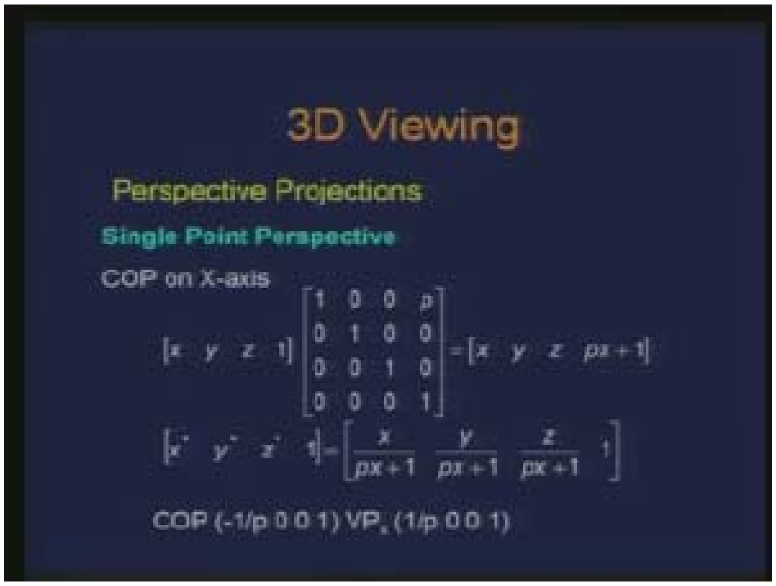
Introduction to Computer Graphics

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

Lecture - 9
3D Viewing (Contd…..)

So we have been talking about 3D viewing. Let me do some of the recapitulation of what we did last time.

(Refer Slide Time: 1:09)



3D Viewing

Perspective Projections

Single Point Perspective

COP on X-axis

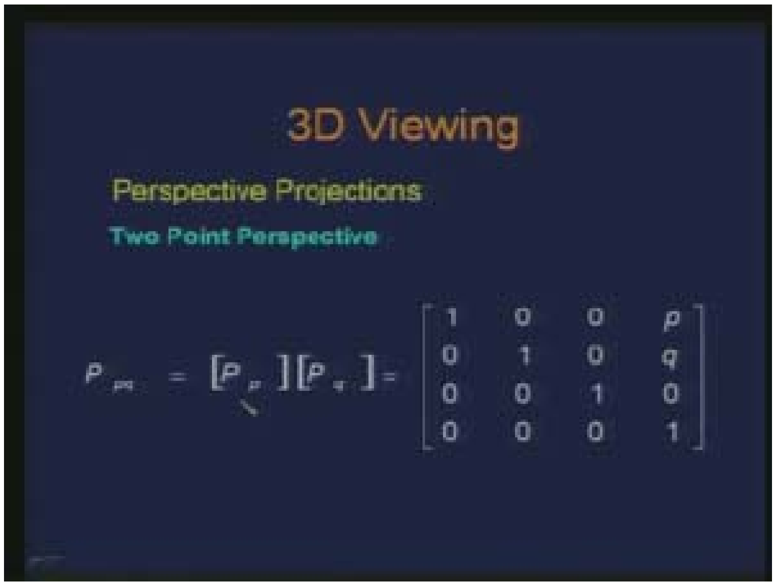
$$[x\ \ y\ \ z\ \ 1]\begin{bmatrix}1 & 0 & 0 & p\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} = [x\ \ y\ \ z\ \ px+1]$$

$$[x^*\ \ y^*\ \ z^*\ \ 1] = \left[\frac{x}{px+1}\ \ \frac{y}{px+1}\ \ \frac{z}{px+1}\ \ 1\right]$$

COP $(-1/p\ 0\ 0\ 1)$ VP_x $(1/p\ 0\ 0\ 1)$

So we looked at basically a perspective projection a single point perspective where the matrix representation of this looked like this. One of the entries in the last column you have is a non zero when we have the center of projection lying on X axis. So a point through this transformation becomes this, this is the homogeneous part of the coordinate and when I get the normal coordinates the Cartesian coordinates I divide it by this w and I get this which in turn gives me the center of projection lying at this point minus 1 by P 0 0 1 and the corresponding vanishing point I get as 1 by P 0 0 1 and we absorbed how to get a vanishing point in this context. Take a point at infinity apply the transformation and whatever points you get corresponding to that is the vanishing point by the definition of vanishing point itself.

(Refer Slide Time: 2:36)



3D Viewing

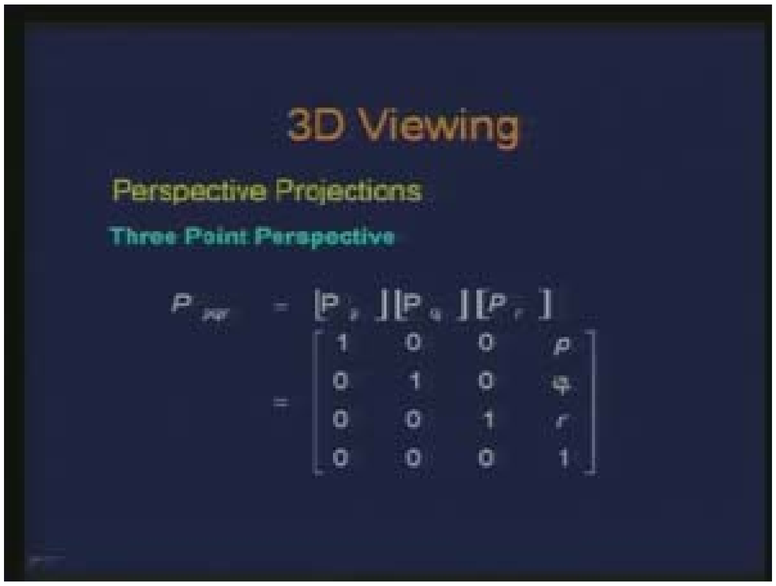
Perspective Projections

Two Point Perspective

$$P_{pq} = [P_p][P_q] = \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then we looked at two point perspective where in fact one can get two point perspective just by concatenating the two single point perspective and that is what is happening here. You have one point perspective here, this is also one point perspective and this is the resulting transformation for two point perspective. And this gives us correspondingly two centers of projections and the respective vanishing points.

(Refer Slide Time: 3:11)



3D Viewing

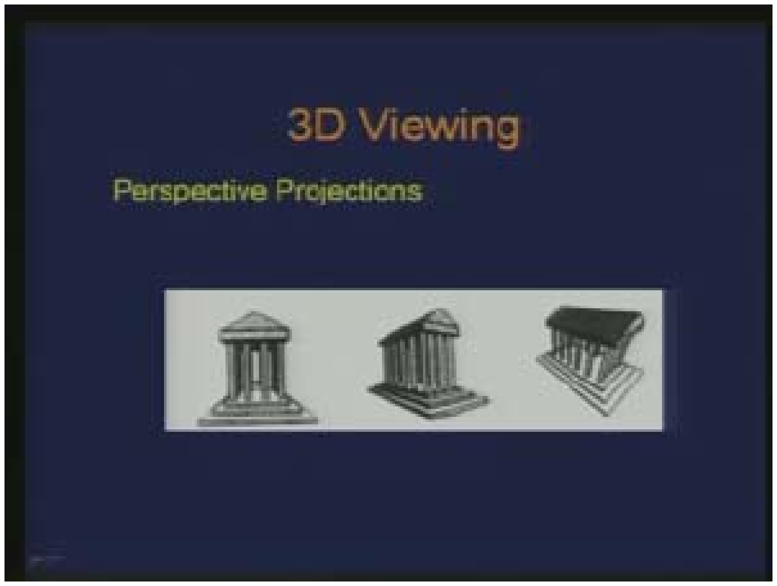
Perspective Projections

Three Point Perspective

$$P_{pqr} = [P_p][P_q][P_r] = \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly you can extend the idea of obtaining three point perspectives where we have all the three entries in the last column to be non zero and this is again a concatenation of the three individual single point perspectives. This is how we can observe the perspective transformation in the form of a matrix, single point perspective, two point perspective or three point perspective. And we also looked at the generation of these perspectives as to how do we sensible generate the two point perspective or three point perspective so which basically required a transformation prior to when you apply a single point perspective. So a single rotation gives raise to a two point perspective and two rotations may give raise to three point perspective so these are the examples. This is the single perspective transformation, this is two point perspective and this is three point perspective.

