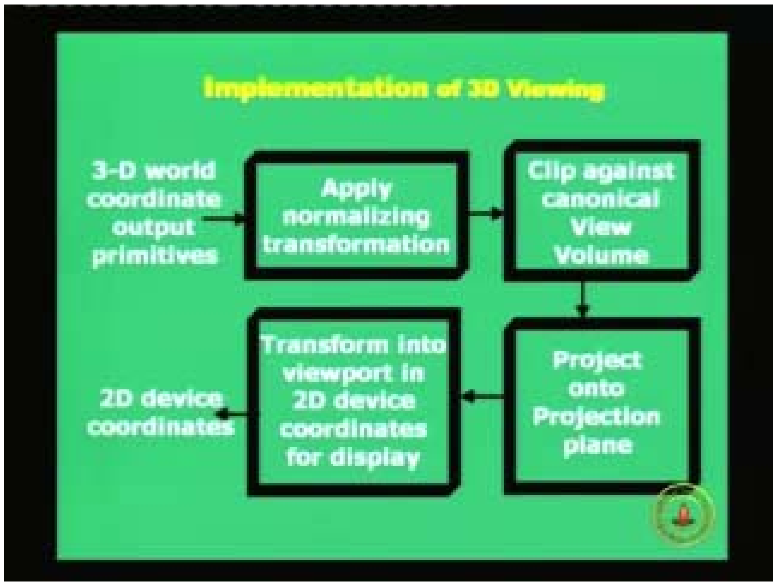
Computer Graphics
Prof. Sukhendu Das
Dept. of Computer Science and Engineering
Indian Institute of Technology, Madras
Lecture - 12
3D Viewing - Projection Transformations and Viewing Pipeline

We continue the discussion on 3D viewing pipeline and projection transformations under the category of 3D viewing. And in the last hour of lecture, you remember that we have discussed basically the viewing pipeline in terms of four different stages of transformations. And in fact within those four we exclusively discussed only one of the stages, the important stage of normalizing transformations.

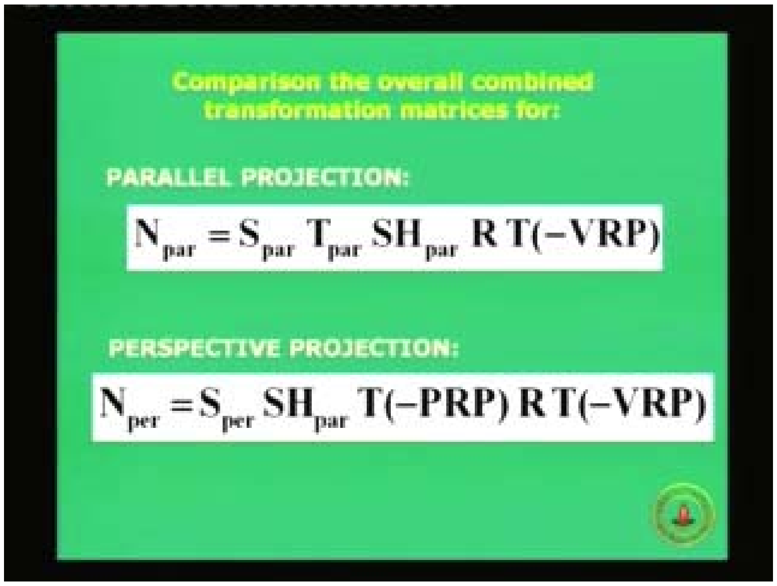
(Refer Slide Time 1:41)


Implementation of 3D Viewing
3-D world coordinate output primitives
Apply normalizing transformation
Clip against canonical View Volume
Project onto Projection plane
Transform into viewport in 2D device coordinates for display
2D device coordinates

So if we look back into what we were doing in the last one hour, the implementation of 3D viewing, the input to that is 3D world coordinate output primitives and which passes through four stages of transformations; the first one is to apply the normalizing transformation, the second is to clip against the canonical view volume. So you know what a canonical view volume is now. And then of course comes the projective transformation projection into projection plane which we will actually take 3D to 2D coordinates and then transform into view port in 2D device coordinates for display. And so output of this entire set of operations or in some sense the viewing sequence of pipeline is the 2D device coordinates.

And if you look, out of these four stages we mainly discussed the first one which is applying normalizing transformation which consisted of five different steps of operations or five different matrices which were doing the operations on the 3D world coordinate output primitives. Now we knew what a canonical view volume is. When we read, understand and discuss about clipping we will see how the clipping operation could be done because when you have a graphics display or in fact seen through your eye you always have a finite amount of display and you need to clip and show out the world which is outside the range of viewing either through the camera or through the eye. And of course in 3D transformations we have also studied projection geometry in generalized matrix so we apply that. And the last step is a two dimensional simple transformation which we will also see today.

(Refer Slide Time 3:29)



Comparison the overall combined transformation matrices for:

PARALLEL PROJECTION:

$$N_{par} = S_{par}\ T_{par}\ SH_{par}\ R\ T(-VRP)$$

PERSPECTIVE PROJECTION:

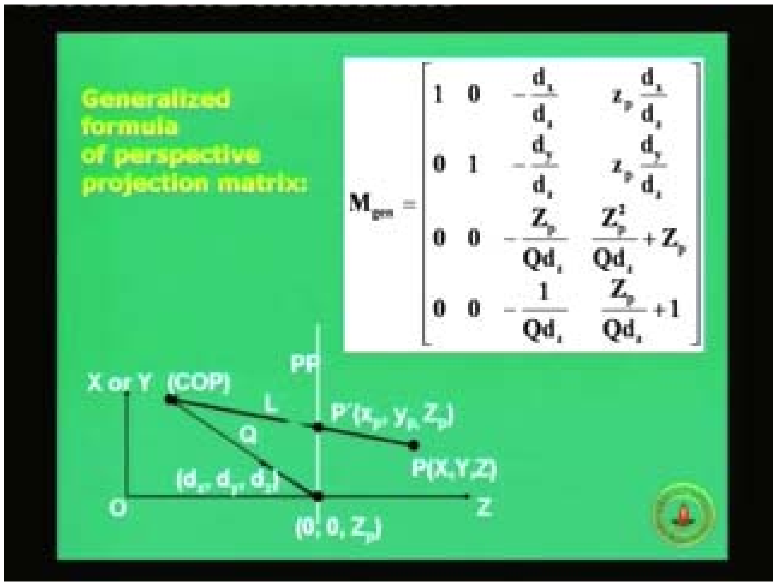
$$N_{per} = S_{per}\ SH_{par}\ T(-PRP)\ R\ T(-VRP)$$

So the first stage if we look which is applying the normalizing transformation and we had seen two different projection transformations; one under the category of parallel orthographic projection, orthographic or non-orthographic in case of parallel and perspective and these were the five different stages for the normalizing transformation in terms of parallel and perspective. So, just to have a comparative study, you remember we had a translation matrix first because the operation is going from left to right. So we are talking about pre-multiplying the matrices or coordinate points by these operations, so translation minus VRP, then a rotation, then again a translation or a shear and then a scale ultimately. That is the combination, the sequence is very important, we cannot jumble up because we have studied under transformations that order is very important.

The sequence of operations has to be kept unaltered to have the correct projections or the transformation done. And out of these five operations the most crucial one was throughout the rotation matrix are where we first transformed a coordinate system, a view reference coordinate system onto a world coordinate system, we knew how to align that using the property or orthogonal property of the rotation matrix and also we discussed about shear and scale, the form of those matrices.

Couple of exercises was given to you during that time to identify the parameters of shear based on the direction of projection vector and scale parameter, parameters of the scale matrix for parallel and perspective as well. Both are of similar types. The shear matrix is optional in the case of parallel projection if it is pure orthographic because the direction of projection in the case of orthographical projection is aligned with the view plane normal so you do not need a shear but if it is not then you typically will need a shear. So this is the first stage of that pipeline, the viewing sequence of five different steps of first stage and then of course after the clipping has been done you can apply the projective transformation.

(Refer Slide Time 5:22)

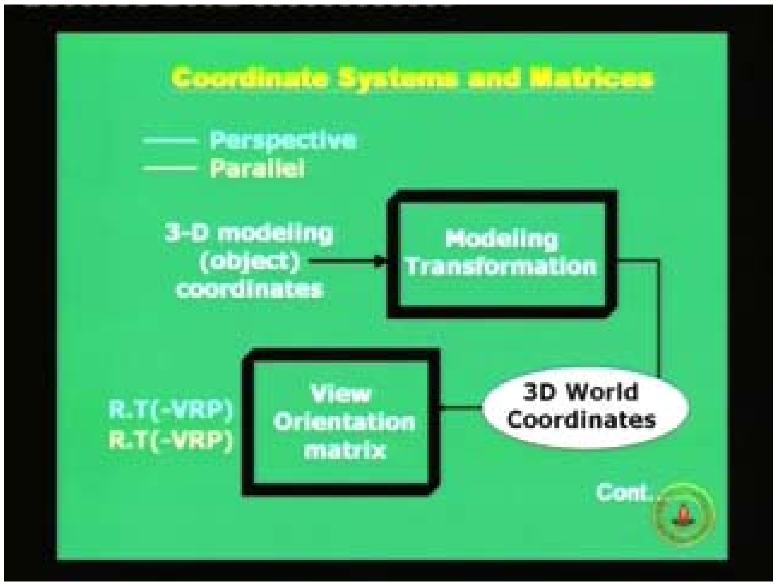


Generalized formula of perspective projection matrix:

$$M_{gen} = \begin{bmatrix} 1 & 0 & -\frac{d_x}{d_z} & z_p \frac{d_x}{d_z} \\ 0 & 1 & -\frac{d_y}{d_z} & z_p \frac{d_y}{d_z} \\ 0 & 0 & -\frac{Z_p}{Qd_z} & \frac{Z_p^2}{Qd_z} + Z_p \\ 0 & 0 & -\frac{1}{Qd_z} & \frac{Z_p}{Qd_z} + 1 \end{bmatrix}$$

This also we have seen a few classes back when we were discussing about three dimensional transformations. And I talked about the generalized expression of transformation matrix in general and it depends on basically mainly three key or four key parameters; one is the Z_p or the distance of the projection plane from the origin and also the direction vector d_x d_y d_z which consists of the direction of projection vector and the length of the point COP or PRP from the point (0 0 Z_p) is q we know that and we have seen the derivation of this matrix a few classes back as well. So first, second, third stage is clear and the fourth stage of course is a simple 2D transformation.

(Refer Slide Time 6:06)



So we look into the pipeline, the pipeline is slightly a different frame work now. We look through a slightly different frame work where we talk of coordinate systems and matrices together for this entire 3D viewing pipeline, a sequence of projective transformations and world coordinate systems moving together and we see how we move from the 3D object coordinates situated somewhere in space and then you have a camera looking through it, your world coordinate system could be somewhere here, right hand or left hand system, your object points could be somewhere at this point, the camera is focusing this way, so what will you see in the graphic screen when a camera is located at an arbitrary point, you will be looking at some arbitrary direction, you have set of objects at this point and the world coordinate system is somewhere here.

