Virtual Reality
Prof. Steve Lavalle
Department of Applied Mathematics
Indian Institute of Technology, Madras

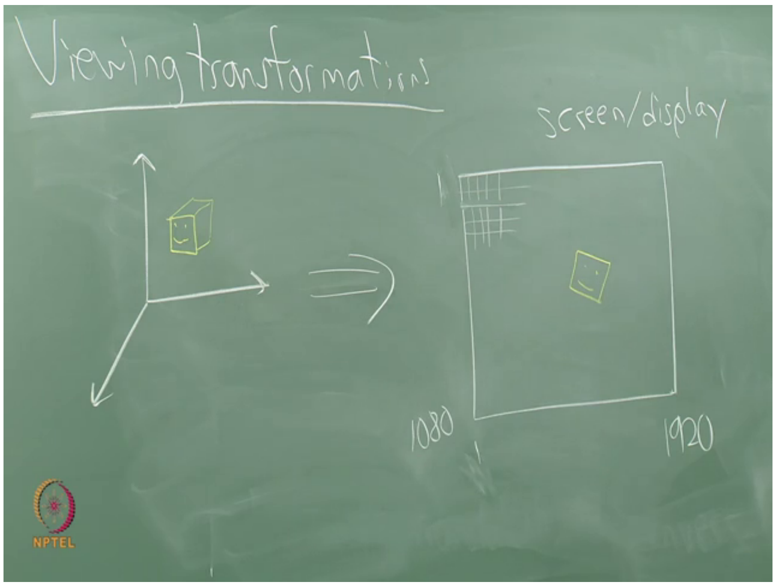
Lecture – 5-2

Geometry of Virtual Worlds (the chain of viewing transformations)

The next part of transformations that I want to get to is called viewing transformations. I want to build up enough transformations that. So, that we can place our models into the world, and also define the stationary part of the model in the world. And then figure out how it should look on the screen based on which way we are looking with our eyes.

So, I got to go through a lot of transformations to do, that it is not a computer graphics part yet it is a step towards, that it does not really talk about how to render the pixels let us say, what I just need to know what points in my world should be appearing at one particular spot on my screen. So, how do I apply all those transformations? So, that is the next part I am going to get to, if you have had a computer graphics class before you undoubtedly gone through this chain of transformations before.

(Refer Slide Time: 01:06)



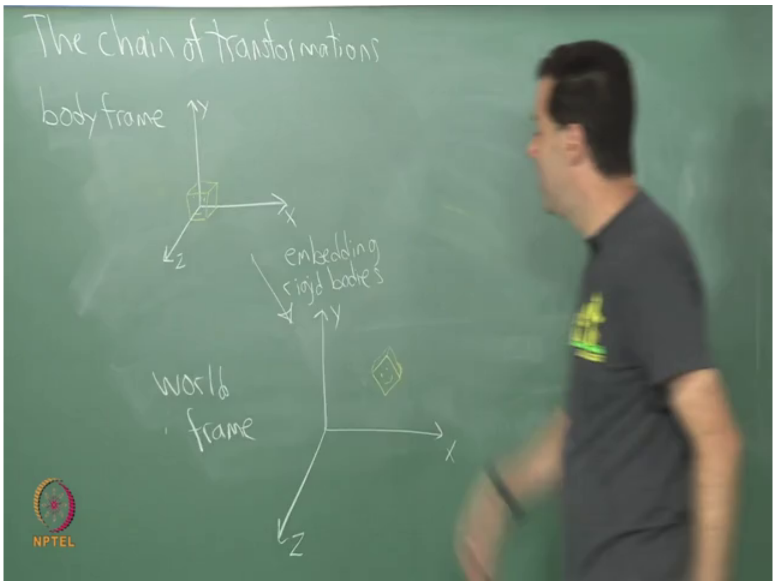
So, let us do that part. So, viewing transformations. So, in other words I want to start I want to start by putting my entire model into the world in some way this was the alternate world generator should build some model of the world, let us say at a current time. So, somewhere in this world I have an object appearing.

So, some kind of object appears, and I have to take this object and through some kind of transformations, figure out where it should appear on a screen or I guess I should say display how is this screen or display arranged perhaps it goes from 0 to 1080 and from 0 well let us see, but I will just say 1 to 1080 and 1 to 1920, and this can be considered as discretized just making up whatever coordinates I want here for the for the resolution it does not matter, but the screen has some kind of resolution and so, we alternatively going to have to figure out what RGB values to assign.