(Refer slide time 4:29)

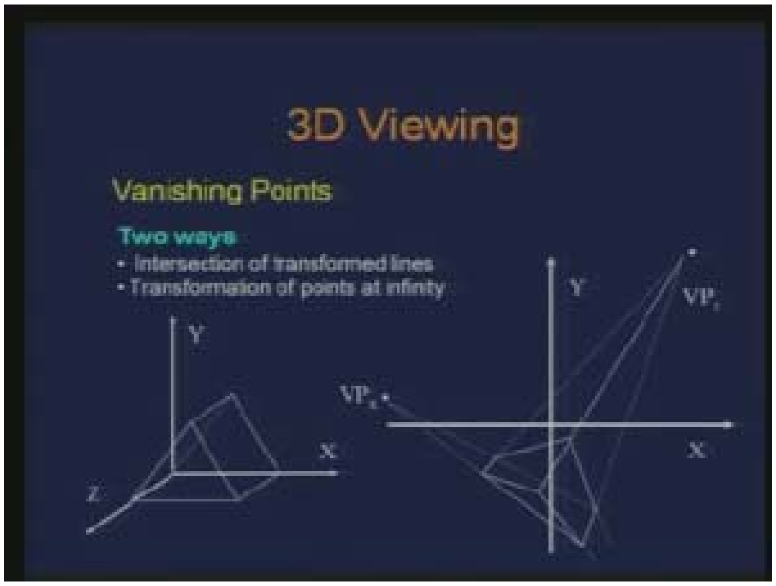


3D Viewing

Perspective Projections

You can see how the parallel lines with respect to the individual axis seem to converge in all these three cases. Parallel lines will converge and they may not be parallel to the coordinate axis. These vanishing points which I am obtaining here are corresponding to the principle axis the coordinate axis. So there might be instances where I have the parallel lines converging to a certain point and these parallel lines are not parallel to the coordinate axis. So those are also the vanishing points and we also refer to them as tracing points. There are parallel lines where this will not converge to a point in the case with respect to the projection plane. If I have a projection plane and lines parallel to the projection plane there will be instances where parallel lines may not converge. So now talking about the vanishing points we have already seen how we obtain vanishing point.

(Refer Slide Time: 6:24)


3D Viewing

Vanishing Points

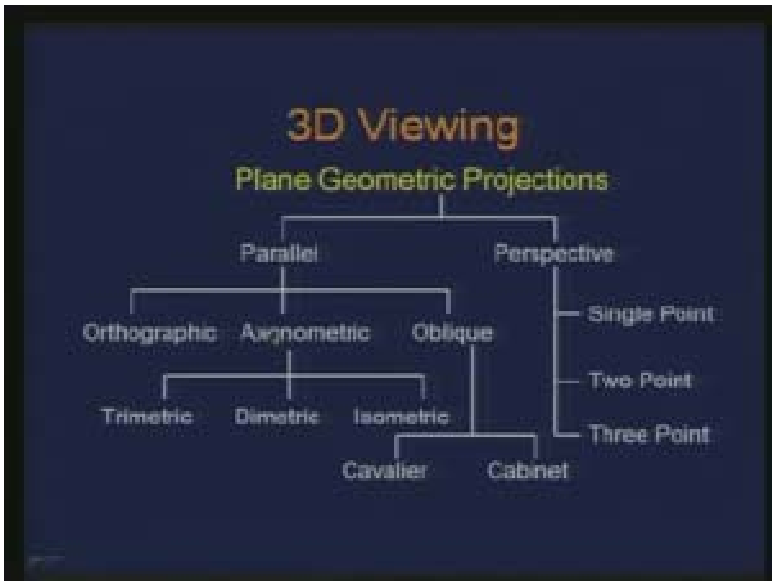
Two ways

- Intersection of transformed lines
- Transformation of points at infinity

In fact one can do this in two ways. Let us say I am trying to locate vanishing point on the projection plane or the image plane. So this is a 3D object and my interest is to find the vanishing points or the image of the vanishing points on to the projection plane. So this is the resulting image or the projection of this object. The way I can obtain vanishing points is either I can look at the vanishing point to be the point of intersection of the parallel lines which have been projected here. So here for instance the vanishing point in X could be the lines which will get projected corresponding to these lines which are parallel to X axis here that is these two and similarly I may get the vanishing point in Z direction by taking the intersection of the lines which are the projected lines parallel to this Z axis. Therefore that gives me the vanishing point.

Here is an example where I have these parallel lines or these parallel lines they may also seem to converge at some point and this may not be my principle vanishing point but this may be some vanishing point for tracing. So a simple approach is that you take the intersection of transform X. But this is computationally expensive. Since I know by definition that vanishing points are nothing but the images of points at infinity a better way to do is take the transformation of points at infinity. I know the transformation which is being used for transforming this particular object. So all I have to do is take correspondingly the points at infinity particularly referring to the principle vanishing points and just apply the transformation matrix that will be the computationally more efficient. Now let us just do a total recapitulation of what we had seen in the various kinds of transformation for 3D viewing particularly referring to the viewing transformation.

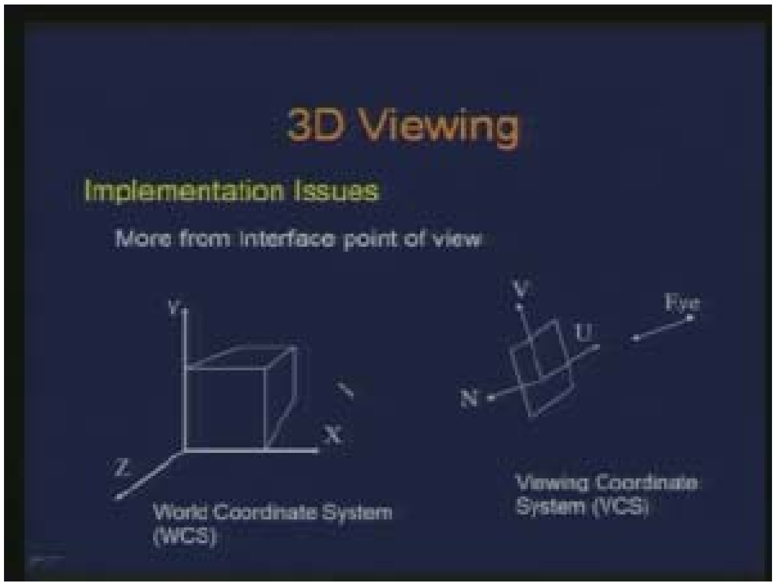
(Refer Slide Time: 9:20)


3D Viewing

Plane Geometric Projections

We have basically looked at two major categories of transformation the parallel projection and perspective projection. Within parallel projection we had orthographic projection, axonometric projection and the oblique projection. So here what we observe is that the projectors are parallel. Depending on how we place the projection plane we get these three kinds of parallel projection. Within axonometric we have trimetric, dimetric and isometric depending on the foreshortening. And similarly for oblique we have the type cavalier and cabinet and in perspective we have single point two point or three point perspective. This gives you the overview of all the transformation we had seen with respect to viewing transformations.

(Refer Slide Time: 10:41)


3D Viewing

Implementation Issues

More from interface point of view

Y
X
Z
World Coordinate System (WCS)
V
U
N
Eye
Viewing Coordinate System (VCS)

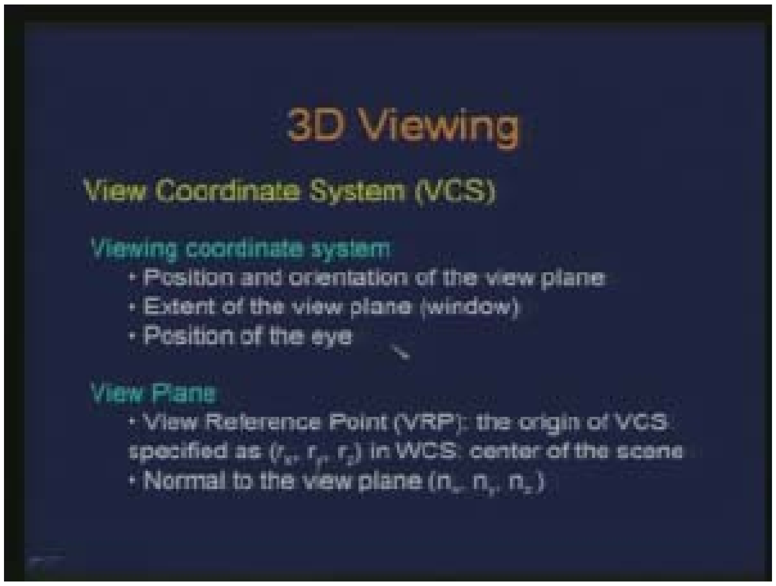
Now one of the things which are of concern is that the implementation. When I say implementation I am basically referring to the interface of having this 3D viewing implementation. So interface by interface I basically mean that how would the user like to specify the various inputs for getting the transformation of 3D viewing. Therefore an object or a scene is specified in a coordinate system which I call it as world coordinate system. So this is the world coordinate system where the world or the scene is specified. Then I have what I call as viewing coordinate system.