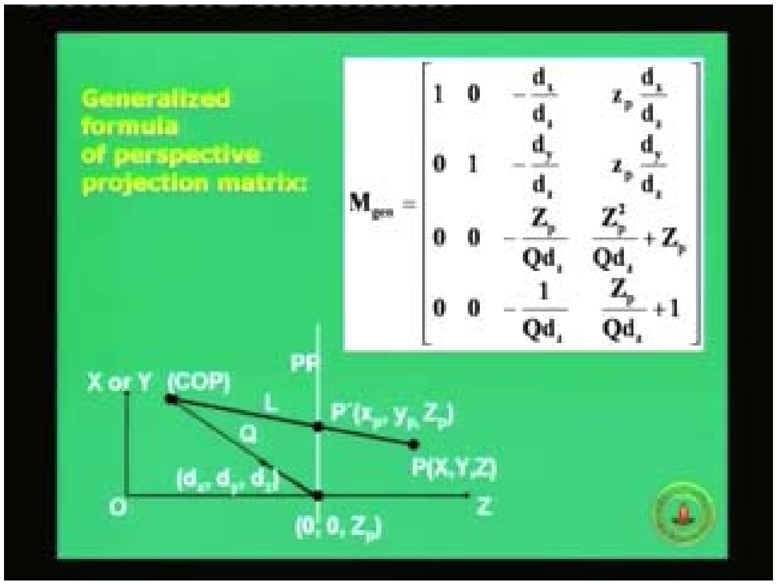
VRC is aligned with, the camera is looking done and the objects are somewhere here. Objects could be outside, away from the camera and will not be in the field of view because your finite zone or a field of view both for the case of parallel as well as the perspective projection geometry is that you need to transform all these points from 3D into the canonical view volume transform the VRC into world coordinate system then do the projection geometry from 3D to 2D and get into the view port. So there is a sequence of operations which are taking place for a generalized scenario. Of course you can have a very simple scenario to start with, if you say my camera is looking in such a manner that it’s pointing towards you towards me and world coordinate system is also align with the camera.

It becomes very easy because most of this transformation matrix will be unnecessary, will not need them to be used in your normalizing transformation and you can simply worry about the camera parameters and the world coordinate points which are to be displayed on the screen. Of course we will see such a simplified model which is only based on the camera. But generalized scenario is very very important because in a generalized scenario most of these graphical software based on PHIGS or OpenGL will ask you to specify the viewing scenario where you have to specify what is world coordinate system, it could be somewhere based on the earth or on the universe or in the sun wherever it is.

Then you have the camera looking at the some point with respect to the world coordinate system, the camera is looking at a particular direction, you have a set of objects somewhere positioned here, a set of points, lines, surfaces, curves, text or whatever the case may be. And you like to have this generalized scenario such that you can handle all sorts of environment. It is not only the simplest case. The simplest case cannot handle generalized scenarios with respect to a camera model which we will see later on and most graphical software.

When we talk of the programming language layer or the high level PHIGS standards it will ask you to specify certain parameters for the viewing pipeline or the projection geometry. So we will come back of course you always keep in mind that there are two main types of projection geometry; one is the perspective geometry and another is the parallel geometry and again we start with the 3D modeling object coordinates. The 3D modeling object coordinates start and they go through a modeling transformation and that is the first stage.

(Refer Slide Time 9:03)



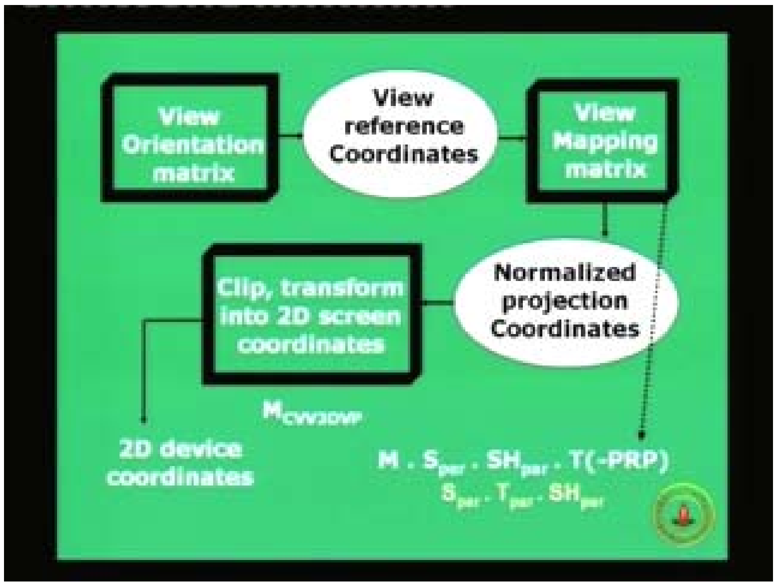
Generalized formula of perspective projection matrix:

$$M_{gen} = \begin{bmatrix} 1 & 0 & -\frac{d_x}{d_z} & z_p\frac{d_x}{d_z} \\ 0 & 1 & -\frac{d_y}{d_z} & z_p\frac{d_y}{d_z} \\ 0 & 0 & -\frac{Z_p}{Qd_z} & \frac{Z_p^2}{Qd_z}+Z_p \\ 0 & 0 & -\frac{1}{Qd_z} & \frac{Z_p}{Qd_z}+1 \end{bmatrix}$$

You have the modeling transformation where basically what you do is, it generates the corresponding 3D world coordinates for the object model because you may have 3D modeling object coordinates with respect to objects center and you need to position them somewhere in the world and that is what we call as the 3D world coordinate generated by the modeling transformation. In this case the modeling transformation talks of positioning those objects, points and lines, curves and surfaces of the objects somewhere in the universe or somewhere in the world. So you get 3D world coordinates and then you probably start to give the transformation which will call the normalizing transformation.

We break it up in two or three different parts; the first part of that is called the view orientation matrix where you can see the expressions that are given on the left hand side corresponding to perspective and parallel, they are basically the same, in fact nothing but the first two steps of the first stage of the normalizing transform that is the translation vector, put the VRP to the origin T of minus VRP and in the rotation matrix which will align view response coordinate system with respect to the world coordinate system which is the origin of the overall system. That is a view orientation matrix which takes the first two steps of the overall five different steps of the normalizing transform. And the flow chart is not complete here it is so big that I split it into two parts, so after the view orientation matrix we move on to the next slide which talks of the remaining stages.

(Refer Slide Time 10:31)

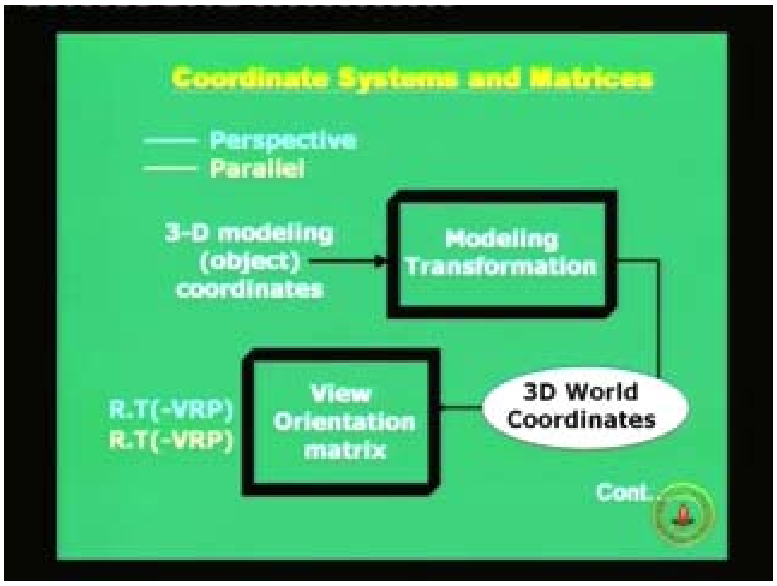

View Orientation matrix
View reference Coordinates
View Mapping matrix
Normalized projection Coordinates
Clip, transform into 2D screen coordinates
$M_{CVV3DVP}$
2D device coordinates
$M . S_{per} . SH_{par} . T(-PRP)$
$S_{par} . T_{par} . SH_{par}$

So the view orientation matrix has the R and T matrices and it will generate what we call as the view reference coordinates. So the dark black boxes talks of the matrices and the electrical areas talks about the coordinates at different stages, the output at different stages. So if I go back, you started with the modeling object coordinates you moved to world coordinates, after that the view orientation matrix will generate what we call as the view reference coordinates and then on the view reference coordinates we apply what is called the view mapping matrix.

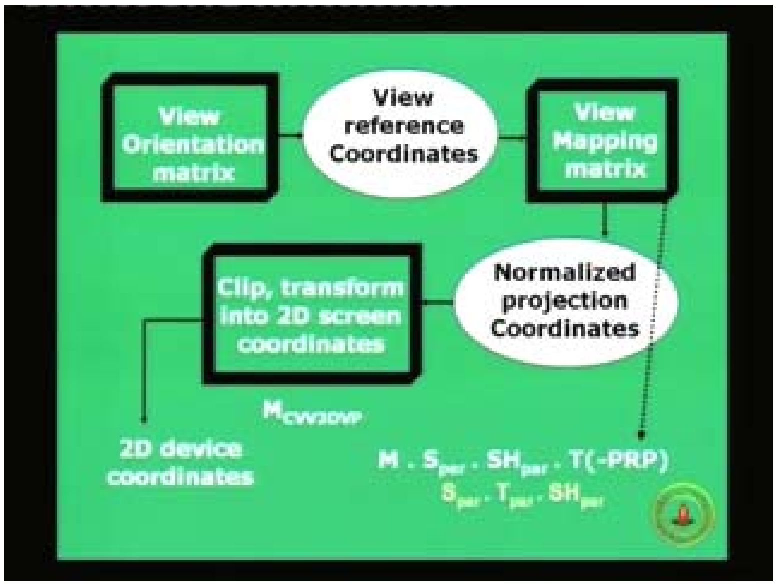
We apply the view mapping matrix which has all the remaining three steps of those five different steps of the first stage of normalizing transform. That is the translation, shear and the scale. So you have the T, you have the Sh and S matrices. Now in terms of perspective and parallel if you see I have tried to minimize the difference by saying that I will have a M matrix which will probably make my Spar is equal to appearing like Spar that the scaling could be little different in case of perspective and parallel transformation because the canonical view volume is different. That is why the extra M has taken care of the same and gives you the difference between perspective and parallel otherwise the shear and translation are typically of the same type.

So after the view mapping matrices is being applied from the view reference coordinates you get the normalize projection coordinates. A view orientation matrix generated the view reference coordinates and the view mapping matrix generated the normalizing projection coordinates. In fact the view orientation matrix along with the view mapping matrix were those five different steps of the normalizing transform which we have discussed in the last hour. Those five steps have been split into two parts; three matrices, three steps and one in two steps and they are called here as the view orientation matrix and view mapping matrix respectively. Expressions are given for the view orientation matrix here in this slide that is the T and the R and the rest of the translation shear and scale is a part of the view mapping matrix and it has generated the normalize projection coordinates.