But at today's stage I just want to figure out where an object like this shows up if at all. So, somewhere in this on the screen, I should have my face appearing right let us say that shows up in the world. So, I want to figure out what these transformations are, I am not going to talk about how does the light propagate how does the discretization happen, should this be visible at all or not or be blocked by something else that is in the way of your view, and correcting for perspective projection. I will get into some of the math of that today, but I am not going to put the whole thing in together, we will come back to that at a later part in the course.

So, probably in the second week when we are covering more of the graphics pipeline issues and then talk about how those relate to virtual reality, which has it is own special set of challenges and problems that are not standard graphics. So, all of that becomes part of the rendering pipeline, but today I am just doing let us say the geometric transformation pipeline. So, how to do the math of it? So, in order to go into that I am going to have to I am going to have to make a rather large figure, but I would like it to all fit on the board. So, let me erase this.

(Refer Slide Time: 03:47)



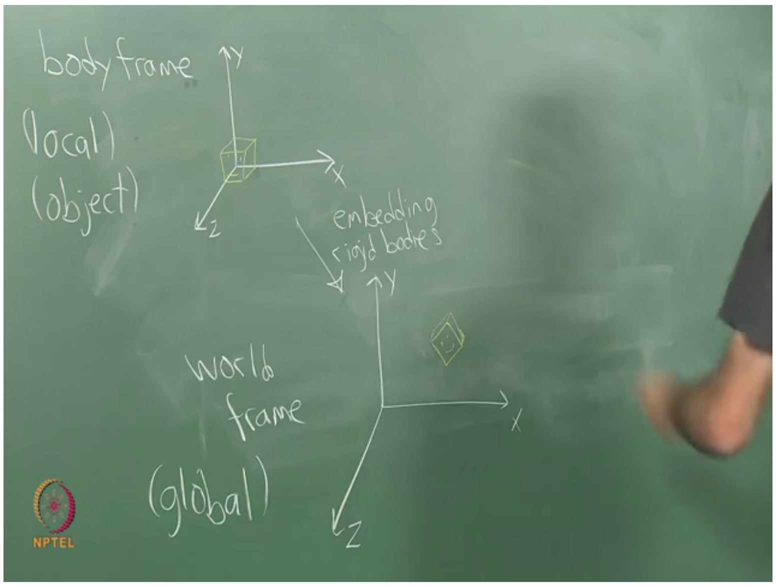
So, I want to consider the chain of transformations that go all the way from defining your moveable models to determining what should appear on the screen. So, I just want to go through the entire chain of transformations, I am going to write it at a high level on the board here and then go through each part. So, we talked about the body frame as the way of writing as the way of expressing our moveable models.

So, somewhere in this body frame all right x y z, I have some kind of object appearing here all right. So, some kind of object appears in this in this frame. We apply the transformations for example, the homogeneous transformation matrix that I just gave to perform the appropriate rotation and translation, to move that body into the world or embed that body in the world frame appropriately. So, that is one step, I will draw another frame here. So, somehow this through the transformation, this object in this case this cube face gets moved somewhere in the world, not going to get the geometry perfect here of course, it is just an illustration.

So, this is the world frame now some of the model is already going to be defined in the world frame right, but part of those were called the stationary model. So, if there is building some furniture that does not move maybe trees, that remain anchored and do not sway in the breeze things like that we talked about this is already defined in the world frame. So, there is no transformation you are already there and then for each and every one of your bodies that can move you have to perform a transformation to bring it into the world frame.

When you get to that stage, then you can start to talk about how it should look when it appears on the screen when we get to the later transformations. So, this particular transformation here going from body frame to world frame, occurs quite a bit right. So, I will call these embedding rigid bodies. So, it happens many many times typically embedding rigid bodies. There is different names for these if you go looking around sometimes this what we call it a local frame or sometimes call an object frame.