Viewing coordinate system is that I have the notion of defining a viewer or an eye or a camera. I also have a notion of defining this image plane or the viewing plane. So I would like to define this in some other coordinate system. Therefore the relationship between this viewing plane and the eye is defined in another coordinate system which I call as viewing coordinate system. And this viewing plane is basically defined by enormal to the viewing plane and the other two orthonormal coordinate axis. So this is my eye coordinate system or camera coordinate system or viewing coordinate system and this is the world coordinate system. So whenever I am establishing the desired transformation for 3D viewing I have to see that the word is specified in one coordinate system and the camera and other parameters may actually be defined in another coordinate system. So there is a requirement of having the relationship between these two coordinate systems and then only I can work on my scene.

Here what we are doing is that whenever I give these two I am basically trying to establish the relationship of the image plane of the projection plane and the camera or the eye and all these are interrelated.

Now, given this i, somewhere I have this coordinate system with respect to that. Actually you can specify everything in the world but that is not perhaps the most natural way of specifying. What we are looking at is the convenient way of specifying these. Another thing is that this world coordinate system is actually a right handed system as you absorb here whereas here when I see the eye coordinate system it is a left handed system. This is also for the matter of convenience I have a convention that the line of sight which is going from the eye towards the scene corresponds to the N axis. This is just a convention so it makes this coordinate system a left hand coordinate system.

(Refer Slide Time: 15:19)



3D Viewing

View Coordinate System (VCS)

Viewing coordinate system

- Position and orientation of the view plane
- Extent of the view plane (window)
- Position of the eye

View Plane

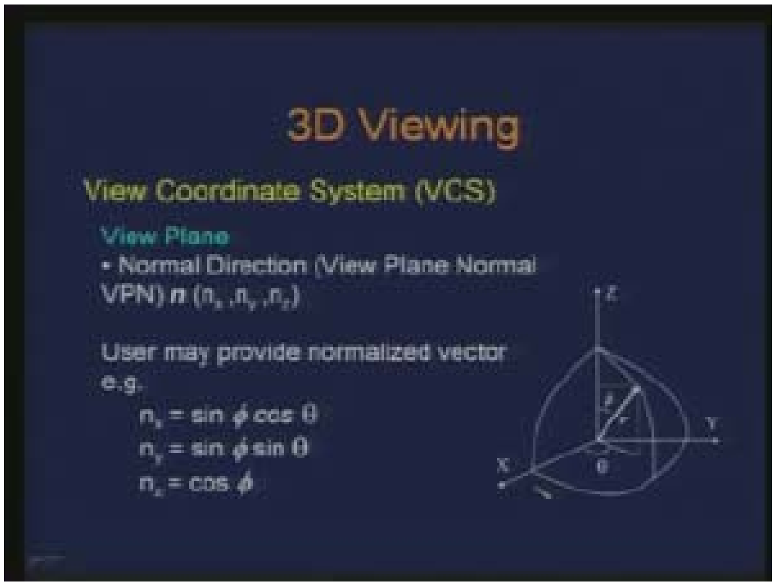
- View Reference Point (VRP): the origin of VCS specified as (r_x, r_y, r_z) in WCS: center of the scene
- Normal to the view plane (n_x, n_y, n_z)

After having seen this let us try to individually look at what we require for specifying the various entities in these two coordinate systems. In viewing coordinate system I need to have the position and orientation of the viewing plane. Where do I locate the viewing plane is important. And the extent of the viewing plane defines the window where I want to capture the projection of the scene as the image. Then the position of the eye basically takes care of having the viewing coordinate system defined.

Considering viewing plane or the view plane what I have is a view reference point I call it as VRP which can act as the origin of the viewing coordinate system. And that may be specified as r_x, r_y, r_z in world coordinate system. Therefore this VRP which is the viewing reference point is some sort of an offset with respect to the origin of the world coordinate system which locates the viewing plane or the origin of the viewing coordinate system. Now this could be actually given as the center of this scene where I can consider the center of the scene to be the VRP or it could be any point defined by the user.

All that VRP is doing is actually telling you where is the origin of the viewing coordinate system with respect to the origin of the world coordinate system. And normal to the view plane is defined by the vector n that is n_x, n_y, n_z. This is normal to the viewing plane. Therefore this basically locates the viewing plane.

(Refer Slide Time: 17:50)



3D Viewing

View Coordinate System (VCS)

View Plane

- Normal Direction (View Plane Normal VPN) $\boldsymbol{n}$ (n_x, n_y, n_z)

User may provide normalized vector e.g.

$n_x = \sin\phi \cos\theta$

$n_y = \sin\phi \sin\theta$

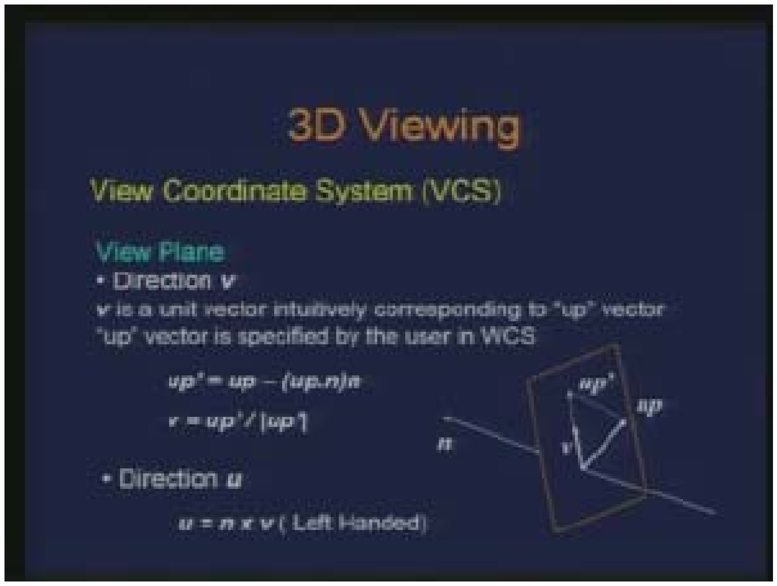
$n_z = \cos\phi$

For defining the normal direction there could be various ways. One can actually define a vector and say that this is the normal of the viewing plane. But there could be other ways which are thought to be more intuitive and convenient.

If you consider in spherical coordinates what we are saying is that the vector r in this spherical coordinate system could actually be defined as the normal vector where r is defined in terms of theta and phi. So what you are saying is that there is a sphere in the world and you are locating the normal of the viewing plane somewhere in the sphere. That could be one way of looking at defining this normal vector. So here I get this normal vector as this which is the unit normal vector.

One could also define the normal vector as just as minus r where r is my offset for VRP. If I locate the view reference point then my normal could just be defined as minus r. So there are various ways in which you can define this norm. So this is the prerogative of the user how the user wants to define. Therefore it is just giving the interface of the facility for the user to define this norm.

(Refer Slide Time: 19:53)



3D Viewing

View Coordinate System (VCS)

View Plane

- Direction *v*

v is a unit vector intuitively corresponding to "up" vector
"up" vector is specified by the user in WCS

$$up' = up - (up.n)n$$

$$v = up' / |up'|$$

- Direction *u*

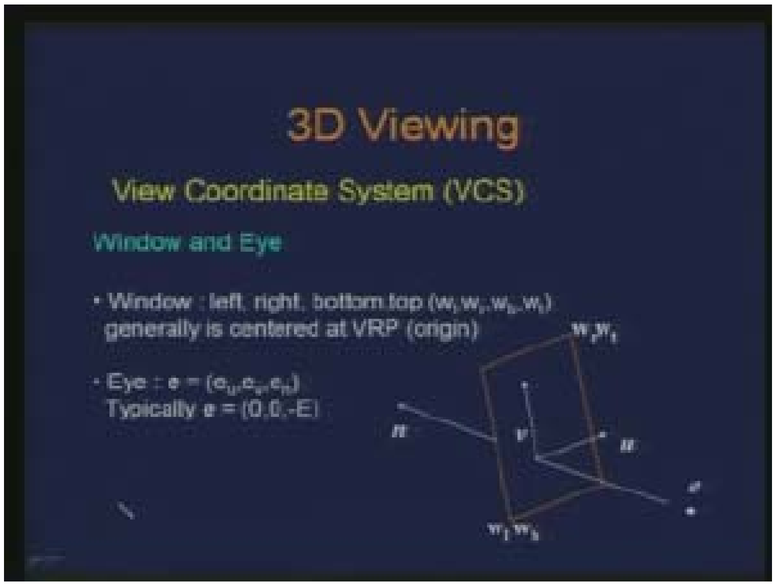
$u = n \times v$ (Left Handed)

We have basically looked at how to define the viewing plane but there are other corresponding entities with respect to the viewing plane that can actually be derived. Now what we are left with while defining with respect to the viewing coordinate system are the two other orthonormal vectors the u and the v. Now as far as the direction V is concerned we try to capture the vector v something which matches to the notion of an up vector.