(Refer Slide Time 12:31)



(Refer Slide Time 12:33)


View Orientation matrix
View reference Coordinates
View Mapping matrix
Normalized projection Coordinates
Clip, transform into 2D screen coordinates
$M_{CVV3DVP}$
2D device coordinates
$M \cdot S_{per} \cdot SH_{par} \cdot T(-PRP)$
$S_{par} \cdot T_{par} \cdot SH_{par}$

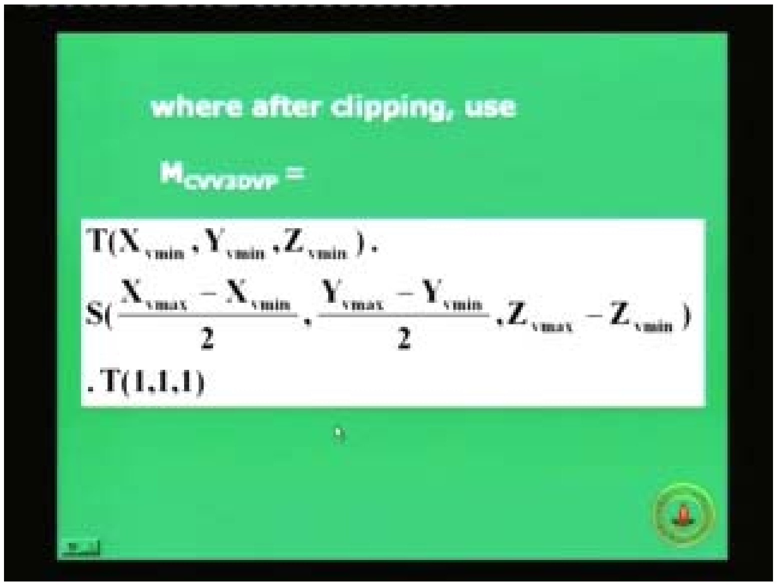
These normalized projection coordinates are still in 3D. You are still in 3D but now what you have done basically is, if the objects were somewhere here, this was the world coordinate system, the camera was looking here and you have actually done the following by transforming the camera, getting in aligned with the world coordinate system so the objects now also follow the same path and you have actually shrunk the entire view volume, the projection geometry into the small canonical unit view volume which is bounded by the six planes.

We know the CVV canonical view volume bounded by six planes; top, bottom, left, right and front and back planes. So now the entire world which we are supposed to see through the camera at some point is now all aligned and kept with the world coordinate system and it is all inside the canonical view volume or inside the CVV but the points are still in 3D, they are still in 3D but in a well bounded and well defined normalized projection geometry and basically you have to apply two or three different steps.

The first step is clipping, first you have to clip with respect to the canonical view volume then apply the projective transform and then do a transformation in 2D. That is all put under one category in the last stage of the viewing pipeline which I will call as clip, transform into 2D screen coordinates, transform into projective from 3D to 2D and then shift it to the 2D coordinates and all those put together except the clipping part if say is all handled by one particular big matrix called Mcvv3dvp.

I will expand what this means but after clipping if this matrix is applied it will basically give you a 2D device coordinates. Now the Mcvv3DVP is basically meaning canonical view volume 3D view port. That means it is the transformation matrix M which transforms the canonical view volume into your 3D view port for projection. So these has to be done after clipping which will transform and then you can apply a projective geometric matrix M general also to get the 2D points from the 3D and give you the 2D device coordinates. Of course we may do a little bit of scaling and translation at the end to get the 2D device coordinates within the view port we are talking about. You need to map everything into a view port which we talked about earlier and these are all coming under the category of MCVV3D view port. That is the terminology used in one book by Foley Van dam, Rogers etc where they will say that we have this matrix called MCVV3DVP which means canonical view volume 3D view port and the expression of that matrix is given here.

(Refer Slide Time 15:19)



where after clipping, use

$M_{CVV3DVP} =$

$$T(X_{vmin}, Y_{vmin}, Z_{vmin}) \cdot S\left(\frac{X_{vmax} - X_{vmin}}{2}, \frac{Y_{vmax} - Y_{vmin}}{2}, Z_{vmax} - Z_{vmin}\right) \cdot T(1,1,1)$$

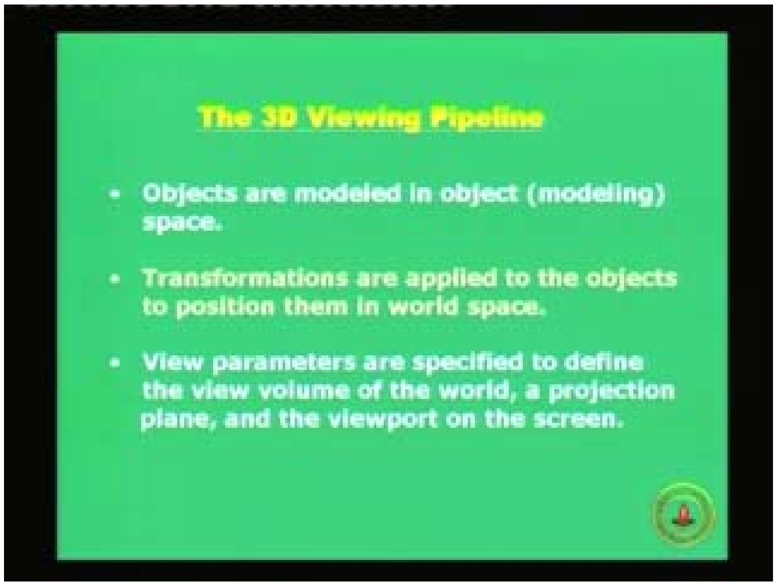
And this matrix you should apply after clipping when the entire 3D canonical view volume has been generated to all points and that you basically apply clipping and then apply this matrix. As you can see from the expression the MCVV3DVP canonical view volume 3D view port consists of a translation scale and a translation. So basically what it is doing is that it is trying to adjust the canonical view volume to your 3D view port such that you can actually apply the M general projection matrix to get the 2D points out of 3D. Remember it is a 3D matrix again the world is still 3D.

Now, after doing all the necessary 3D manipulation you basically apply the projective transform to create your 2D coordinate points from 3D. So clipping, apply MCVV with 3DVP then applying a general to create your 2D coordinates and then of course you have to do a little bit of scaling and translation to position whatever you want to see in a particular view port. This is the screen which you are seeing right now and you will say you do not want to see all the text in the entire screen but would like to select a small portion here or may be some portion here. So that is what a 2D view port is; I will say all I want to display on this particular graphics package to be positioned here or somewhere here. That is what we call as a view port. And what you may need to do is the 2D device coordinates which will be generated with respect to the whole setup. You need to scale it down and then translate it in a position in the view port which you demand or the user asks. So you need to map it into a two dimensional view port. Those expressions are very simple and since you know how to do three dimensional transformations, 2D transformation should be very simple. A simple scaling on a translation brings you from this entire screen down to a small view port because if you have a big screen then you basically do a small scaling operation to bring it down and then you can position it anywhere in the screen basically the view port in which you are viewing the world that you decided to and that is a simple translation operation.

So basically at the end you will require a scaling and translation in 2D which is not there in the expression. You can visualize that it comes at the end after all the projection geometry is done, after all the operations of normalizing transformation of the viewing pipeline and after the clipping and after the MCVV with 3D view port and then after you do the projective geometry M general matrix to transform from a 3D to a screen 2D and then you want position this screen because you may not like the entire display screen to be covered by just one window of graphics screen. You may need have a graphic screen here, you may have multimedia operations, some text running here and some other operations at the bottom.

So obviously you will not ask one particular application to cover the entire screen. So the entire device coordinates where you want to display will be generated by a 2D scaling operation and then positioned by a suitable translation. So that is the viewing pipeline which we have discussed now and again I will repeat these steps to complete the entire 3D viewing pipeline. We can also take a scenario where it involves not only this transformation but of course you need to generate pictures with the help of different algorithms which will come due course of time as we go long in terms of shading, drawing, lines and things like that. And we will discuss in general now the various steps of what we have seen. Now we will see a very elaborate or a very general 3D viewing pipeline. And most of these operations you know, the objects are modeled in object modeling space that is the first step and you know that.

(Refer Slide Time 18:38)



The 3D Viewing Pipeline

- Objects are modeled in object (modeling) space.
- Transformations are applied to the objects to position them in world space.
- View parameters are specified to define the view volume of the world, a projection plane, and the viewport on the screen.

So you can have a object center coordinate system and a object centered or modeled or kept somewhere in the object space and then you have to actually apply transformations which will be applied to the objects to position them with respect to the world space or the view reference coordinate system at least with respect to the camera coordinate system or world space or world coordinate system WC or VRC that is view reference coordinate system, you would have to apply transformations to keep this object.

The second stage says: transformations are applied to the objects to position them somewhere in the world space so that is important. View parameters are specified, we have seen in the earlier class, a table with a few examples of what are the different view parameters you need. A typical set of example you can think of R yes you need to specify the view plane normal, which direction you are looking, you need to specify the view up vector. Let us say you will also need to define other parameters for the viewing system but typically at least you need have the view up vector and view plane normal to specify the world coordinate system and the others. As we look back into the table which was given in the previous class to have examples of different types of perspective and parallel projections and view specifications and that is all necessary for now. The third step as we say the view parameters are specified to define the view volume of the world, a projection plane. Of course you need to tell where is the projection plane and also the view port, the two dimensional view port on the screen.

These are some of the parameters that could be optional but you need to specify at least minimum some of these parameters to specify where you are looking. You can have a world, you can have a virtual reference coordinate system, but you can have a camera looking from this angle onto the object or from this angle or from here. So you need to specify where your camera is and what direction it is looking at. Of course the object is here the camera could be looking somewhere else, you will not able to see anything unless the object is somewhere there. But typically you need to position it so that you want to see what sort of view you would like to have. Would you like to have the top view or the bottom view of an object? If this is an object let us say the side view and the front view etc. So you need to position a camera here to specify the viewing reference coordinate system with respect to the world coordinate system. Here you know where the object is placed. Hence, all these viewing parameters are very very important which talks of a general three dimensional viewing pipeline.

So I again repeat the steps, the first step was the objects were modeled in the object space or object modeling space, second you apply your transformations which are applied to the objects to position them somewhere in the world space and then you specify the view parameters which are specified to define the view volume of the world and the projection plane and the view port on the screen. So these are the first three stages of the viewing pipeline. Then once the transformations are applied of the projection geometry and all normalizing transforms are available the five different steps which we talked about rotation, transactions, shear, scale and all that in 3D you need to clip this object with respect to the view volume now.

Actually sometimes the objects are often clipped with respect to the canonical view volume because the expressions for clipping actually become very simple when we use with respect to the canonical view volume. You remember the expressions of the six planes which bind the canonical view volume both in the case of perspective and parallel, there were six planes and there were six different simple expressions. You can almost visualize it as x equal to plus 1, x equal to minus 1. x equal to y, x equal to minus y, z equal to minus z and z equal to 1 and so on.

So the expressions on those planes are so easy. It helps you to make the clipping algorithms work very fast. Almost you can get a close form, can get a very nice expression to obtain the clipping. Clipping means what? You have a line and you have a bounded volume within which you are seeing a line. So of course what it means is, if this is your view volume and the line is somewhere inside you will not be allowed to see the outside, the view volume or whatever the line is. Remember, a virtual world, a line or a rod can actually go through any particular material.

