**Introduction to Computer Graphics**
**Dr. Prem Kalra**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Delhi**
**Lecture - 23**
**Rendering (Contd...)**

We will continue today on rendering and then start on ray tracing. So what we looked at yesterday was basically the shading models which were defined for polygon. Also, we looked at three models; the first model was this one which was the flat shading and it was basically the model where we considered computation of illumination per face or per polygon. The entire polygon was given one intensity so computationally it is definitely simpler but we observe the discontinuity from one polygon to another polygon at the boundaries which is for obvious reasons. And we also saw that there is a pronounced Mack band effect at the boundaries.

Then in order to smoothen this shape across the polygons we looked at the methods which employed interpolation of intensity. So, this one was gouraud. Just the Gouraud shading actually came around 1971 or so that was early 70s so you can see the type of contribution. It is just applying linear interpolation to the calculation of the shading. This was done by Gouraud in the University of Utah, lots of stuff was done in University of Utah and he got a PhD.

This Gouraud shading had limitations in terms of specular reflection. Since the interpolation is being done on the intensity values at the vertex so we are doing computation of illumination at the vertices of the polygon. Then we do the interpolation for filling the interior of the polygons using those intensities. Since the computation of the intensity has been done a-priori these speculars of the shininess could not be handled properly. So in order to account for proper handling of specular reflection another modification to the shading model that was done was phong. Here we can see that these specular reflections which are the shininess of the object could be modeled.

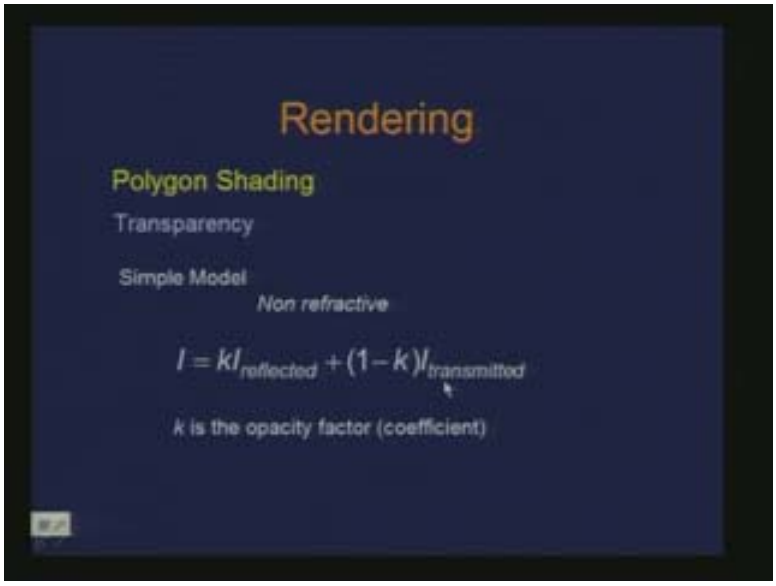
(Refer Slide Time: 00:05:34)


The side effect of this computation is where we are doing the interpolation on normals now instead of intensities at the vertices and then interpolate the normals within the polygon and compute the intensities for those normals. The computational effort is more in this case. Then we also looked at some of the limitations of using interpolation as a method to compute either the normal or the intensity. There are inherent problems of using interpolation.

Let us look at the transparency. We basically looked at the reflections such as diffuse reflection, specular reflection and there was no notion of transparency. Here is a very simple model of transparency which could be also employed in the context of rendering polygons. First of all we are assuming that the material which we are trying to model the transparency for is non reflective. For example, a transparent object like a glass is generally reflective so there is a deviation of the ray which is incident onto the object.

In the case of a non reflective transparent object if I am interested in getting the total intensity for the polygon or for the object to be rendered then the simple linear interpolation gain can be employed for calculating the final intensity. And this linear interpolation is doing nothing but some sort of a blending of the reflected intensity which is computed at the object and the transmitted intensity.