Whenever we look at a scene we have a notion of what is the up vector so whenever we look at the scene we have a notion of what is the up vector. For example, in this room for instance when we are looking at we have the notion that yes this is my up vector if I go upside down then I have some other up vector so there is a notion of up vector. So what we try to do is we try to basically capture that up vector in the definition of v direction.

Now what we are saying is that let this up vector be defined by the user and world and we will try to take that information of up vector and define the corresponding v vector. So let us say this is the up vector defined then I can define the v vector, now if I take the projection of this up vector this is in the direction of the n vector then I am interested in getting this vector. So these are basically related as the up prime which is the projection along the n vector and this is the up vector prime so this is given as the up dot n in the direction of n that is this vector and this is my up vector so this is the up prime vector which corresponds to what I want to fix as the v vector. Therefore I just normalize that dividing by the magnitude of this and I get the unit v and that gives me the direction for v. Once I fix the vector v I can get the direction u which is by taking the cross product and since the system is left handed I would take n into v. So this basically fixes the three directions I needed for the viewing coordinate system and the location of its origin which is defined by the v_i.

(Refer Slide Time: 23:42)



3D Viewing

View Coordinate System (VCS)

Window and Eye

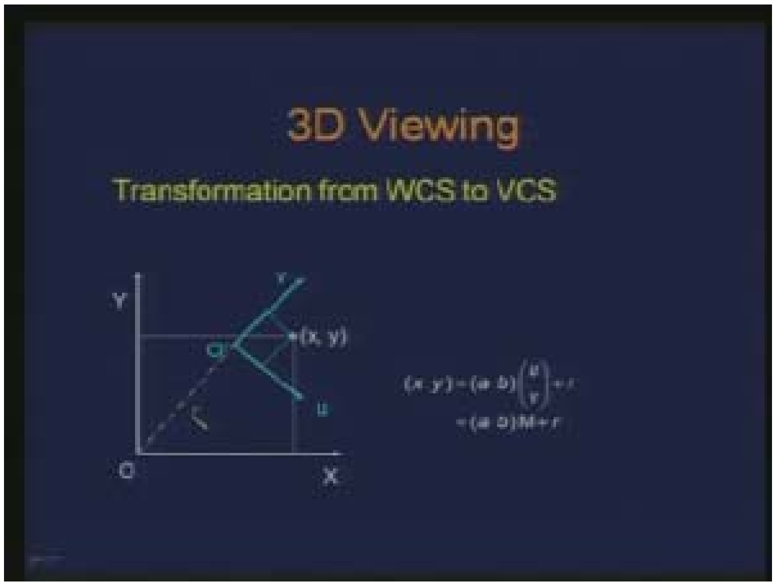
- Window : left, right, bottom top (w_l, w_r, w_b, w_t) generally is centered at VRP (origin)
- Eye : $e = (e_u, e_v, e_n)$ Typically $e = (0, 0, -E)$

$w_r w_t$
n
v
u
e
$w_l w_b$

Now the window which is the extent of the viewing plane or the image plane can just be defined as the extent in terms of the left extent, the right extent, the bottom extent and the top extent wl, wr, wb and wt. So this is the just the extent of the plane. And the i now which could be located in this coordinate system I am referring to the viewing coordinate system also as u v n system where this e could be located as ev, eu, en in this coordinate system. So this is located in this coordinate system.

Now typically we absorb that i is actually defined in such a way that it coincides with the normal of the viewing plane. So line of sight is actually the normal of the viewing plane. Now what has happened is I have basically defined a viewing coordinate system, I have the world defined in the world coordinate system. Now what do I need is some mechanism by which I can make a transformation from one system to the other system. Therefore in order to do that let us try to see a simpler case.

(Refer Slide Time: 25:37)



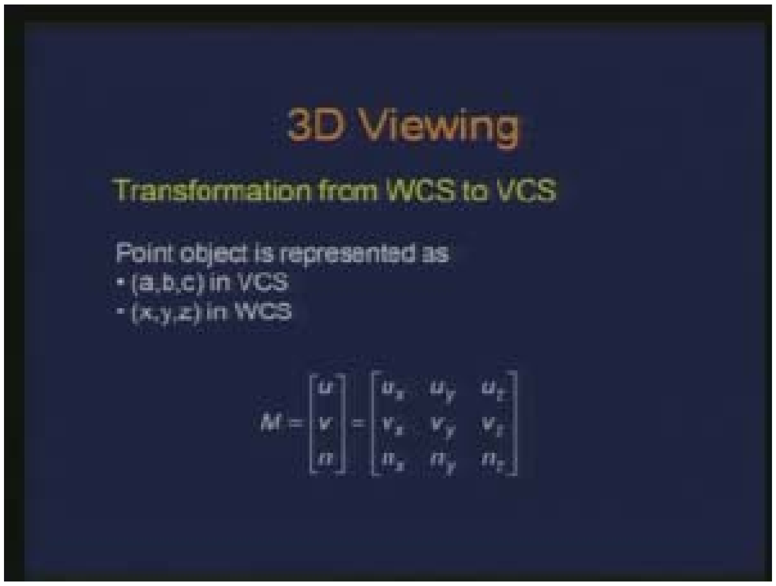
3D Viewing

Transformation from WCS to VCS

$$(x\ y) = (a\ b)\begin{pmatrix} u \\ v \end{pmatrix} + r$$
$$= (a\ b)M + r$$

In a 2D when I need to make the transformation from one coordinate system to another coordinate system. For instance, I had this coordinate system where I had this x and y axis and the other coordinate system defined by u and v a point located here (x, y) can be related to a point in this coordinate system which could be given as a and b using this where r is the offset between O and O prime and if I define (u, v) as just my matrix of transformation then I get (x, y) as a transformation of (a, b) through m and r because (u, v) is nothing but a transformation of the unit vectors 1 0 and 0 1 so I am basically saying that uv is equal to 1 0 0 1 transformed by M. Therefore M is nothing but uv. So now I have the relationship of the two representations of the same points in the two coordinate systems. So this was in 2D and now I can also do this in 3D. This M is nothing but basically the unit vectors in the other coordinate system.

(Refer Slide Time: 27:30)



3D Viewing

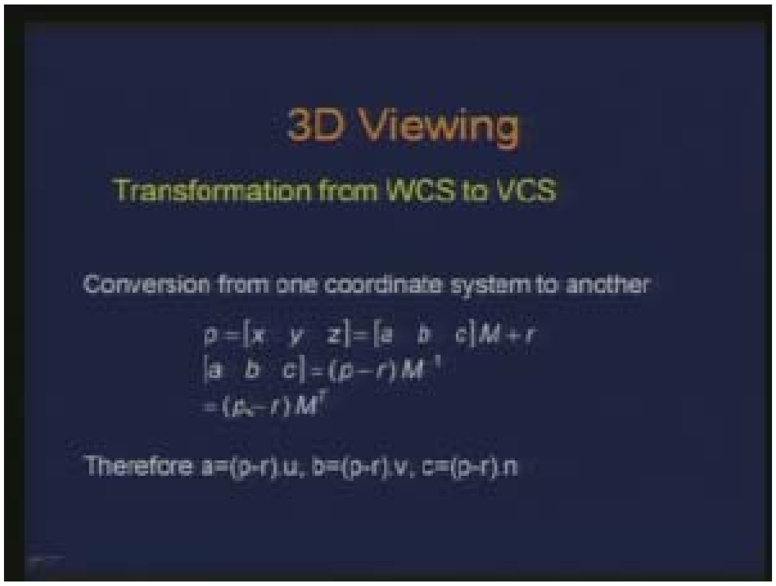
Transformation from WCS to VCS

Point object is represented as
- (a,b,c) in VCS
- (x,y,z) in WCS

$$M = \begin{bmatrix} u \\ v \\ n \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{bmatrix}$$

Now in our context what I have is the point a b c in the viewing coordinate system, x y z in the world coordinate system and this M is nothing but instead of just having (u v) I have (u v n). This matrix gives me the transformation M. So now this point p defined in world by (x, y, z) is related through the matrix M and the offset distance r with (a, b, c) and now I can get (a, b, c) from the point in the world.