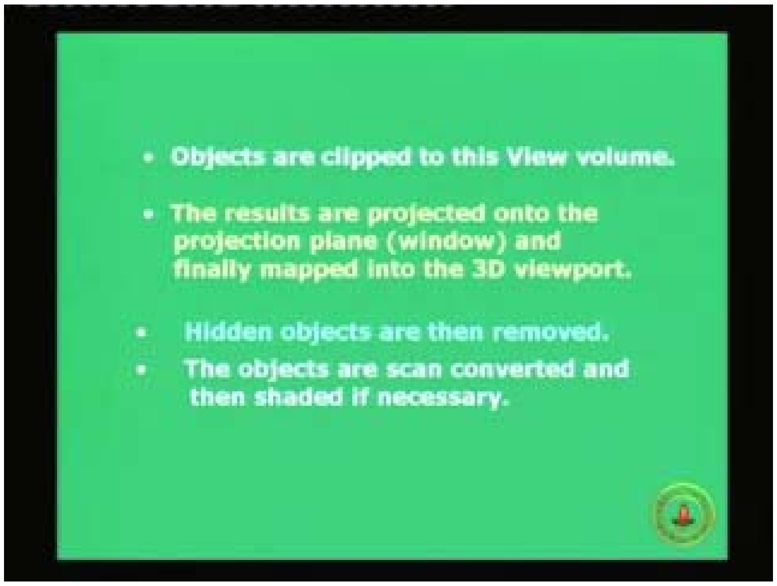
Thus, if you have a virtual volume you will be allowed to see only the inside of it and that is why you require to clip the line and come out with only that section of the line which should be displayed on the graphics screen not the outside. And the expression for the clipping becomes very simple when you operate with a canonical view volume because the canonical view volume has very simple expressions for the planes which bind the canonical view volume. But of course generalized expressions available for clipping in which you can actually clip in any 3D space with any arbitrary view volume and that is also possible. So you either clip with respect to the view volume prescribed by the viewing geometry are you clip with respect to canonical view volume.

So, after the clipping is done the results are projected onto the projection plane window. So this is still in 3D and it is mapped in to the 3D view port. So this is something like, the canonical view volume comes into the picture and it defines a small 3D view port through which you are able to see the entire world where the entire world is squeezed into a canonical view volume and 3D view port.

In this 3D view port you are still in 3D remember. Before going to 2D if it a simple case then you slightly apply the projection geometry and go from 3D to 2D. Otherwise what is done is, in the case of a complicated structure like a house or a unique complex building or environment where there are lots of objects and each object has lot of surfaces in the front, back, side and all that, you need to remove certain hidden objects.

Now this is a special case, there are very good algorithms to do with this particular task of removing hidden objects, hidden surface removal algorithm which you can see. So we will see what is this hidden objects removal? For the time being you can assume that if the camera is looking towards me let us say in the specific example you are not able to see your back. When I am looking at you I am able to see your front or face or may be the side but may not be the back side. So my backside and the backside of yours or all objects on the back of mine are all hidden. They are all hidden then notionally you can visualize that there is no point drawing those on the screen because anyway I will be overlapping these objects and you will be overlapping what is behind you. So you need to eliminate those or remove those hidden surfaces from the screen in terms of drawing it on the screen because otherwise you will spend a lot of time, waste a lot of time in drawing those. So hidden surface removal algorithms are very very essential, very important to reduce the compression complexity of drawing objects. In fact typically what is found that in most of the cases 20 or 30 percent or at the maximum on an average 50 percent of the surfaces what you need to draw for a general three dimensional scenario and that is why it is made possible with the help of hidden surface algorithms where all the hidden surfaces are removed. So you need to have that in the graphics viewing pipeline as well and so the hidden are then removed. And then the hidden objects or the hidden surfaces, lines etc all are removed, now you are only left with objects which are visible to the camera, visible to the user and you want to show those objects rendered them, so the objects are what are called as the scan converted and then shaded if necessary.

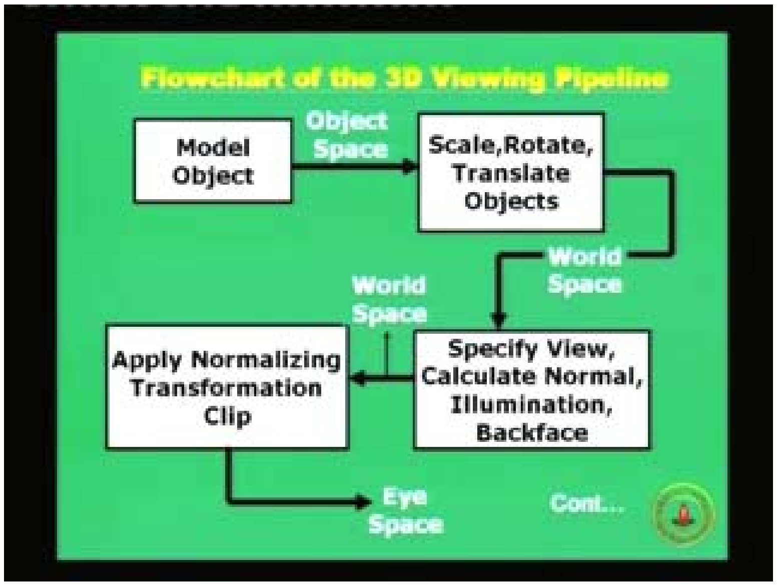
(Refer Slide Time 25:29)



- Objects are clipped to this View volume.
- The results are projected onto the projection plane (window) and finally mapped into the 3D viewport.
- Hidden objects are then removed.
- The objects are scan converted and then shaded if necessary.

That means we are talking of two more types of sections, methods or algorithms which are dealing with scan conversion and shading. So in along with hidden surface removal, after that we have to study about scan conversion and also shading algorithms to find out how to effectively not only shade objects but if possible and necessary create shadows, that is very very important. So we will study about that but we can say that this is under the general category of 3D viewing pipeline to which rendering algorithms are necessary under the category of general viewing pipeline as well.

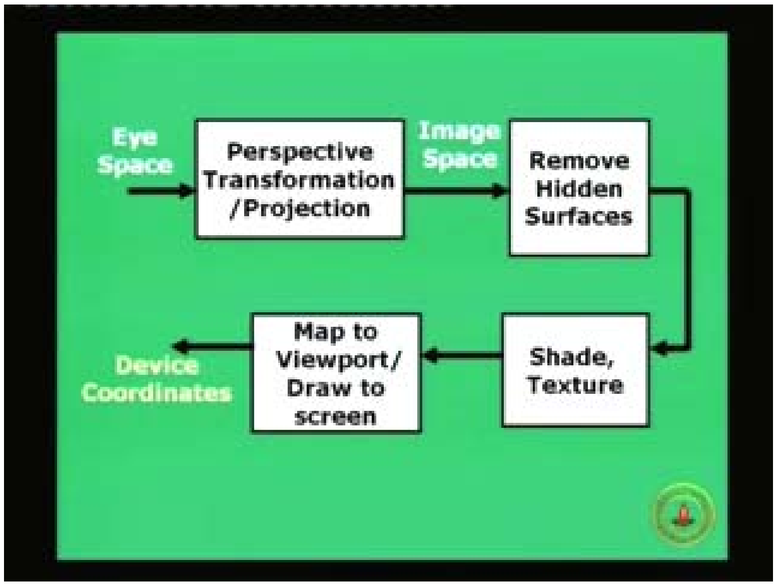
(Refer Slide Time 26:20)


Flowchart of the 3D Viewing Pipeline
Model Object
Object Space
Scale,Rotate, Translate Objects
World Space
Specify View, Calculate Normal, Illumination, Backface
World Space
Apply Normalizing Transformation Clip
Eye Space
Cont...

So well we will again look at it from a flow chart angle when you have the flow chart of the 3D viewing pipeline you have the object model which create the object space and then under the object space you scale, rotate and translate the objects generalize transformation and go to the world space. In the world space you specify the view, calculate the surface normal, calculate illumination, remove the back faces, we have seen few of these terminologies now. We look into all these algorithms as we go along in the sequence of lectures and you still in the world space after you specify the view, calculate the surface normal, calculate the elimination etc. And then you apply the normalize transformation and clipping to create, go into the eye space, go into the canonical view volume 3D view port from the world space.

And in the second step the flow chart is not complete. We are giving the general steps of the 3D viewing pipeline. We gave it in steps, now I get it in the flow chart again and you know where the matrix is to be put after each matrix operation which space you are talking about.

(Refer Slide Time 27:29)



Therefore, you move from world object space to world space, world space to eye space and from eye space is where you apply your projective transformation or projection transformation matrix or parallel projection transformation as the case may be. And there is where we move from 3D to 2D, you move from 3D to 2D and that makes you move from the eye space in coordinates to the image space in 2D and there is where you can apply the hidden surface removal algorithms in image space or you can apply in 3D space as well. And there you provide the shading, the texture and all that is necessary for drawing the objects. Then you map it into the view port and draw the screen.

You talked about these view port and the screen because it may not be necessary and you may not want to draw the entire picture on the entire screen. Today we talked about dynamic environment with several users, several tasks in a multi tasking multi program environment where you may not be having graphics programs but other programs also running in a system and you may also have multiple graphics program, one of them could be just rendering a scene with a house or some building structures and all that. You can have another graphic screen displaying a movie or a multimedia animation and stuff like that, another could be doing something like if you are giving a visualization of a multimedia or a molecular biology or some other visualization of computational fluid dynamics or some other engineering cat game applications let us say.

So you need of course not always the case that the entire screen should be completed with one graphics screen. You define a view port and display. Put the size of the view port depending on the demand of your application you can always scale the view port if necessary and pull it up to cover the entire screen and then bring it back that is all right. But you always need to put in on a view port. So you map the view port and draw the raster on the screen and that generates the device coordinates. As you see on the screen, as you see the last step of the flow chart, when you map into the view port and draw on the screen the corresponding device coordinates are generated and they come through starting from the object space. So I go again, object space, world space, eye space, image space and ultimately the device coordinates where within the view port you want to draw graphic objects is now specified starting from an arbitrary three dimensional world where the objects were given. The objects were here, the camera was in some position, world coordinate is here, somewhere it has gone through all transformations clipping 3D to 2D, hidden surface removal, shading and all that algorithms and finally you have it in a view port. You have in a view port the objects which are drawn which the camera is supposed to see on the objects. You change the objects, the position of the objects or you change the camera position by altering the 3D view specifications, whatever is been drawn in the view port will also change if you change any of the parameters in the corresponding viewing pipeline which we have seen today.

This is the overall 3D viewing pipeline which talks of a very generalized scenario of viewing and we have discussed the first four stages in a very gross framework, in an approximate framework and then we looked into different steps of the 3D viewing pipeline.

Finally we also came back into a very exhaustive flow chart about 3D viewing pipeline. Now what happens, let us say if you do not want to use, let us say you want to create a very simplified scenario where you know that the camera is aligned with respect to the world coordinate system, you have the objects in front of it and you just want to display the system. But of course if you are involved in using standard graphics software, standard graphical packages like PHIGS and OpenGL you have to specify, go to this 3D viewing pipeline to display in most cases, they will ask you to specify parameters.