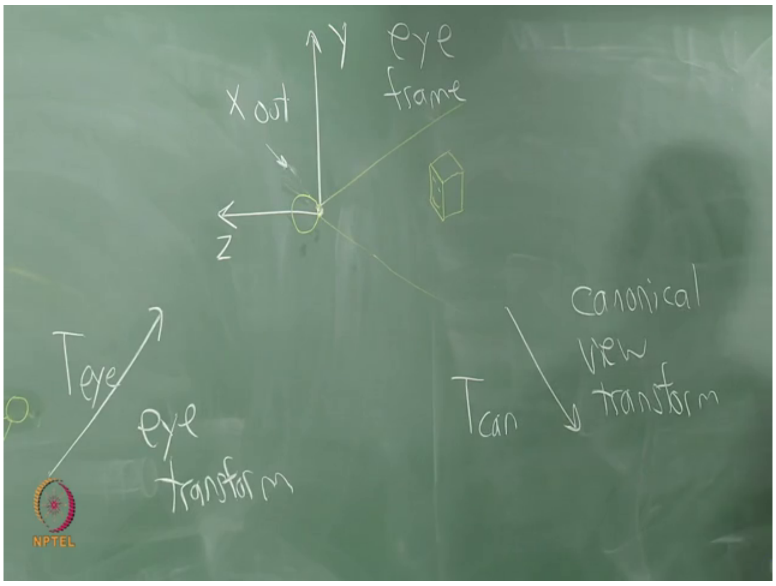
(Refer Slide Time: 07:00)



Local frame is more general from mathematics, it shows up in a differential geometry and related areas and object frame is very common in computer graphics as well. So, body frame is very common in engineering. So, pick what you like the same concept over and over again, and then the world frame ends up if you use the term local frame nice word for that is global it is a global frame. It is the kind of common frame of reference that you would like to put all of your bodies into right.

So, the next frame I want to get to is called the eye frame.

(Refer Slide Time: 07:49)



I will draw it like this, where z is going this strange direction, y is going up and I will say that x is out. So, x is pointing out at you and then I will have an eye here, as the name suggests too many things on the way here. So, x is coming out, but I have an eye here with you can imagine a pupil right there at the point the eye will have some field of view and somewhere inside of that field of view, will be this body perhaps maybe it will be outside of the field of view in my example I will make it inside the field of view, otherwise they always have to eliminate it will not be considered further.

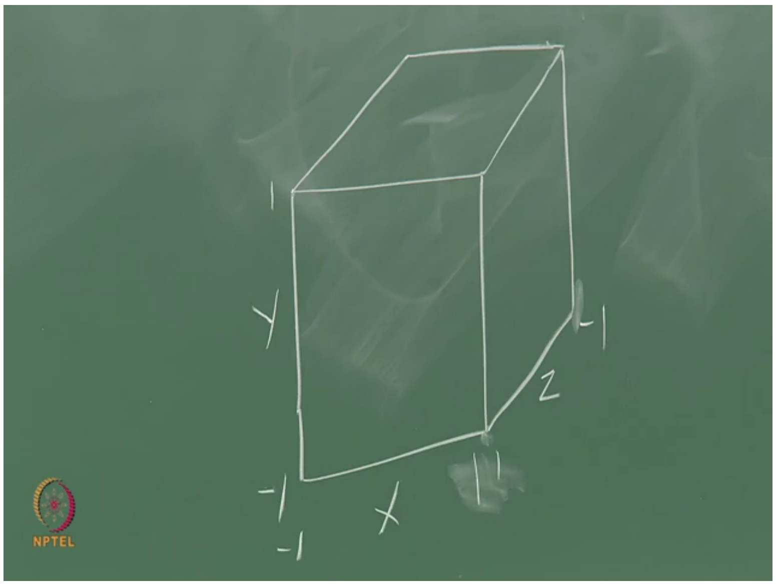
So, somewhere inside the this field of view of this eye there will be this body, and I am not really sure how to render it anymore after doing these transformations, but I will just put it in here anyway. So, we get the idea. So, I need to figure out this part I will call this the eye transform and also in computer graphics very often called viewing transform or camera transform, but all of these are transforms related to viewing. So, I do not maybe do not want to use that word, but you could call it a camera transform and what I call the eye could be called the camera, it is not a camera in the physical sense it is a kind of virtual camera, but you know when we are using virtual reality, then your eye corresponding in the virtual world becomes like a virtual camera alright.