(Refer Slide Time: 28:16)



3D Viewing

Transformation from WCS to VCS

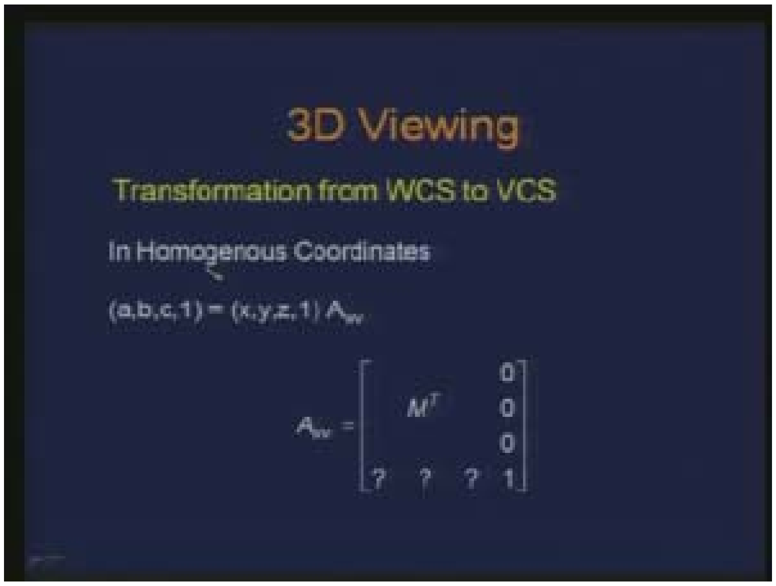
Conversion from one coordinate system to another

$$p = [x \quad y \quad z] = [a \quad b \quad c]M + r$$
$$[a \quad b \quad c] = (p - r)M^{-1}$$
$$= (p - r)M^T$$

Therefore a=(p-r).u, b=(p-r).v, c=(p-r).n

Remember that this matrix is actually an orthogonal matrix because it is going to be a combination of rotations and nothing more than that. So the inverse of this matrix is given by the transpose of this matrix. I basically have (a, b, c) related to the point p in world as (x, y, z) through this. And I can also write this in terms of the three dot products. So we have (p minus r).u, (p minus r).v and (p minus r).n. Now if I represent this in homogeneous coordinates as we deal with it, we have these transformations represented in homogeneous coordinates.

(Refer Slide Time: 29:40)



3D Viewing

Transformation from WCS to VCS

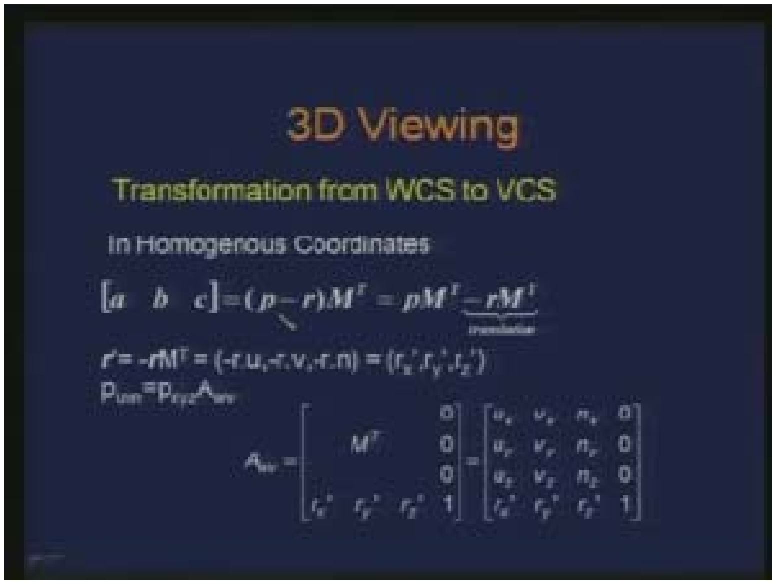
In Homogenous Coordinates

$(a,b,c,1) = (x,y,z,1)\, A_{wv}$

$$A_{wv} = \begin{bmatrix} & & & 0 \\ & M^T & & 0 \\ & & & 0 \\ ? & ? & ? & 1 \end{bmatrix}$$

So (a, b, c, 1) is obtained from the world coordinate system point (x, y, z) through a matrix A from world to viewing. Now Awv is nothing but a matrix consisting of the transpose of M plus some other terms which could be for the translation. So this sub matrix the 3 into 3 sub matrix here is nothing but the transpose of M.

(Refer Slide Time: 30:35)



3D Viewing

Transformation from WCS to VCS

In Homogerous Coordinates

$$[a \quad b \quad c] = (p - r)M^T = pM^T \underbrace{- rM^T}_{\text{translation}}$$

$r' = -rM^T = (-r.u, -r.v, -r.n) = (r_x', r_y', r_z')$

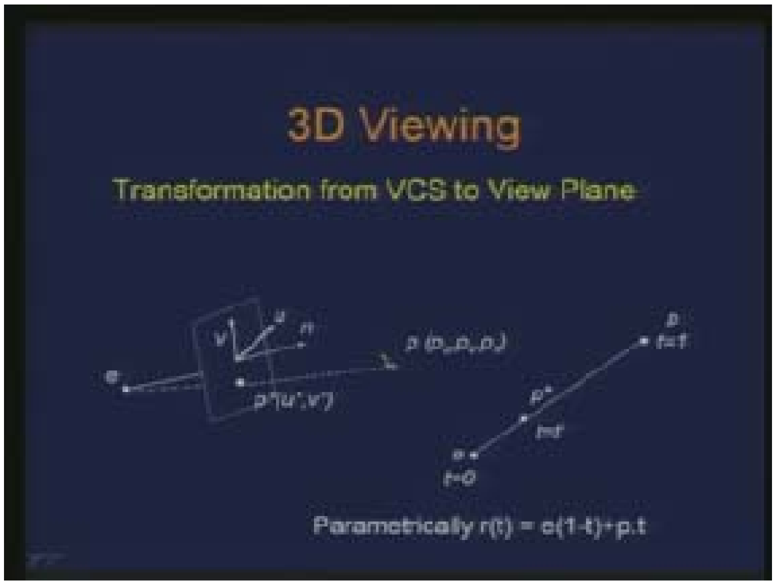
$P_{uvn} = P_{xyz} A_{wv}$

$$A_{wv} = \begin{bmatrix} & M^T & & 0 \\ & & & 0 \\ & & & 0 \\ r_x' & r_y' & r_z' & 1 \end{bmatrix} = \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ r_x' & r_y' & r_z' & 1 \end{bmatrix}$$

Now if I go back and see the exact form of (a b c) from the point p I have this which I just rearrange and get a term what gives me the translation. So this is nothing but the transformation of point p through the matrix M power T and this is the term which is accounting for the translation. So I just look at this which is nothing but the r getting transformed by M power T which I can write as r_x prime, r_y prime, and r_z prime so these are the three translations corresponding to this.

Now the point in (u v n) or the viewing coordinate system is related to the point P in the world as (x y z) by this matrix A_{wv} where A_{wv} is given by this. These are the translation terms added and this is nothing but the column vectors of the unit vectors in the viewing coordinate system. So I can basically derive a procedure of converting the point in the viewing coordinate system and the point in the world coordinate system I can go either way from world to the viewing coordinate system or viewing coordinate system to the world. What is r? r is the VRP. Remember that we have defined this view reference point so it is nothing but the offset vector between the origin of the world coordinate system and viewing coordinate system. It is basically locating the origin of the viewing coordinate system.

(Refer Slide Time: 33:08)



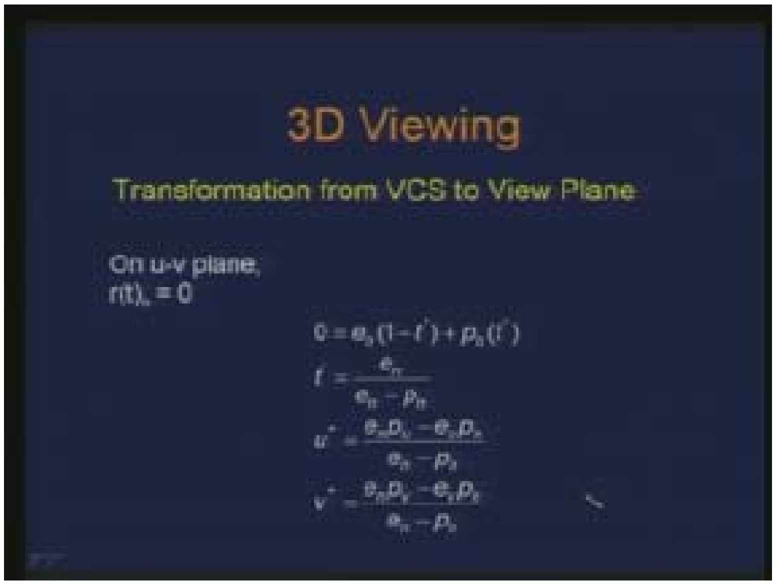
3D Viewing

Transformation from VCS to View Plane

Parametrically r(t) = e(1-t)+p.t

When we look at it in a geometrical sense what is basically happening now Is, I have a point p given in the viewing coordinate system through pu, pv, pn this is my viewing plane and this is the eye. So I get the projected point at p star on this viewing plane and I am defining the viewing plane at n is equal to 0 so the third coordinate is 0. So I am basically having the coordinates as u star and v star. Now if I consider this is nothing but a ray emanating from the point e which is the eye point in the world but now the world is in viewing coordinate system. I have already obtained this point in the viewing coordinate system that I can do now. So I can define this ray from e to p and t star is some point on this line or on this ray at some t is equal to t prime. Therefore if I take the parametric definition of the ray or the line this r is not the same r which we had seen because that r was a vector r distance from the world coordinate system but I am representing this as just a parametric ray.