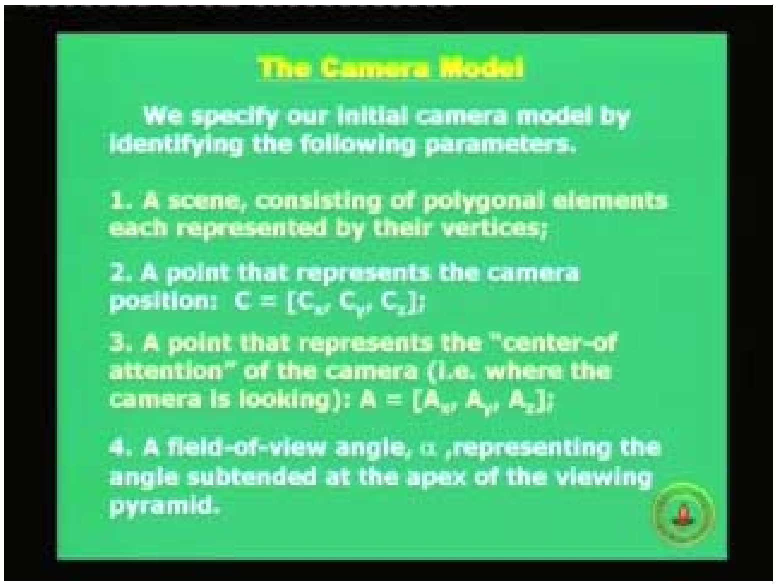
But let us say if you want to write a small program by yourself in OpenGL where it simplifies this entire 3D viewing pipeline and at least the first few stages of normalization and projection geometry of course and then you will say if I can specify what are my camera parameters in terms of in what view I am looking at, what is the focal length or some parameter for the camera in terms of its lengths and screen with something like that and what is the view volume I am looking at, can I not transform a point in 3D into the 2D screen? Yes it is possible.

You may not have to go through the entire viewing pipeline to do that, you can use the 3D viewing pipeline there is absolutely no problem because it is so general that it can take care of all sorts of transformations, all sorts of scenarios which is possible. But let us say you want to create a simple formula one or two steps let us say where you know that I have a camera in a virtual reference coordinate system or VRC view reference coordinate system aligned already with the world and I have an object in front.

Can I pick up the 3D points on the object transform to 2D on the screen?
We will look now, in the last few minutes of the lecture today a camera model which gives a very simple formula to transform a 3D point to the 2D point, this is something similar to the projection geometric combined with the transformations. I am just taking a simple example as a case study where we specify our initial camera model by identifying the following parameters. So you know what is the scene which consists of polygonal elements, each represented by their vertices. So basically you assume for the time being that we have

not talked of solid modeling so for, we assumed that there are 3D vertices specified in the world and it is in front of the camera and the camera is also defined by the following parameters, the camera position may be it could be the origin of the world coordinate systems but you can still have [C_x C_y C_z] as the position of the camera.

(Refer Slide Time 33:34)



The Camera Model

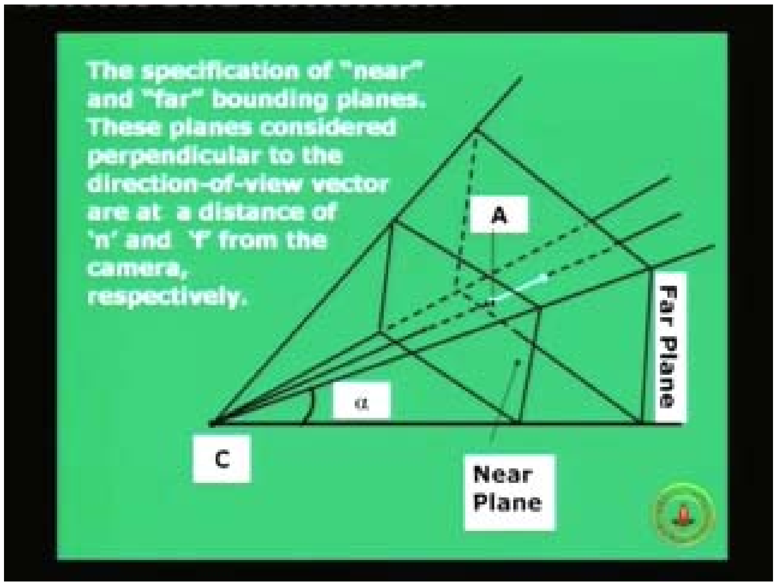
We specify our initial camera model by identifying the following parameters.

1. A scene, consisting of polygonal elements each represented by their vertices;
2. A point that represents the camera position: $C = [C_x, C_y, C_z]$;
3. A point that represents the "center-of attention" of the camera (i.e. where the camera is looking): $A = [A_x, A_y, A_z]$;
4. A field-of-view angle, α ,representing the angle subtended at the apex of the viewing pyramid.

You have a point that represents the center of attention of the camera or the COA which is the center of attention, sometimes also called as the focus of attention FOA. In fact it is true with our eye that we always have a focus of a center of attention to the subject or object we are looking to and you have a center of attention not a focus of attention for a point for this camera which is also specified by a point in 3D as [A_x A_y A_z] the coordinates for the point A. And the last important point is a field of view angle alpha. This is something which gives you the canonical view volume type of a scenario. Now you specify that north east plane but a field of view angle alpha. I will give the diagram by meaning where is this C? Where is this A? What is this alpha which represents the angle subtended at the apex of the viewing pyramid.

The last four points I repeat again which is very important. The field of view angle alpha which represents the angle subtended at the apex of the viewing pyramid. So you need to know to C, you need to know A, you need to know alpha and with these parameters can I take points in 3D which are the vertices of polygon elements and shift to 2D. Just a simple projection using these parameters which are there in points two, three and four C, A and alpha. We have points two, three and four and the parameters C, A and alpha. If they are known C and alpha where C and alpha are not scalar quantities they are 3D vectors and if C, A and alpha, alpha is a scalar quantity, it is angle in degrees or radians. If these three are known can we come up with a simple projection model?

(Refer Slide Time 34:33)


The specification of "near" and "far" bounding planes. These planes considered perpendicular to the direction-of-view vector are at a distance of 'n' and 'f' from the camera, respectively.

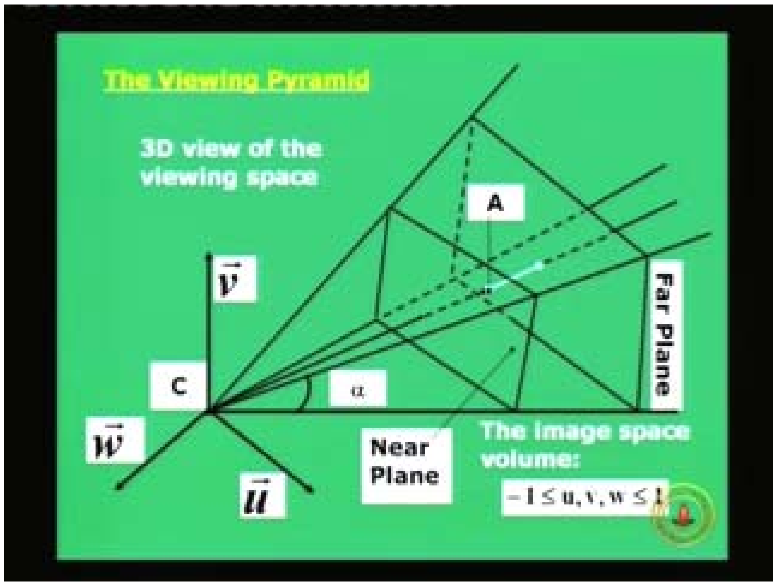
Let us look at this scenario. The specification of C and A is given in this figure already. You have an idea what is this alpha? It is angle subtended at that vertex C of that pyramidal structure. Now this is similar to a canonical view volume for perspective and that is the most common and natural model for what is called as a pineal camera which is used for image processing task as well. When a camera is modeled of course there are other parameters, linear or non-linear parameters to model a camera but it is quit simplified and ok for a computer graphics point of view where I have C, A and alpha three parameters.

And if look into the figure again, in addition of course we have two more surfaces; one is called a near plane and a far plane. Or it is also called the front clipping plane and the back clipping plane. You have already known about FP and BP front clipping plane and back clipping plane you know with respect to those six surfaces where you are clipping your 3D points and surfaces and lines. So, now this pyramid was infinite and I made it finite by putting a near plane and a far plane on the front and back side with respect to the camera which is looking to the center of the pyramid. You have a center line which passes from C to A where A being the focus or center of attention of the camera or the other binding four surfaces of the pyramid which defines these view volumes. So now what I say is, all objects which are inside this finite pyramid, I hope you can visualize this pyramid, this is infinite pyramid but putting the near plane or the front clipping plane or far plane is the back clipping plane if we put these two then you have come up with what we call as a finite pyramid.

So the specification of near and far bounding planes must be given. And these planes are considered perpendicular to the direction of view vector, direction of view vector which is nothing but the vector going from C to A. A is marked here, C is here, A is here. C to A is a direction of projection or it could be even A to C it does not matter but the direction is more important. And these planes, near and far planes are considered perpendicular to the direction of view vector or direction of projection DOP. You remember this? And at the distance n and f from the camera respectively. So the near plane of distance n from C, so the perpendicular distance along this central line which passes from C to A are the direction of view vector or direction of projection vector along that line the distance from C to the near plane is n and from C to the far plane is f.

So n minus f is the distance along the center of line within the finite view pyramid within which all these objects must lie to be viewed. Clipping will do the operation where any object which are outside or in front of the near plane between the camera and the near plane will not be viewed, anything which is beyond the far plane also will not be under the view. Again the viewing pyramid, the 3D viewing space again I have just added the coordinate system U, V and W.

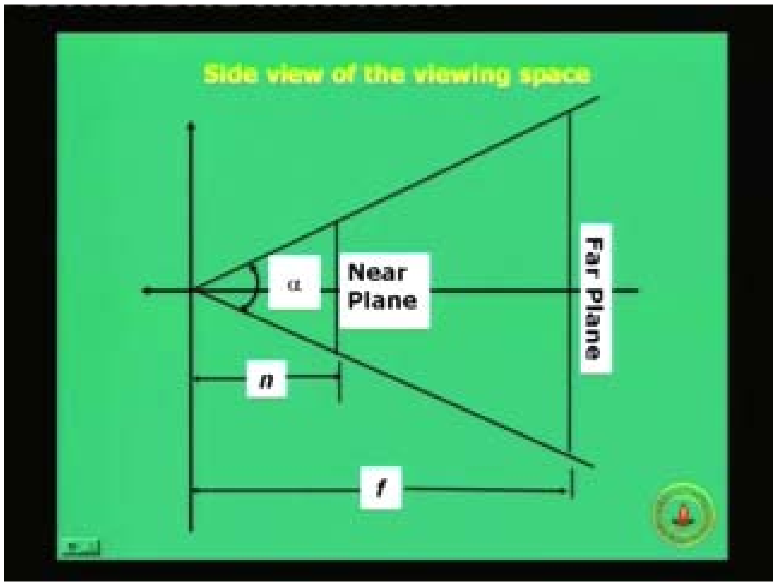
(Refer Slide Time 37:13)



Of course you can align the W with the center of line which is piercing through the center of the finite pyramids C to A. You can align the W with that but it could be very general. You have a near plane and the far plane. And the image space view volume is u v w which I will say although it is 3D ultimately when you say a three dimensional image space with u v w what you are talking basically is a unit cube at u v w, the unit cube or unit pyramid and very simply switch off w and you will get u and v. That is another way, in the case of perspective projection you switch of w you get the device coordinates or view port coordinates or 2D coordinates u and v. But of course in the case of perspective projection in which the image space volume is also a pyramidal structure you need to have a projection matrix. So this is the scenario.