So, it makes some kind of sense. So, I am going to call the eye here. So, what is essentially going on with this transform is that, somewhere in the world the eye is looking at this. So, imagine the eye itself is like a body that appears in the world right imagine we just grab onto an eyeball, and we put it in the world and we point it somewhere and we orient it some way, and then it sees something. We would like to then have a transformation, that puts the entire world into the perspective of the eye right.

So, it is not going to do any shrinking distorting other strange things, it is not even going to decide what is in the field of your what is not, I just want this transformation to express all of the points in the world in the coordinate frame of the eye. And then when I have gotten to that stage I have two more transformations left. So, this up here it is called the. So, the eye transformation puts you in the eye frame. So, we will do one more at this point and it will be called the canonical view transform, I am going to give some notation of these as well.

Let us see I will call this one T rb rigid body and by T I mean a 4 by 4 transformation matrix, just like we have been doing. So, homogeneous transformation matrix. So, T r b the I 1 I just called T eye e y e i. So, you remember where it comes from and I will call this one T can for canonical, what we end up with after performing the canonical view Transform, there is a kind of perfect cube centered at the origin x z y, just want to render the show the whole cube here, and I wanted it is coordinates to go from minus one to one for each of the cases.

(Refer Slide Time: 11:47)



So, I do not know where to draw these 1 minus 1 2 oops. So, this is 1 1 either way minus 1. So, these going minus 1 in the back up to 1, because z is coming outward in this case and I have x going to the right negative 1 to 1. So, the origin is in the center of the box I do not want to draw the origin and there to make more of a mess, but the origin is inside of the box. Why does the box have with 2 instead of with 1, it does not really matter and some people in computer graphics all are view that maybe it should be 1, they are always trying to find ways to simplify the algebraic computations to make them more efficient later right. In fact, not even simplifies not even the right word sometimes they will make it more complicated. So, that the calculations are efficient later when it is done in the GPU.

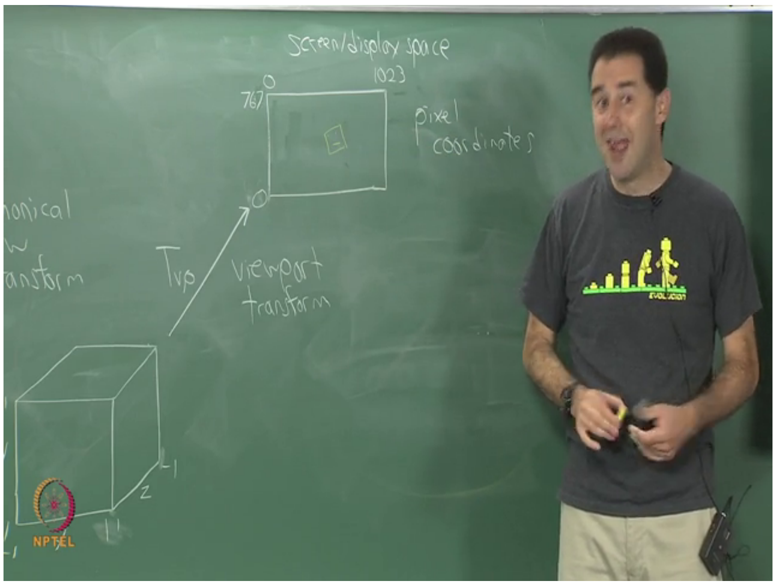
So, a lot of little funny things are going on like that, but I just want to say that what is happening when you go from this case to this case is the last remaining step left to figure out how objects should appear on the screen, is going to be done by a straightforward orthographic projection which basically means that once you get down to here, the x and y coordinates have been nicely figured out for each one of your points in the world frame.

So, from the perspective of the eye. So, you have figured all that out you have figured out the distortions due to perspective projection, and I will give the details of that, but that is what you have gone and figured out right. So, that is something that is very very far away, a tree that is very far away should look small on the screen right and a tree that is up close should look large on the screen.

So, those kinds of transformations are being taken into account in this canonical view transform, and then we are here all you have to do is take all of the transform objects they look like a like they are in a bunch of slices, here along the z direction, that are let us say they are they are x y plane or slices, and all the objects just project in a very easy way than to the this part the x y plane as if this were the screen, except the screen coordinates are usually not running from negative one to one there will be pixel indices.