(Refer Slide Time: 35:10)



3D Viewing

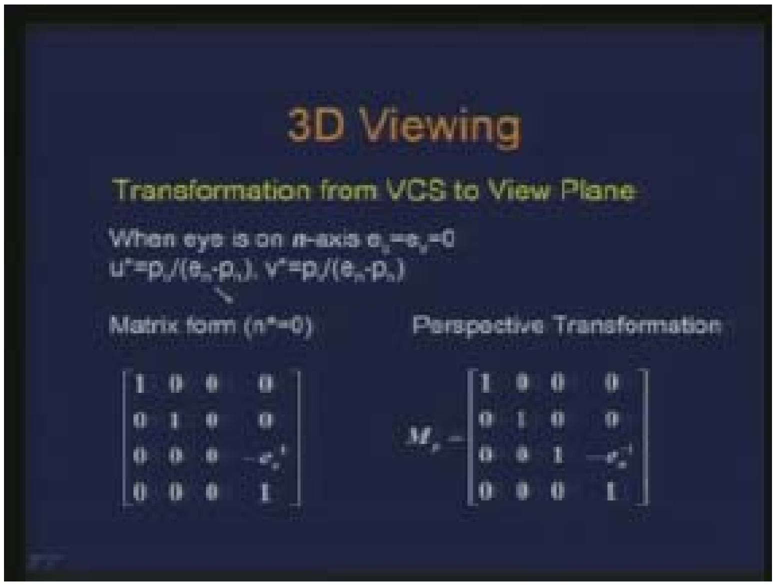
Transformation from VCS to View Plane

On u-v plane,
$r(t)_n = 0$

$$0 = e_n(1 - t') + p_n(t')$$
$$t' = \frac{e_n}{e_n - p_n}$$
$$u^* = \frac{e_n p_u - e_u p_n}{e_n - p_n}$$
$$v^* = \frac{e_n p_v - e_v p_n}{e_n - p_n}$$

Now this can be parametrically written like this and if I am interested in finding out t star, hence all I am saying is that on (u v) plane where I consider n is equal to 0, I have this r_t at n equal to 0 given as this. So this gives me the t prime corresponding to the point on the viewing plane. And now by substituting this t prime I can get u star and v star. So these are my transform points u star and v star. Now I further simplify and say that i is located such that this e_u and e_v goes to 0.

(Refer Slide Time: 36:11)



3D Viewing

Transformation from VCS to View Plane

When eye is on n-axis $e_u = e_v = 0$
$u^* = p_u/(e_n - p_n)$, $v^* = p_v/(e_n - p_n)$

Matrix form ($n^* = 0$)

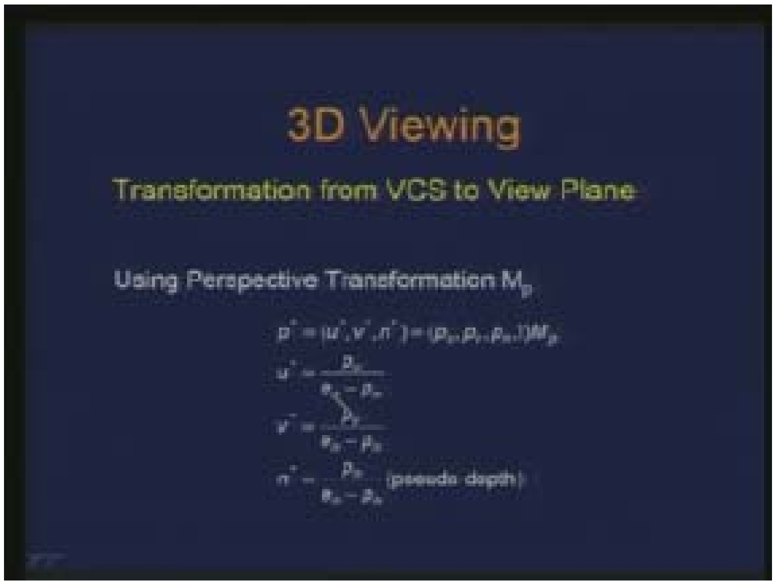
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -e_n^{-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective Transformation

$$M_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -e_n^{-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

That means direction of i is coincident to the normal vector. In that case I observe that u star becomes this p by e_n minus p_n and similarly v star becomes p by e_n minus p_n. In fact I can write this in a matrix form like this, this is 1 by en and this is nothing but actually a single point perspective projection and the corresponding matrix for single point perspective transformation is this. All I am saying is that I have this transformation which is happening to this matrix provided the i is on n axis and that is what I am assuming.

(Refer Slide Time: 37:57)



3D Viewing

Transformation from VCS to View Plane

Using Perspective Transformation M_p

$$p^* = (u^*, v^*, n^*) = (p_u, p_v, p_n, 1) M_p$$

$$u^* = \frac{p_u}{e_n - p_n}$$

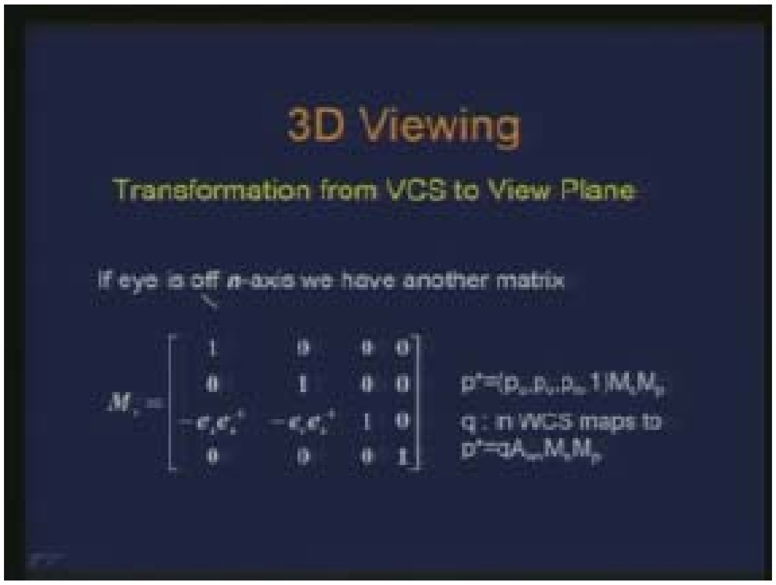
$$v^* = \frac{p_v}{e_n - p_n}$$

$$n^* = \frac{p_n}{e_n - p_n} \text{ (pseudo depth)}$$

Therefore what is said here is, if I use this perspective transformation given through the matrix Mp to get the points on the projection plane as u star v star n star in general so this n star is NOT0 because it does not act to change anything then I have the u star v star and n star given as this and this n star is actually capturing the depth information. This is important when we discuss hidden surface elimination.

So even after applying the perspective transformation we need a mechanism by which you can order the depth such that the points which are further and if they are occluded by the points in front they are not showing. That would be obtained by using information from this third coordinate. What we have basically established is that there is a matrix Mp which is giving us a transformation to get the point on the projection plane and that we have seen earlier also. But within the interface which we are specifying how we can obtain the big matrix A_{wv} is what we are trying to see.