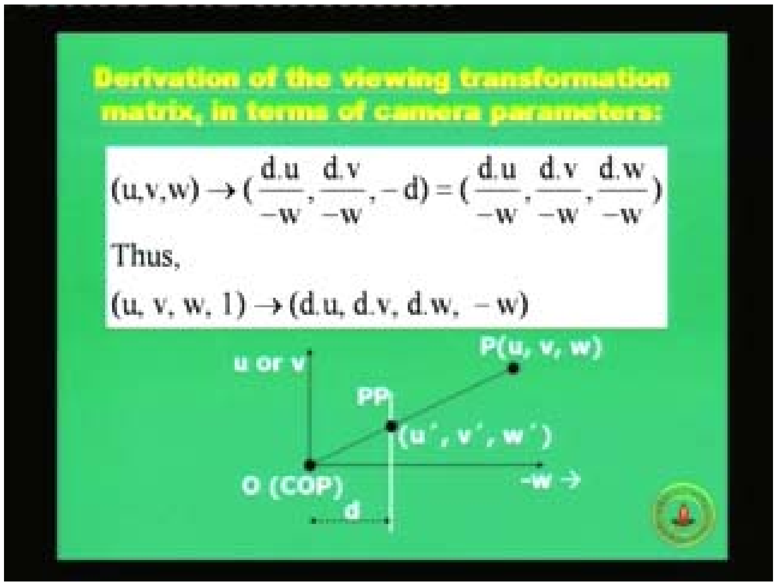
Let us see how we can generate the projection matrix for mapping points in 3D onto 2D. This is the side view of the viewing space, I was talking of its distance n and f from the center C to the near plane and the center is vertex bearing an angle alpha and from the center to the far plane the distance is f.

(Refer Slide Time 38:11)



This is another side view, I hope the picture of the pyramid is getting clear and this was a 3D picture of the finite pyramid obtained from the infinite structure which defines the finite view volume and the image space also defined in the coordinates of the right hand side here in terms of u v w. Along each access you have two minus 1 to plus 1 and this is the two dimensional view. It could be the side view, it could be a front or top view through which you can get this type of triangular structured for the near plane, far plane and the two sides binding along the u v axis. Let us go ahead and derive the viewing transformation matrix in terms of the camera parameter. And as we go ahead you note down the equations and matrices which comes along.

(Refer Slide Time 39:13)



Derivation of the viewing transformation matrix, in terms of camera parameters:

$$(u,v,w) \rightarrow \left(\frac{d.u}{-w}, \frac{d.v}{-w}, -d\right) = \left(\frac{d.u}{-w}, \frac{d.v}{-w}, \frac{d.w}{-w}\right)$$

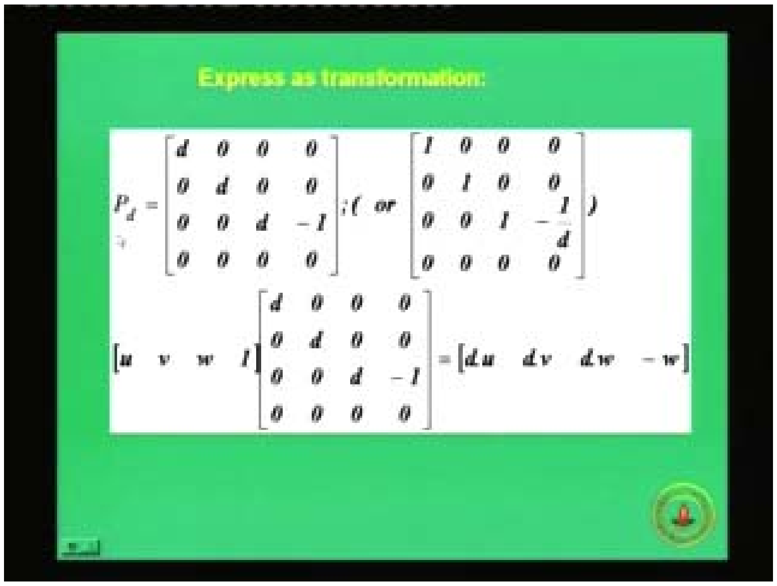
Thus,

$$(u, v, w, 1) \rightarrow (d.u, d.v, d.w, -w)$$

Let us look at this equation and the figure as given. Here you find a point p with coordinates u v and w which is projected on projection plane at a distance D from the COP. The projection plane is lying in the u v plane and the projection point is u prime v prime w prime and you can easily derive u prime v prime w prime using symmetry of triangles. You have done this a little bit earlier as well and they are nothing but d dot u by minus w, d dot v by minus w and minus d respectively. And minus d can easily be found out because the minus w coordinates is minus d.

You can derive these three u prime v prime w prime as given here equal to d dot u by w d dot v by w and minus, just multiplying and dividing by w. So basically the 3D to 2D is nothing but representing a homogenous Cartesian coordinate form u v w and 1 is transformed to d dot u d dot v d dot w and minus w because if you divide the first three coordinates by the fourth one you are going to get your 2D coordinate which are nothing but your u prime v prime and w prime and anyway you know it is equal to minus d. So once you know I can write this in the form of matrix, express this is a transformation matrix Pd I can use any one of these forms as you know.

(Refer Slide Time 40:25)



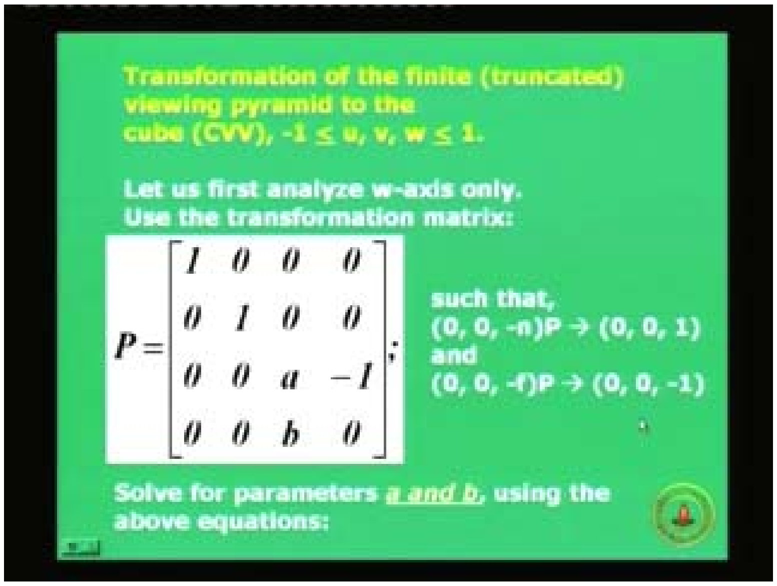
Express as transformation:

$$P_d = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} ;\left(or \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\frac{1}{d} \\ 0 & 0 & 0 & 0 \end{bmatrix} \right)$$

$$\begin{bmatrix} u & v & w & 1 \end{bmatrix} \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} du & dv & dw & -w \end{bmatrix}$$

It is diagonally d and then minus 1 in the third element of the fourth column or you can write this as the identity matrix three 1's and the 1 replaced but the 0 in the fourth row fourth column. But the third element in the fourth column will be minus 1 by d. Now either of the matrices could be used and this is absolutely not a problem because you are working in a homogenous Cartesian coordinate representation. And if you look back in to the equation again you will find that this is a post multiplication. The coordinates in 3D are post multiplied with one of this matrix to get the corresponding u prime v prime w prime as given by d dot u d dot v d dot w and minus w respectively. So if you look back into these equations please note down the matrix form and try both these matrices post multiply a point u v w in 3D space, post multiply by P_d any one of them will do and you will get this. So this projective transformation is fine. But you have to work with the view volume, that is needed to transform the finite truncated viewing pyramid to the canonical view volume of size u v w ranging from minus 1 to plus 1. We will do that with one axis of another but let us first analyze w axis only.

(Refer Slide Time 41:31)



Transformation of the finite (truncated) viewing pyramid to the cube (CVV), $-1 \leq u, v, w \leq 1$.

Let us first analyze w-axis only.
Use the transformation matrix:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & -1 \\ 0 & 0 & b & 0 \end{bmatrix};$$

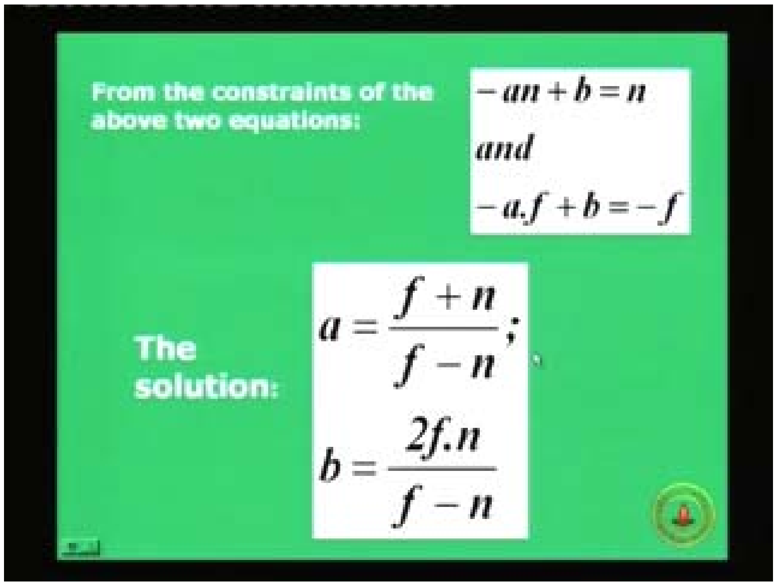
such that,
$(0, 0, -n)P \rightarrow (0, 0, 1)$
and
$(0, 0, -f)P \rightarrow (0, 0, -1)$

Solve for parameters *a and b*, using the above equations:

Let us say if you use a transformation matrix, basically some sort of scaling and translation which you have to do and if we use such a transformation matrix where the unknown elements are let us say a and b, why these two because the a will handle the scaling along the z axis or w axis, a will handle that and b will handle the transformation projection also minus 1 you need to provide that. Basically this helps you to transform a point 0 0 minus n multiplied by p to a point 0 0 1 that is the near point and it will be transformed to the front clipping plane in canonical view volume which is plus 1 and the far clipping plane 0 0 minus f in 3D will be transformed to 0 0 minus 1 that is the back clipping plane.

So if you know that P is able to give these transformations from 3D to 2D basically or in fact in 3D itself then you basically solve for a and b using the above equation. That means you substitute P in these two and you basically get expressions like this.