So, that requires one more transformation which to me just looks like a change of units or something. You know it is very simple because you want your units in the end to be pixels. So, pixel coordinates. So, there is one last transformation, I would say the canonical view transform is probably the hardest of the ones that will do, the I transform also has something unusual going on, but this final one is quite simple which is just.

(Refer Slide Time: 14:40)



Converting from this canonical coordinates where everything’s running nicely x y and z from minus one to one to the actual pixel coordinates. So, this is called the viewport transform I will call it T v p, and we get up here now we are finally, in some kind of nice screen space. I am just writing some numbers there just to get you thinking about resolutions and so, this is the screen or display.

In other words you have pixel coordinates; you also have an interesting question if you want y to increase as you go down or do you want it to go in the other direction maybe it looks like this 0 to 767 for example, maybe it goes in this direction. It depends on how it gets addressed in whatever low level software you have. So, you may have it going in one direction or another, I believe for historical reasons y increases going downward, but that does not look like nice planar Cartesian coordinates does. It the historical reasons are one is well we do that for matrices, if you noticed right row one is up here and row two is down here. So, we count down this way also if we print lines out on a printer or an old punch card this increase is going down.

So, I think they went with that it is just one of these unfortunate cases, where that is not the way we would like it in order to make it consistent with the rest of mathematics. So, there will always be a little bit of confusion showing up in here as well you right any questions about this. So, that is the chain of transformations, unless the question is can you give me more details because I would like to give you more details next, but at this point this is what we have. So, again we start here define each one of your bodies in it is own local coordinate frame transform them using rotations and translations into this global world frame and include all the other parts of the model that do not change, then based on where the eye is that now the eye can be moving of course, the in virtual reality then we perform the eye transformation. So, that we get all of this model, in terms of the eyes perspective.

Then we transform into this canonical view transform. So, that a bunch of graphics pipeline operations can be performed with ease and efficiency before the final step of just easily projecting them all onto the screen, but we got the coordinates wrong because we put them in this nice canonical form. So, we can then do this final viewport transformation, and then we have exactly where my body is that I want to have rendered in terms of pixel coordinates wherever this might show up. And of course, there are discretization issues here, but I am not dealing with that yet either that is part of the graphics pipeline. So, it is as if these pixel coordinates are continuous variables here at this stage.

So, I do not mind if this corner ends up being 542.9973 so, forth right. So, that will be quite fine and then we will have to deal with problems of aliasing and anti aliasing to compensate for that or correct for that later as part of rendering, here we are just trying to figure out where does everything go right. So, we have this I am going to write out the chain of transformations algebraically after racing it. So, this is your last chance to ask questions, while this is on the board because I am not going to draw this again yes.

Student: So, what exactly (Refer Time: 18:33).

So, what you are doing in that stage here is, let see. Let us look at the start in the finish. So, the start here is I have brought all of the model in the world into the eyes perspective and that is what; that is you write here when you are down here you figured out what would go on the screen and you still have a z depth to, all these things that are going to go on the screen. So, you figure out the perspective transform you figured out when you do this transform and then it will become clear hopefully when we get to it.

So, in other words if there is something far away it will become smaller, when you are in this what when it is transformed into this space. So, there will be some objects let us say that are far back, they are at a let us see if they are far away then they are at some location like z close to minus 1. So, they are very far away, but I figured out what their size is going to be. So, that if I just simply delete the z coordinate, where do they show up in x and y. So, everything's been arranged at after this transformation.

So, that the scale and location of everything is appropriate based on the perspective transformation due to this eye and the geometry of this and I have to work that out for you. So, this is put in a form that, let us say graphics developers can use immediately and perform all the operations they need to determine what is visible from what is there an obstruction, what kind of anti aliasing do I need to do and so forth. So, so pixel shading operations things like that are ready to occur here.

I could have jumped everything, I could have taken the x and y part and just made that b pixel coordinates already and skip this step, but it seems that people like to work in this and do all of their computations in one coordinate system and then if someone decides to change the display, which as we plug and unplug monitors and do kinds of other things resize windows and things, then it is only a change of the viewport transform. So, it is nice to have this as an intermediate transform any other questions yes.

Student: (Refer Time: 20:39).