(Refer Slide Time: 39:58)



3D Viewing

Transformation from VCS to View Plane

If eye is off *n*-axis we have another matrix

$$M_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -e_u e_n^{-1} & -e_v e_n^{-1} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

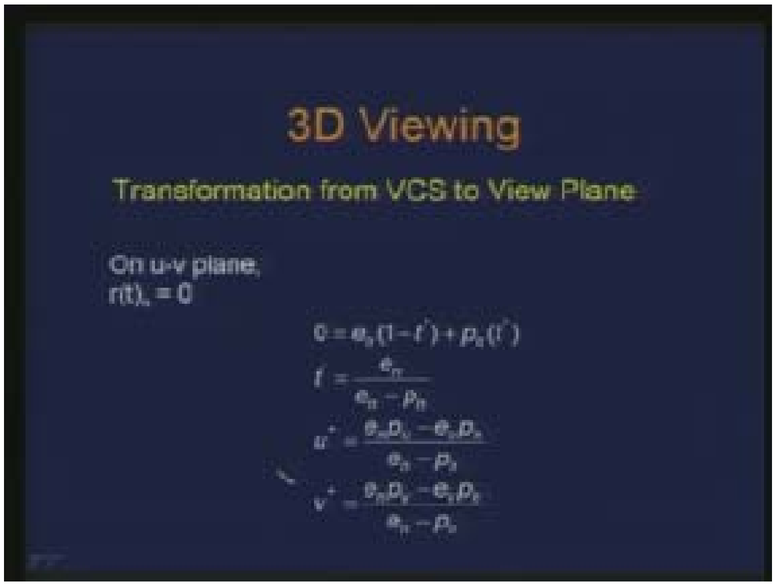
$p^* = (p_u, p_v, p_n, 1) M_s M_p$

q : in WCS maps to

$p^* = q A_{wv} M_s M_p$

Now I relax this condition where I considered the eye lying on the n axis. So, if I have the eye lying off n axis what happens is that I would get another matrix, earlier we saw that the u star v star coming as this, but now I want to capture this through a matrix.

(Refer Slide Time: 40:22)



3D Viewing

Transformation from VCS to View Plane

On u-v plane,
$r(t)_n = 0$

$$0 = e_n(1 - t^*) + p_n(t^*)$$

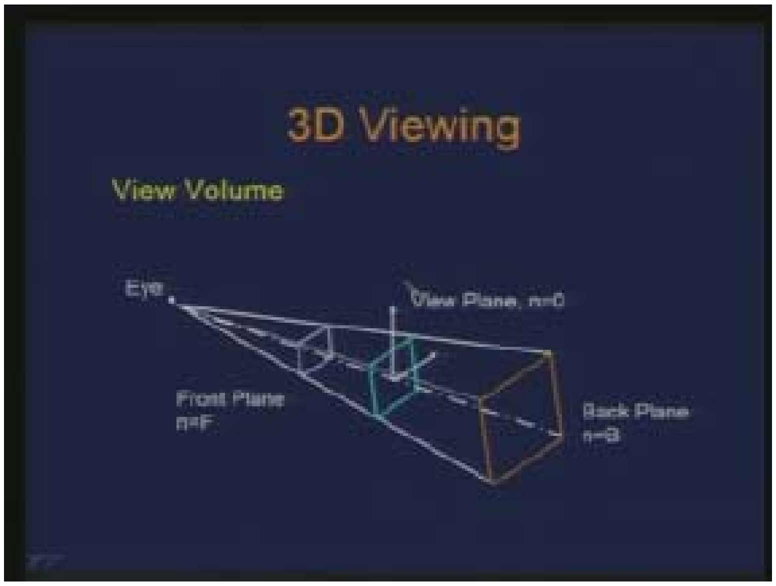
$$t^* = \frac{e_n}{e_n - p_n}$$

$$u^* = \frac{e_n p_u - e_u p_n}{e_n - p_n}$$

$$v^* = \frac{e_n p_v - e_v p_n}{e_n - p_n}$$

Basically I observe that there is an additional matrix which I need before I apply matrix Mp. Corresponding to the points we have obtained u star v star so matrix of this nature can be obtained. By looking at the matrix can we suggest what kind of transformation it would be? There is a term in the off diagonal, this would cause some sort of a shear. Therefore what we are saying is that this point p star after the matrix transformation this Ms and the transformation matrix Mp will give me the point transform when I consider eye not to be coincided with the n axis. It is a more general case. If I have a point q in the world coordinate system it will map to p star through this. So A_{wv} was nothing but the transformation from world to viewing. That is where we defined the point in viewing coordinate system. And after that we further applied these two transformations. Now what has been basically achieved?

(Refer Slide Time: 42:53)

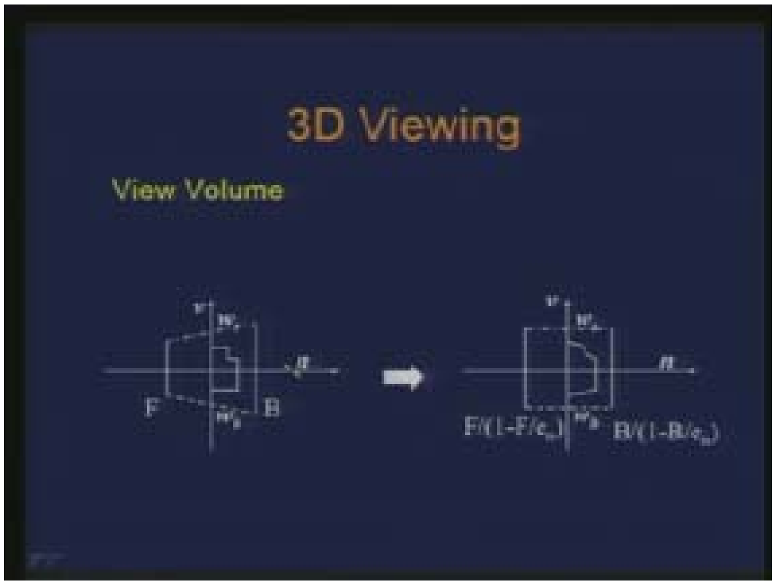


3D Viewing

View Volume

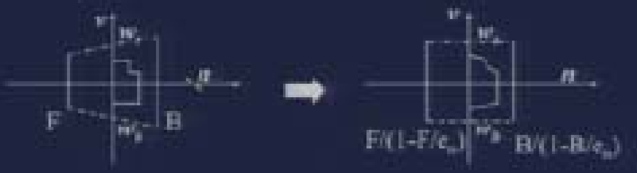
We have everything in a coordinate system like this where I have the eye, the viewing plane. So what I basically define is some sort of a volume within the coordinate system. Remember that this viewing plane actually has got extents. If I just join the point eye and emanate a ray from the extent of the window I will basically get some sort of a volume if I also define certain planes to clip this volume in the n direction the direction towards the line of sight. So all I am saying is that I can actually put the scene in this view volume. This is also necessary and important that whatever scene gets intercepted within this volume is of interest and that is where your clipping comes from. Eye can also perform clipping of the scene with respect to this volume. So whatever gets intercepted here will be of interest. I define the two planes of the extents towards the line of sight so there could be two planes defining the front plane and the back plane. So all it is saying is this is the front of the viewing plane here and this is the back of the viewing plane. It is just to give the extent in the depth which I am interested in.

(Refer Slide Time: 45:08)



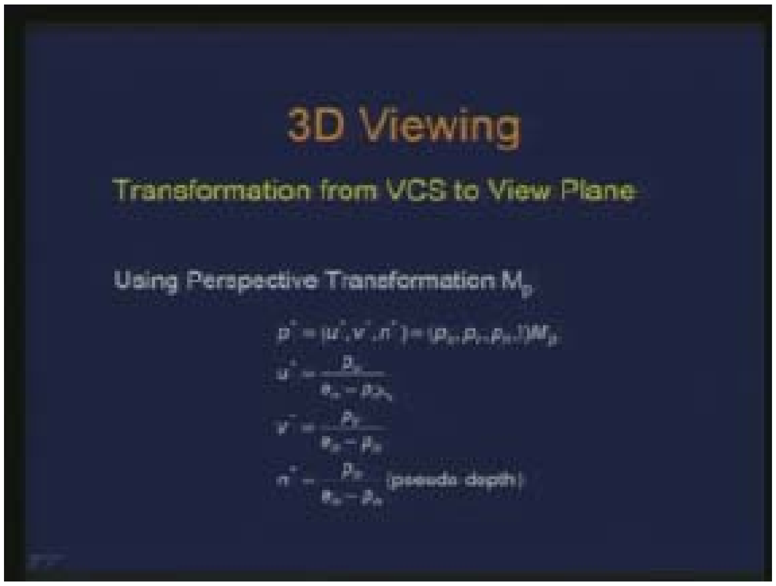
3D Viewing

View Volume



Now having seen this viewing volume if we just look at the location of the front plane and the back plane in the direction n then I see that the front plane is located at F and the black plane is located at B in n direction. So what has been achieved is that after having transformed from world to view I would get something like this. And these are my window extents W_b and W_t in v direction. Now what happens when I apply the matrices like M_s and M_p? Remember that if I have the point at a distance F in n direction will actually get mapped to something like this. Similarly a point B in n direction will actually get mapped to something like this.