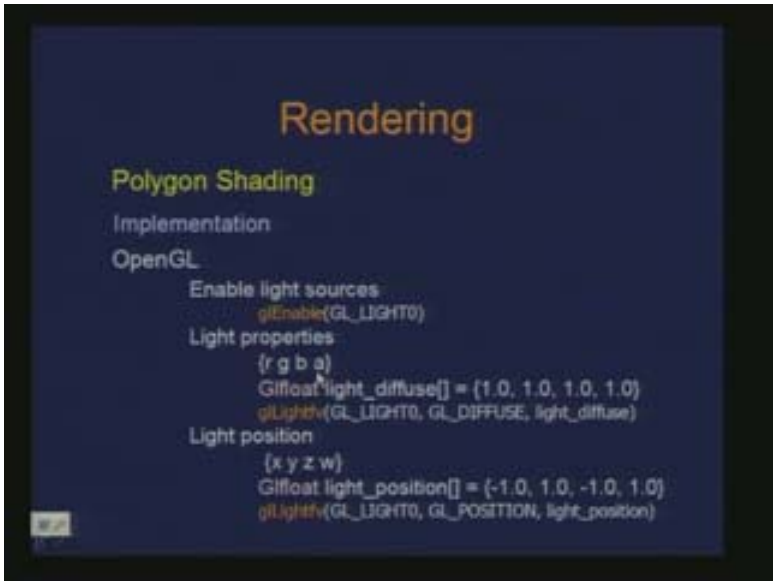
(Refer Slide Time: 00:09:56)




So what we are saying is that there is an object behind the transparent object and we need to just combine the intensity of the object which is transparent or translucent which is computed reflected intensity with its opacity factor and then whatever is transmitted as 1 minus k which is the rate for the transmitted intensity coming from the object behind. So we are just combining these two intensities to get the final intensity here and this blending is just governed by this factor k which is the opacity factor. If k is 1 then only this part is there, so the object is completely opaque therefore I is equal to the reflected intensity. And if the object is the completely transparent all I am going to receive is this intensity which is the intensity of the object behind which I can see through. It is a very simple model of transparency and again uses linear interpolation.

Later on let us look at the transparency model which may use a reflective material. Now let us see some of the issues which are more like implementation issue concerning what happens in the OpenGL.

(Refer Slide Time: 00:13:26)

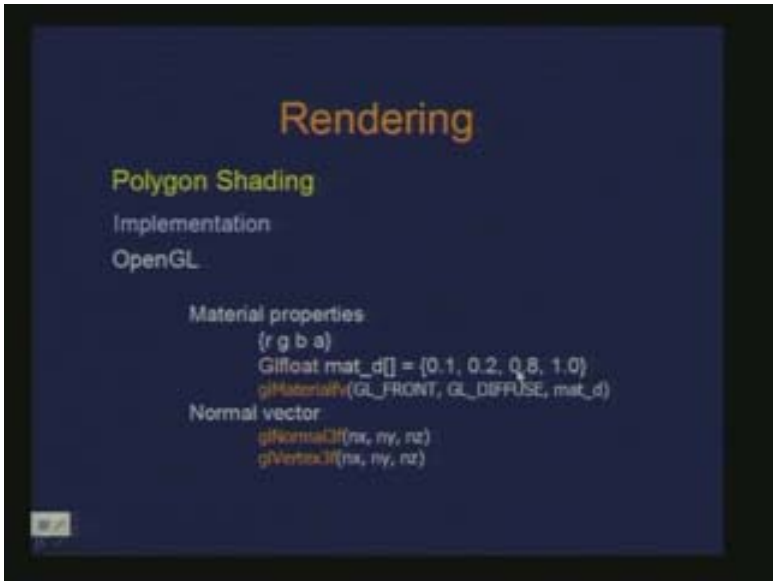



Here the idea is that, if you are allowing lighting or the light sources to be defined in the scene then you have to enable them. You should be enable light sources using GL_Enable and GL_LIGHT0 so could be number of light sources you can define and for a maximum of up to eight sources. And then the light properties concerning the color of the light are basically defined from the four value tuple r g b a where r is red color, green, blue and a is this opacity or the transparency factor. For having light this transparency factor could be debated but this is sort of a generic definition of some color attribute which could be associated to the light and which could also be associated to the object.

For instance, if I want to define the diffuse properties of the light source which could be, if I am looking at the white light source so all of these are 1 1 and this is opaque. So I can embed this light diffuse to the light source using GL_LIGHT fb. Similarly, I can define the light position using x y z w. So this w in fact indicates whether it is a point light source or it is a directional light source. If it is 0 then it is a directional light source. If it is 1 then it is a point light source. If I had divided by this w then I would have obtained the point at the infinity. So again I can embed this position and attach to the light source GL_LIGHT0 using this.