Yeah that is right when we get to this part we will have a viewing frustum, that is a region that it corresponds to everything the eye can see, and outside of that it just will not end up in the box, and then we will have the problem of a calling or clipping when we get to the rendering pipeline, which is figuring out what needs to be thrown away. But when we are up here in the eye part even though I have drawn the field of view of the eye it has not been used yet, that part will get used and there will be going to get to it in just a little bit, but there would be an ear plane and a far plane and that will make a frustum shape that will be transformed into a cube.

Other questions? You are asking the right kinds of questions you are just anticipating what is coming in just a little bit yeah.

Student: Canonicals (Refer Time: 21:23).

That is exactly right once we get to the world frame. So, so this operation T rb is applied to each and every one of your rigid bodies. So, they are all put together in this nice place here, you know it is now you have like a snapshot of the world in 3 D of this virtual world in 3 D or alternate world let us say in 3 D. Now you figure out what it looks like from the eyes perspective, you have all of it and then you carry it down here and the parts of the eye could possibly see are inside of this box, but you have not figured out occlusions yet, when you do this it just they are going to be a very simple homogeneous transform.

So, inside of this box, there could be some objects blocking other objects, but that is going to be handled by the rendering pipeline the graphics pipeline it is using this representation. And it will use the z values to determine, that one common technique is called z buffering it will use the z information for that to do it quickly to figure out how the pixels need to be ultimately rendered. But today I am just trying to figure out where should the points go if you start with a point over here, where does it end up when you get to this right yes.

Student: That why do not we use the world (Refer Time: 22:38).

There is a lot of there are lot of interesting questions like that, I suppose where I could say why do not we eliminate any one of these transformations right and so. Because you see that this is just a trend chain of transformations, I could you say why do not I just recognize which I am going to right as soon as I erase this I am going to multiply them all together and say look it is one big transformation.

So, each one of these intermediate representations, makes it easier to understand what is happening, makes it easier to develop and debug code, and it is especially true for the canonical view transform as I said because people might change the resolution and go out this way, it is helpful to have a coherent view of the world, one of the reasons why might be that you could have multiple eyes looking at the same scene right.

If you have a bunch of people together sharing a virtual reality experience, you may calculate this first and then have multiple eye transforms that is a simple answer and, but I think it is I think I would find it quite confusing to apply rigid body transforms that automatically take into account the eye view point.

I have very difficult to I very much difficulty visualizing that or imagining that it is mathematically possible yes.

Student: (Refer Time: 23:47) why we want to use (Refer Time: 23:48).

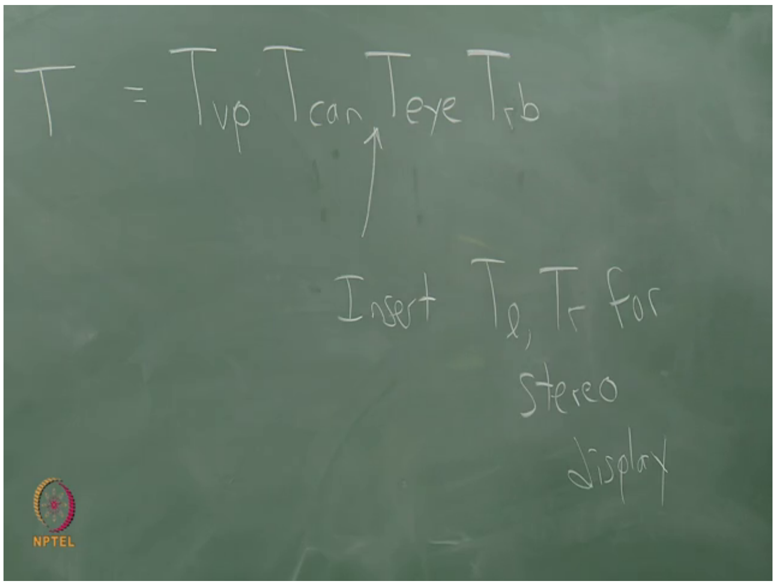
That is right many bodies are put into the world frame you going have to reason about all of those simultaneously taking into account this view transform. So, you could do that, but you know again it is all these things are mathematically possible, I just think that through some amount of experimentation and skill over many several decades, people have settled on these intermediate representations to make it easier clear for development to separate everything out piece by piece.