(Refer Slide Time: 47:03)



3D Viewing

Transformation from VCS to View Plane

Using Perspective Transformation M_p

$$p^* = (u^*, v^*, n^*) = (p_u, p_v, p_n, 1)M_p$$

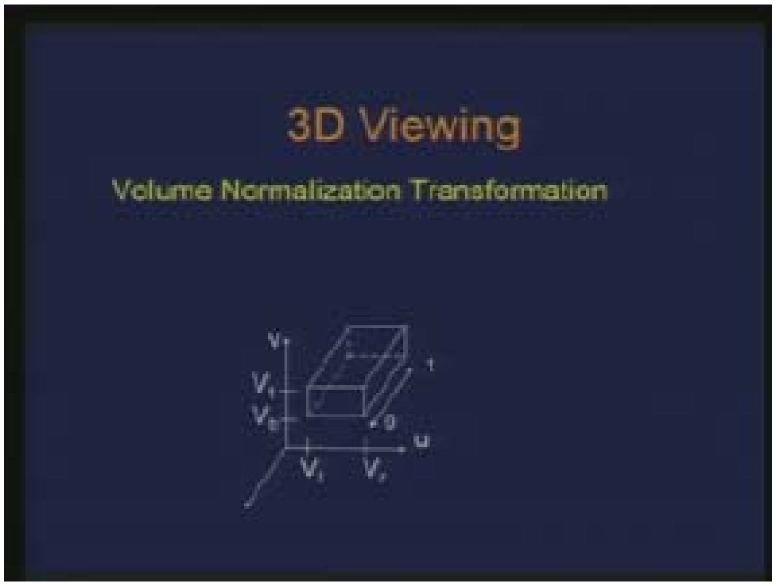
$$u^* = \frac{p_u}{e_n - p_n}$$

$$v^* = \frac{p_v}{e_n - p_n}$$

$$n^* = \frac{p_n}{e_n - p_n} \text{ (pseudo depth)}$$

Now I am looking at only the points in n direction so it is located at F so this F will basically get mapped to this and B will get mapped to this. Now what is basically being achieved is some sort of a distortion or a warping of this volume into this and that is what we obtained from the perspective transformation. Perspective transformation is basically some sort of a distortion or a warping. So a scene like this would actually appear to you like this. So this whole thing is a matter of a warping or a distortion. Now what I have done is I have applied Awe from there I applied the other matrices Ms and p to get to this situation. Now what I do is I basically put this volume or make this volume normalized.

(Refer Slide Time: 48:32)

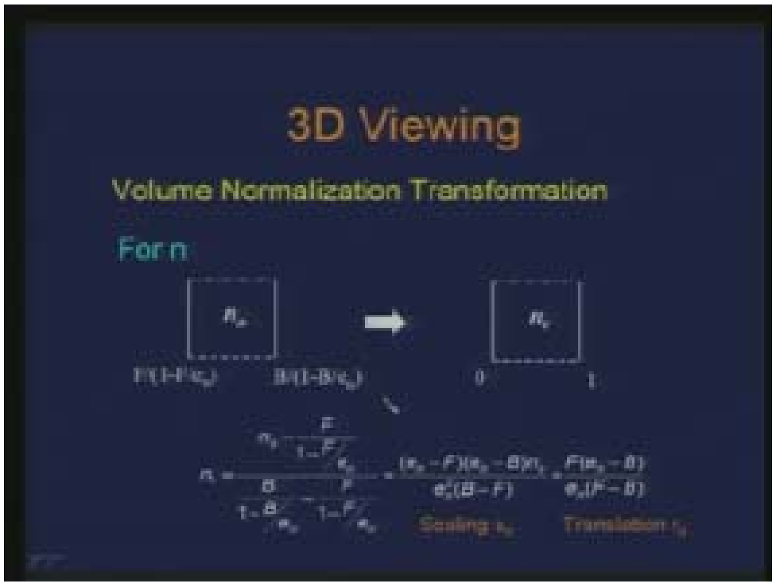


3D Viewing

Volume Normalization Transformation

What do I mean by normalized volume? I just define an extent of this volume for the directions u and v using V_l and V_r and V_t and the depth between 0 and 1. So the depth is basically normalized between 0 and 1. Here I am giving you an example where the normalization is done between 0 and 1 but in many cases the normalization is done between minus 1 and 1. For instance OpenGL does between minus 1 and 1 but that does not change anything here. Now what I want to do is after having obtained the distortion or the distorted volume I want to map that to this normalized volume. Therefore just consider for the part using coordinate n.

(Refer Slide Time: 49:44)



3D Viewing

Volume Normalization Transformation

For n

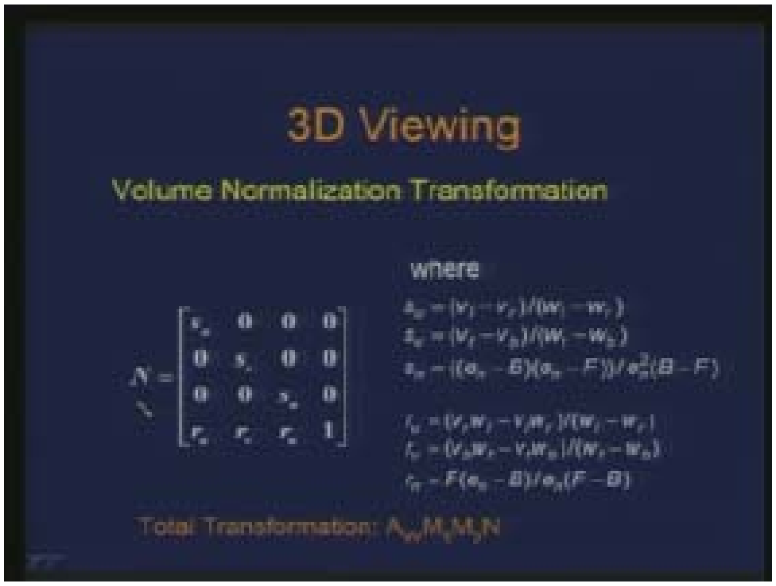
n_o
$F/(1-F/e_n)$
$B/(1-B/e_n)$
n_t
0
1

$$n_t = \frac{n_o - \dfrac{F}{1-F/e_n}}{\dfrac{B}{1-B/e_n} - \dfrac{F}{1-F/e_n}} = \frac{(e_n - F)(e_n - B)n_o}{e_n^2(B-F)} + \frac{F(e_n - B)}{e_n(F-B)}$$

Scaling s_n Translation r_n

Now this was my front plane, this was my back plane, now I want to map this to something like this, this maps to 0 and this maps to 1. So any point just for the part N that is (n_0) which is the original N gets mapped to something as N transform. So this is just a linear ratio which I am considering for the transformation. So all I am saying is that if I want to locate n_t within this then I can locate through using the ratio of n_0 within this, n_0 minus this divided by the whole line locates nt. What I absorb is that with respect to n_0 I have this part which is acting like a scaling to n_0 and there is some translation. I am just decomposing that to be a transformation in terms of scaling and transformation.

(Refer Slide Time: 51:40)



3D Viewing

Volume Normalization Transformation

$$N = \begin{bmatrix} s_u & 0 & 0 & 0 \\ 0 & s_v & 0 & 0 \\ 0 & 0 & s_n & 0 \\ r_u & r_v & r_n & 1 \end{bmatrix}$$

where

$s_u = (v_r - v_l)/(w_r - w_l)$
$s_v = (v_t - v_b)/(w_t - w_b)$
$s_n = ((e_n - B)(e_n - F))/e_n^2(B - F)$

$r_u = (v_l w_r - v_r w_l)/(w_r - w_l)$
$r_v = (v_b w_t - v_t w_b)/(w_t - w_b)$
$r_n = F(e_n - B)/e_n(F - B)$

Total Transformation: $A_{vw}M_sM_pN$

Once I do this I can actually form a matrix N which will do the normalization in all the three directions. Here this was done only in N direction, similar treatment can be done in other directions u and v and I can obtain a matrix of this nature in a very similar way just using the ratios. So these values s_u, s_v and s_n and r_u, r_v and r_n are basically given here derived in a very similar way. So I get a total transformation which looks like this from the world to the normalized volume basically given like this. And once I specify the input in terms of where the i is located, what is the N, what is the view reference point, what is the extent of the window then I can actually establish this. So your assignment is going to be based on this where the user is going to specify those parameters and you will achieve this. Therefore this basically gives you the entire viewing transformation pipeline. Once you do this you can actually get your clipping done in this pipeline. Instead of 2D clipping you can do 3D clipping. And now the question comes, is it better to do a 2D clipping or it is better to do 3D clipping.