Similarly, I can have material properties defined through r g b a so I can define the material diffuse property where this shows the color of the object or the surface which is having more of blue components then I can attach this to the object of the surface. Hence GL front is telling you that I am associating this to a polygon and I am looking at the front of the polygon.

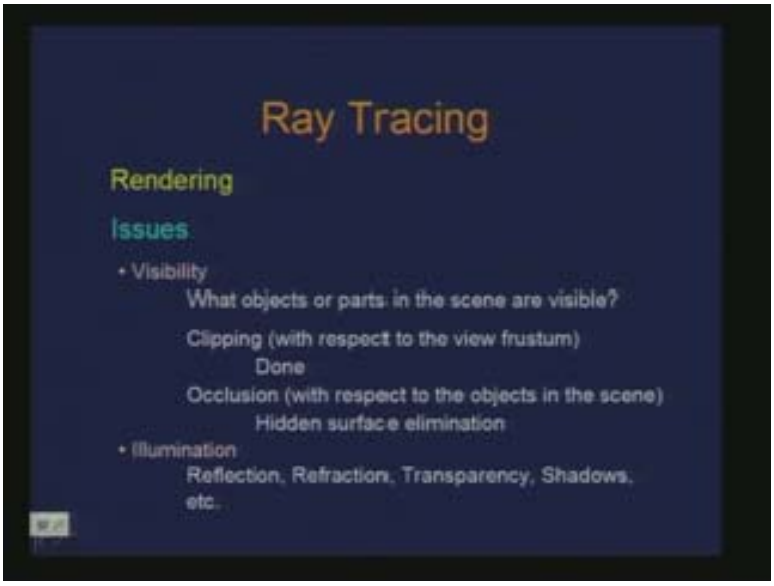
(Refer Slide Time: 00:15:43)




The polygon actually could be defined with two sides the front side and back side and I can attach material properties to both sides. But mostly I will be interested in the front of the polygon. Now we also require the definition of the normal vector which is just defined above the definition of the vertex which is just a vector (nx, ny, n0). So OpenGL supports flat shading model and Gouraud shading model in order that you invoke Gouraud shading model you have to say GL shade model GL smooth first you have to indicate that. So these are some of the implementations with respect to the OpenGL. There is an alternate way of defining these color attributes using color. You can also define a vector of color which is just r g b a.

Now let us look at the method of rendering. When we are talking about rendering in general there are two primary issues. One is that you want to have the visibility right. It means that basically to determine what objects or parts of the objects in the scene are visible. While we discussed about clipping we partially dealt with this problem. Clipping is with respect to the view frustum and there also you are trying to determine what is inside that frustum therefore that is the part which is going to be visible.

(Refer Slide Time: 00:19:02)

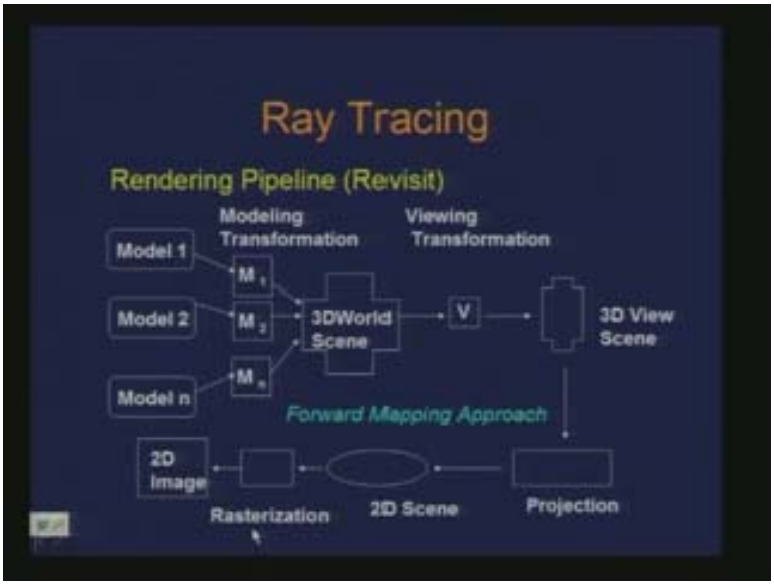



But there is also this visibility due to the occlusion. That means an object can occlude another object though they reside in the viewing frustum. That is what occlusion handling is or also the hidden surface elimination. What is the hidden surface part which should be eliminated or visibility determination? As we study the various methods of hidden surface elimination we can know more about occlusion.