So, and professor (Refer Time: 24:23) is right that if you have many many objects, here in many many bodies, you are going to have to take into account the eye transform for all of them each and every time. This does all the transformations once and then you just have a single eye transform two to compensate which I find very very nice. I figure out where all of you are located in the room in the alternate world, and then I move over here and I look for me from this direction and I can figure that out very quickly and in virtual reality you are going to be changing the eye very frequently. In fact, you might look at a completely static world, and you start moving the eye around and it is very natural to have this global picture of what is going on right.

So, I am going to erase this now, and then start going into these particular transformations, I do not need to cover r T r b, but I do need to talk about the eye transform in the canonical view transform. The last one I will briefly mention, but it is very simple like I said it is more like a change of units or something like going from meters to feet. So, something like that right I think from your questions you are paying good attention I think you are anticipating what is coming next.

So, these are the right kinds of things to be asking. So, if we put all of these together.

(Refer Slide Time: 25:34)



$$T = T_{vp} T_{can} T_{eye} T_{rb}$$

Insert T_l, T_r for stereo display

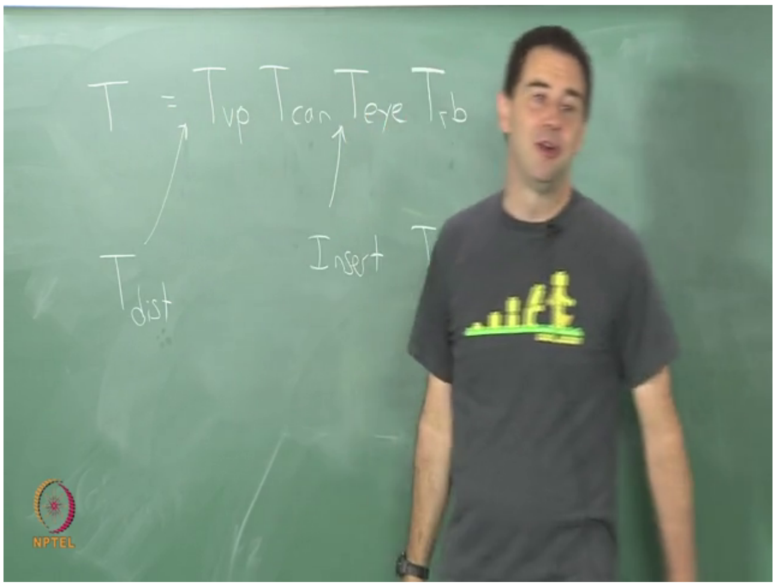
I get a single transformation let us say T, that is I got to go in the proper operation order which as I just erased it, it was here and we need to go from right to left across these. So, imagine it is still there. So, the last one was T v p the viewport transformation. So, one before that was T canonical, the one before that was T eye, and the one before that was T rb for rigid body which is only applied in the case of moveable bodies said right.

We had five coordinate frames that I showed you and there were 4 transformations between them right. So, there is one less. So, these are the 4 transformations. So, we just chained all of these together as homogeneous transforms, I should point out that one of these the canonical transform is going to have some non rigid body transforms inside of it. So, it will not be an example of a rigid body transform, but it will be an example of a homogeneous transform, it is just a little more general because it does some scaling due to perspective projection. So, it is not going to be rigid, but it is still homogeneous.

So, before I gave homogeneous only 4 rigid bodies just pointing that out to avoid potential confusion. This is standard computer graphics set of transformations there is a little bit extra that will come in for virtual reality, which I will provide at various points one of them is that in here we insert T l and T r for stereo rendering. Let us put stereo display I am not talking about rendering today right.

So, we have to perform some kind of shift that corresponds to your left and right eye. So, when I cover it today before I get to the left and right shifts, I will consider to be a cyclopean view you will be a cyclops with your eye right in the center between your right and left eyes, and then we will add the stereo part and another thing that happens was.

(Refer Slide Time: 27:48).



Another transform here which I will call t d is t and I would not get back to this one for a while this one T I put this here for distortion not distance, distortion which will compensate for optical distortions. So, we will do some final kind of warping that will account for warping due to a wide field of view lenses.

So, we will have to do some final operation, it is possible to move this T disk early in the pipeline it may be advantageous, but it is also more complicated. So, the easy fix is just to do it at the end, but there could be some sound reasons for trying to do it earlier and I am not sure it I think in most of these virtual reality pipelines and the current state, it is at the end here. So, it is one final fix to account for the distortion.