(Refer Slide Time 42:36)



From the constraints of the above two equations:

$$-an + b = n$$
and
$$-a.f + b = -f$$

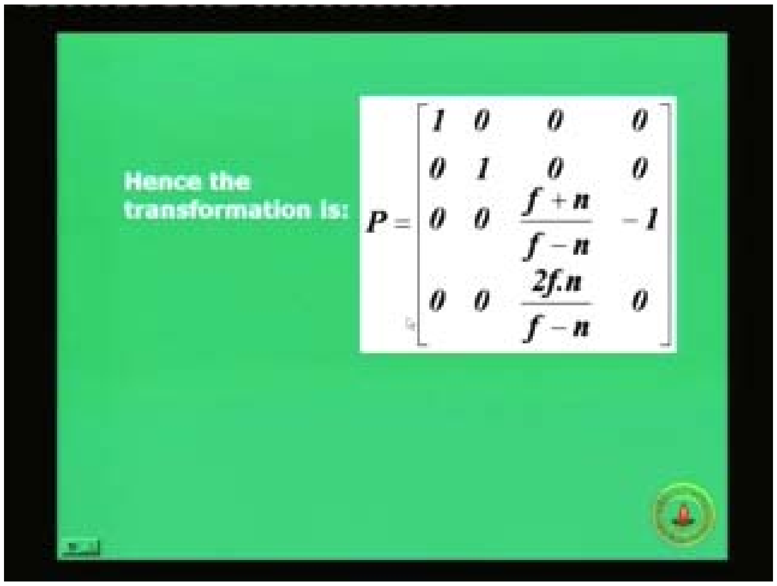
The solution:

$$a = \frac{f+n}{f-n};$$

$$b = \frac{2f.n}{f-n}$$

These are the two constraints you get for a and b. Remember n and f are known to you and since you have two equations with two unknowns a and b you can solve for a and b straight away using these two equations and to get very simple nice forms as the suggestion for a and b based on f and n. Remember f and n are also camera parameters in some sense because they are the distance to the near plane and near front clipping plane and back clipping plane respectively the f. So let us look back into the matrix.

(Refer Slide Time 43:15)

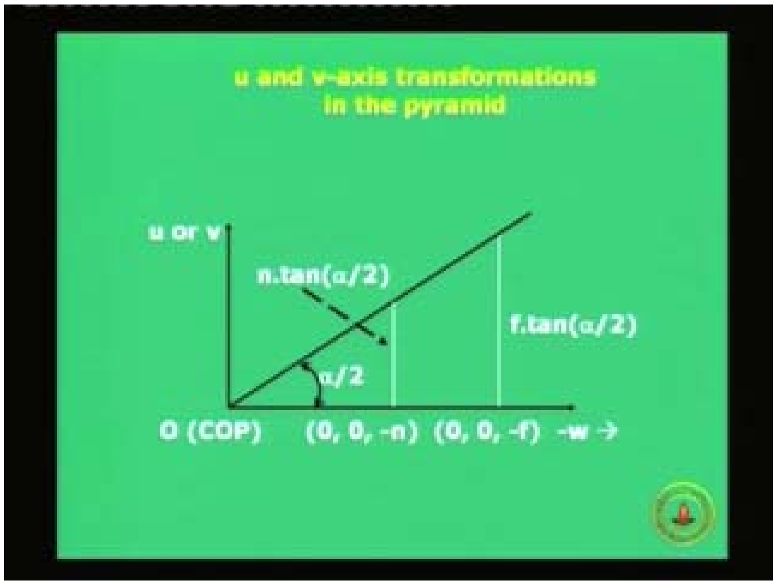


Hence the transformation is:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & -1 \\ 0 & 0 & \frac{2f.n}{f-n} & 0 \end{bmatrix}$$

This is the transformation matrix. Now this transformation matrix will handle the z transformation from the finite view parameter to a canonical view volume. What about u and v with respect to x and y, well we also have to go to any one of these axis and the u v axis transformations of the pyramid are understood by taking the side view again.

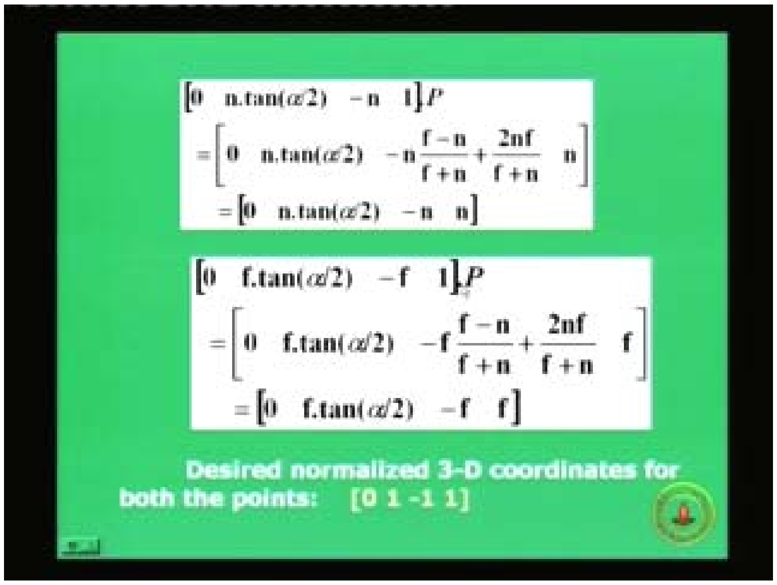
(Refer Slide Time 43:20)



u and v-axis transformations in the pyramid

If I take the pyramidal side view you can see these two white lines are of length. If this is 0 to n and then 0 to f with alpha by 2 it is because the entire angle is alpha. We are taking half of this pyramid which is alpha by two the angle at the vertex. So these two vertical lines will of length n. tan alpha by two and f alpha by 2. It has to be like that and what basically happens is now the points of these two at the edge of the pyramids will be having these 3D coordinates 0 n. Tan alpha by 2 and minus n and this point will also have 0 either u or v we are talking about and f. Tan alpha by 2 which is the vertical along u or v and z part is minus f. So these have to be transformed to correspondingly the coordinate 0 1, minus 1 and 1, that is desired normalized 3D coordinates for both these points. So if you employ the matrix P which you have seen, then you get this transformation from this point n tan alpha by 2 minus n to a transformation matrix like this and similarly f tan alpha by 2.

(Refer Slide Time 44:09)



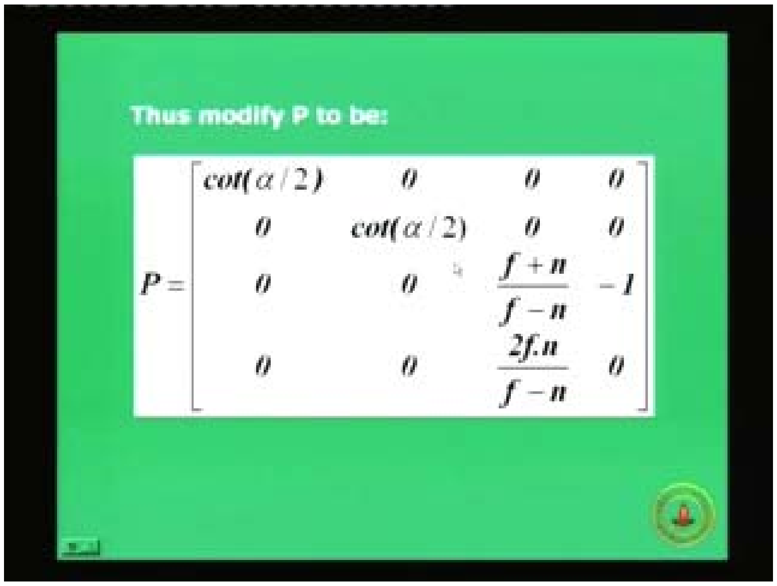
$$[0 \quad n.\tan(\alpha/2) \quad -n \quad 1]P$$
$$= \left[0 \quad n.\tan(\alpha/2) \quad -n\frac{f-n}{f+n}+\frac{2nf}{f+n} \quad n\right]$$
$$= [0 \quad n.\tan(\alpha/2) \quad -n \quad n]$$

$$[0 \quad f.\tan(\alpha/2) \quad -f \quad 1]P$$
$$= \left[0 \quad f.\tan(\alpha/2) \quad -f\frac{f-n}{f+n}+\frac{2nf}{f+n} \quad f\right]$$
$$= [0 \quad f.\tan(\alpha/2) \quad -f \quad f]$$

Desired normalized 3-D coordinates for both the points: [0 1 -1 1]

You please note down the equations and try it yourself multiplied by P and you get this point. Now if you see here, you basically need to have a transformation matrix P in such a manner that this becomes 0 1 minus 1 and plus 1 and similarly this also becomes 0 1 minus 1 and that can be done by modifying P.

(Refer Slide Time 44:47)

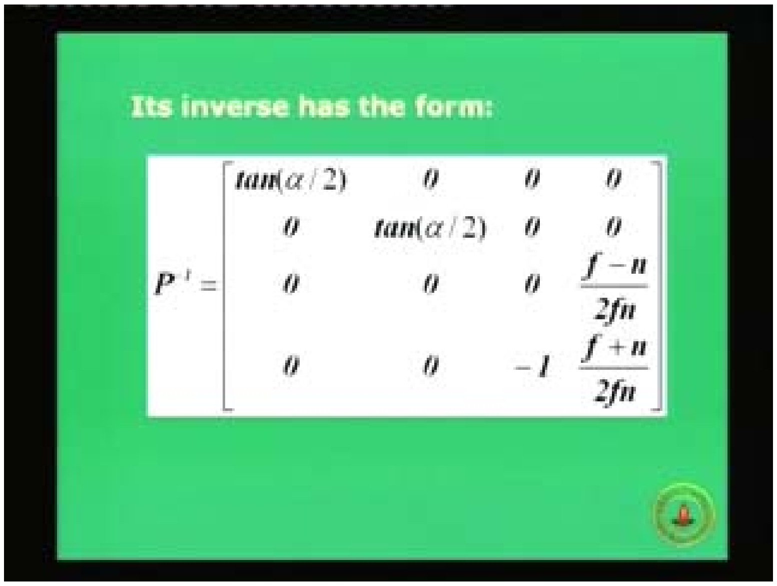


Thus modify P to be:

$$P = \begin{bmatrix} \cot(\alpha/2) & 0 & 0 & 0 \\ 0 & \cot(\alpha/2) & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & -1 \\ 0 & 0 & \frac{2f.n}{f-n} & 0 \end{bmatrix}$$

This can be done by modifying P because the P which we derive only with Z axis has these two terms f plus that is the third column has these two terms. If you introduce these two terms in the diagonal element cot alpha by 2 it will help you basically to get these two terms vanished and this will become 0 f minus f and 0 and minus 1 and the transformation will take place basically. And of course you can note down this new form of P using the matrix and I hope you are able to note it down. The inverse of p I leave it as an exercise for you to find out, a little bit of matrix manipulation.