But today the technique we are going to look at will actually answer much about what occlusion is. The other issue which pertains to rendering is illumination of the shading of finding the color or proper shade of the color. For this we basically looked at issues related to reflection, we did not go through reflection but we know that it exists, transparencies and there is also a possibility of shadows. All these are issues which would decide illumination or shade of this point on the object so that is also an issue for rendering.

Ray tracing is a method which combines these two really well. When we look at the rendering pipeline there was this model building where the world scene was defined then there was viewing in a particular eye coordinator system that gives the scene in that coordinator system then you have a projection which gives you a two dimensional scene then you have rasterization and then you get the two dimension. Therefore this is sort of a forward mapping approach. So if I need to answer the issues which I talked about like illumination and so on that could be done and this rasterization would again be process of shading where we are trying to decide the shade of the pixel.

(Refer Slide Time: 00:20:44)




That could be an integral part of rasterization and the hidden surface of the occlusion problem could be handled somewhere there where I tried to find out what object hides the other object and which is the front most object which I need to render. So everything is in this forward mapping Y. There is another way of looking at it and that is what ray tracing does which is something like this.

(Refer Slide Time: 00:26:21)


First of all try to emulate or look at what happens in reality when we see a scene. What you have is a light source or light sources, these are the objects of the scene and here is our image plane or the camera. So what happens is that these rays interact with the

objects in the scene and then they come to this image plane or viewing plane which eventually forms the image of what we want to see. So there is basically interaction of these light sources with this environment in the scene and whatever comes to this image plane or the camera is the resulting image. So, one could actually use this method of forming this image.

Now what is the problem in this? We are trying to say that there could be infinite number of rays which are emanating from the light source and not all these rays are coming into this image or viewing plane. So the effort which you are investing in terms of tracking all these rays and looking at the interaction of this race with respect to the environment or the seen is just too much. Only a portion of that is required which is seen here.

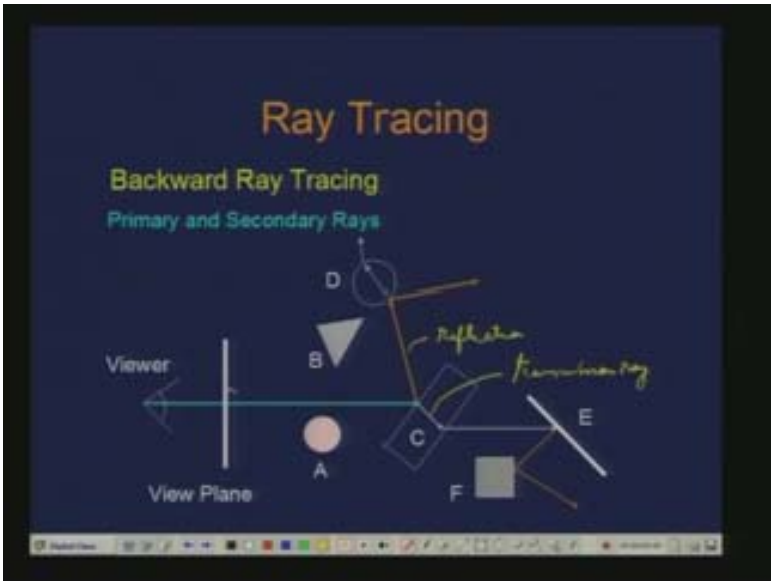
Now the question is can we do anything still capturing this process and avoid too many rays. So this is sort of a forward way of looking at the process of ray tracing. If your ray starts from the source or the sources of light they interact with the object so they are of course reflections from the object, there could be reflections within the object and eventually all those interactions whatever rays pass through this image plane or camera is what I see.