(Refer Slide Time 45:20)

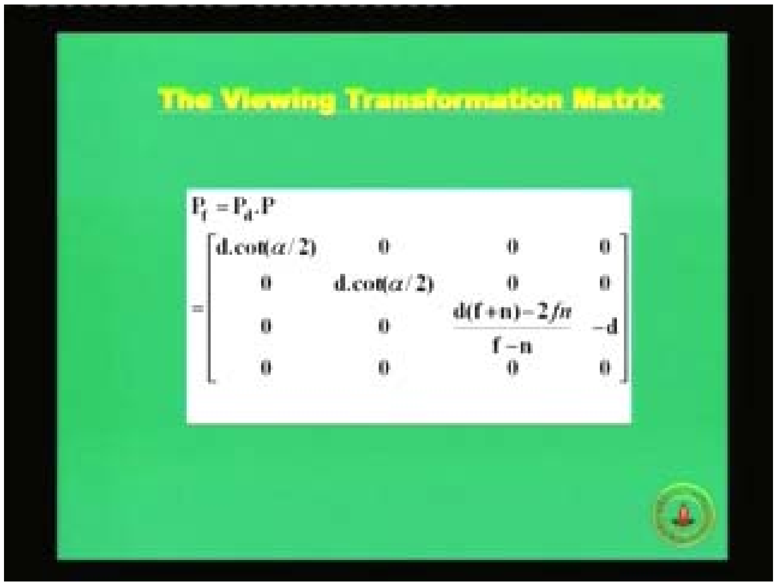


Its inverse has the form:

$$P^{-1} = \begin{bmatrix} \tan(\alpha/2) & 0 & 0 & 0 \\ 0 & \tan(\alpha/2) & 0 & 0 \\ 0 & 0 & 0 & \frac{f-n}{2fn} \\ 0 & 0 & -1 & \frac{f+n}{2fn} \end{bmatrix}$$

You can try out the inverse of p which should be this. You can note it down for the time being and practice at home, 4/4 matrix not a very difficult one. This is the matrix p and this is the inverse of the matrix which is given as cot alpha becomes tan alpha by 2. Diagonal terms are very straight forward but the other matrices will also change place. So the overall viewing transformation matrix the product of two matrix is P_d and P.

(Refer Slide Time 45:43)



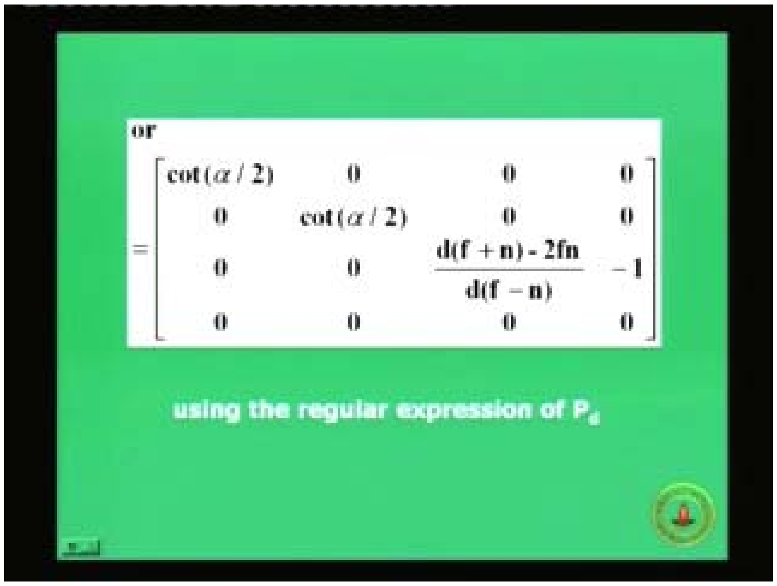
The Viewing Transformation Matrix

$$P_t = P_d . P$$

$$= \begin{bmatrix} d.\cot(\alpha/2) & 0 & 0 & 0 \\ 0 & d.\cot(\alpha/2) & 0 & 0 \\ 0 & 0 & \dfrac{d(f+n)-2fn}{f-n} & -d \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

And remember the P_d, you can take any one of these two forms d d, d minus 1 or 1 1, 1 minus d and post multiplied by the P matrix which we have just seen. Remember P is this matrix and the P_d had diagonal elements d and the first three diagonals and it had a minus 1. So if you multiply P_d with that P the product of the matrix is what you will get as these. One of the forms of P_d gives you this, the other form of the P_d is what you also basically get in this particular form.

(Refer Slide Time 46:25)



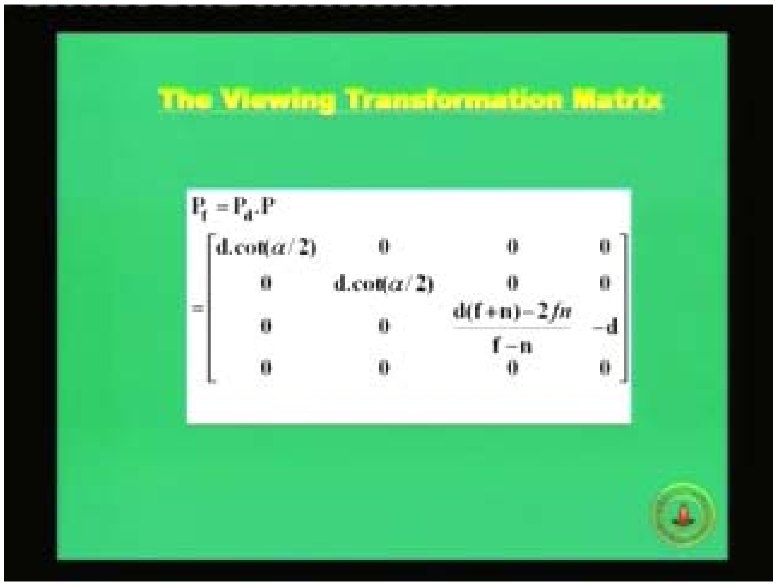
or

$$= \begin{bmatrix} \cot(\alpha/2) & 0 & 0 & 0 \\ 0 & \cot(\alpha/2) & 0 & 0 \\ 0 & 0 & \dfrac{d(f+n)-2fn}{d(f-n)} & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

using the regular expression of P_d

And please note down these forms, I am displaying it for you. This is the form of the viewing transformation matrix.

(Refer Slide Time 46:31)



The Viewing Transformation Matrix

$$P_t = P_d.P$$

$$= \begin{bmatrix} d.\cot(\alpha/2) & 0 & 0 & 0 \\ 0 & d.\cot(\alpha/2) & 0 & 0 \\ 0 & 0 & \dfrac{d(f+n)-2fn}{f-n} & -d \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

And if you look into the parameters, all the parameters are based on the camera model. If you take d what is the d? d is something you can visualize it to be the focal length of lens, camera parameter where basically the distance of the projection plane from the origin and that is the d. What is the alpha? That is the angle subtended at the vertex of the pyramid.

We define and see a view volume as how much you are looking both sideways and from top to bottom. And what are the infinite? How much you are looking? You cannot look beyond a certain point and something very close also you cannot visualize properly which is very realistic. So you have a front clipping plane distance f and a back clipping distance n f, you have the d where you have the projection plane, you have the alpha. So all these parameters are there in the projection equation as you see. This is one form of the projection transformation matrix P_f which is P_d multiplied by P.

If you take the other form of the matrix Pd which is the identity matrix type 1 1 1 and minus by d you typically have the nature of the expression of P_d in this form. So kindly note down these two matrices and I hope you start from scratch and derive this matrix expressions yourself. I hope you are able to copy down the expression yourself as given in the slide. Please copy down these expressions and try to derive these from the figure which I have given.

I will probably jump one or two stages here, steps basically so that I leave something in this exercise for you to write down matrices from the expression directly because you have gone to the 2D transformation, gone to the 3D transformation, gone through projective transformations, gone through viewing pipeline and now given equations you should be able to write in matrix or matrix back to equations, do the manipulation and again the most important thing is to visualize what these transformations mean because these transformations are nothing but basically implementing a graphical scenario from you for the user for the viewer and you must know why these transformations are very important.

You will have the algorithms in graphics but the mathematics part is also very very important which you cannot forget and along with that the geometrical interpretation which you must have in mind. So we will wind up this lecture today on 3D viewing pipeline. We have in fact discussed about general transformations of viewing pipeline which consists of four broad stages applying normalize transformation, clipping to canonical view volume, use the projection matrix and then actually scale and translate in to view port. These are the four broad stages.

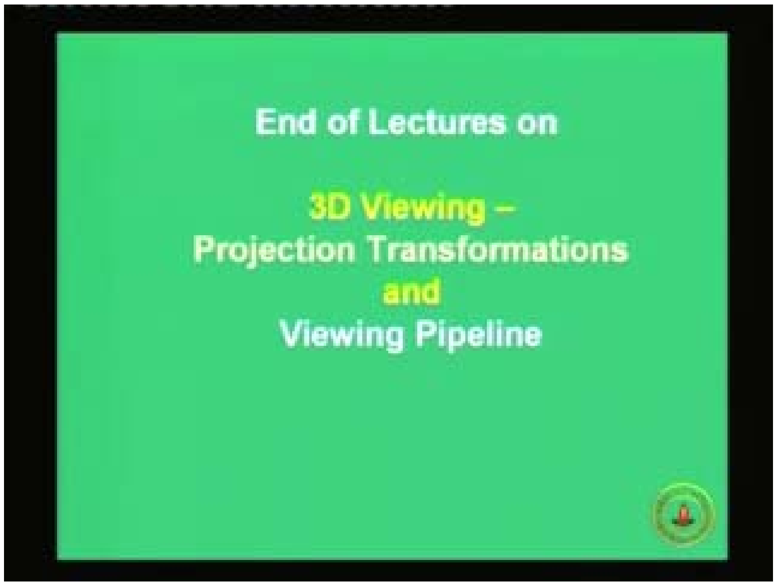
We discussed the second normalize in the last lecture the normalizing transformation which have five different steps such as rotation, translation, shear, scale and finally a scale again at the end. And those were the five different steps for the normalizing transformation which helped you to bring the view reference coordinate and the entire objects, align them with respect to the world coordinate system for proper viewing then of course you need to scale it down the canonical view volume or 3D view port clip. You have the clipping done which we will discuss later on. And then of course do the perspective projection from 3D. That was the general viewing pipeline and then of course we went through an example case study where, what about for a simplified situation? Let us say only camera model parameters are given is it possible to generate a view? Yes it is possible, you can by pass if not necessary the entire general 3D viewing pipeline and if the camera parameters are given.

What are the camera parameters given? The d which is the distance of the projection plane from the center of the camera, the angle subtended at the vertex alpha which defines the pyramidal view volume and n and f the distance of the front and back clipping plane. If these parameters are given you can also define, come up with a projection transformation matrix which will take you from 3D to 2D.

We have seen the last two expressions in the two different forms, almost similar expressions which take you from 3D to 2D and without in fact going to the extensive pipeline. But you must also note that the extended viewing pipeline which helps you to understand the generalize scenario which all graphic standards and packages mostly follow. So if you are using standard graphics packages and software and graphical standards let us say the PHIGS is an example which we have been following in this course.

But extensively we have not come up with the specific examples but will devout one hour at least for certain examples of PHIGS and OpenGL together. We may need to follow such graphic standards. You have to follow a 3D viewing pipeline. But of course if you want a small software interface which is based on simple camera parameters or camera models, what you can basically do is use such a transformation matrix and get the transformation done from 3D to 2D which includes the canonic concept or even canonical view volumes and helps you to even clip if necessary. So that is the last slide and we wind up this lecture.

(Refer Slide Time 51:31)



End of Lectures on

3D Viewing –
Projection Transformations
and
Viewing Pipeline

End of the lectures on 3D viewing projection transforms and viewing pipeline. Thank you very much.