Now the question is that can I do something similar but avoid too many rays? So how about going the opposite way? Just go the backward way instead of having this forward interaction I just do the backward way.

Basically I look at each of these pixels so this is my image plane and this is the viewer plane, so I define the rays from the viewer with respect to each pixel on to the viewing plane and the image plane and then I look at what happens to these rays in this world look at the interaction of the light source or light sources with respect to wherever it is intersected and do the illumination computation. So in this way I am dealing only with the rays which are pertinent to me which are defined through this viewing plane or the image plane. That is what we call as backward ray tracing.

Here I have this green ray coming from the pixel of the viewing plane. Now there could be other rays which start due to this ray as a result of the interaction with the objects and these are called as secondary rays. This is the primary ray and these are the secondary rays and these are nothing but here we have the reflection ray whereas this is the transmission ray or reflection ray so these are nothing but the secondary rays.

(Refer Slide Time: 00:32:57)




So what we saying this ray comes here intersects here with this object and depending on the type of the object we have, this an object which is semi transparent with some reflectivity so it reflects something transmitted and then again it is transmitted it comes here it is reflected again reflected again and so on. And eventually we assume that when it sort of escapes after having interacted with the objects it goes to the light source. Therefore if I have to compute the intensity I have to trace these rays, this primary rays goes there goes there and goes there.

So now at this particular point I will see the interaction with the light source wherever it is and that contributes to the illumination at this point. Also this point also contributes because of this reflection. Now I will look at the interaction of the light source or the light sources at this point and then it escapes to the environment and wherever it goes. Therefore I am trying to avoid the infinite number of light rays which is computationally intractable. Now what would have happened is that this point, if there was no secondary rays so the illumination at this point is nothing but interaction of the light source or the light sources with respect to this point.

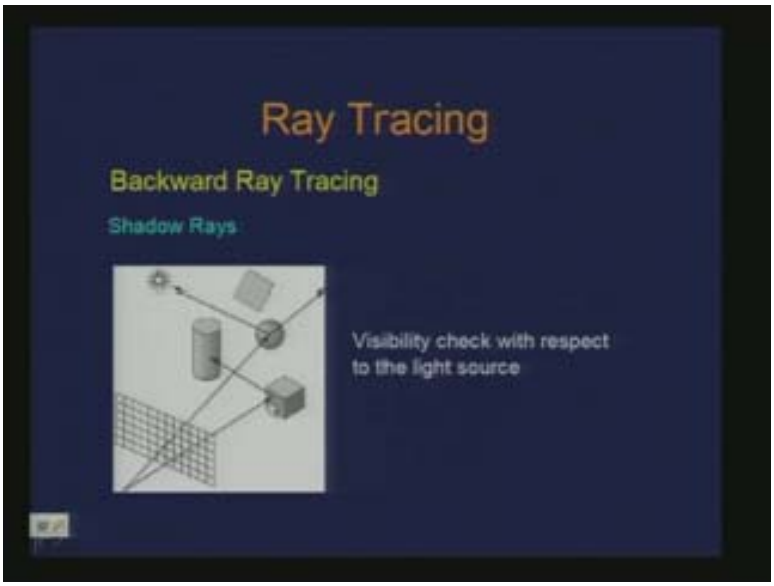
Now the fact is that there is a possibility that this object is a reflective object so there is a reflected ray and there is again a reflected ray. And if I just see the reflection of this point onto this it is a mirroring effect all it is saying is that I need to account for the intensity at this point computed through the light sources I had at this point.

In a forward way what would have happened is that the light source would have given you some intensity at this point and due to the reflection it would have given intensity at this point as well then there is also a local illumination due to the light source at this point. What we observe is a pattern which is in some sense recursive in computation because I am talking about computation here due to the computation here due to the computation somewhere else so there is some sort of a T formation. Therefore anyway at

the first instance let us ignore these secondary rays. There is also another type of secondary ray. These were actually reflection rays and transmission rays which is due to the characteristic or the properties of the object. Now there are also shadow rays.

What is a shadow? If this is the light source I have and if I am trying to compute the intensity at this point with respect to this ray what I observe is this object actually comes on the way between the light source and this object which allows the shadow to be caused.

(Refer Slide Time: 00:34:48)




So when you see shadow of this here it is due to the fact that this is occluding the part of the table this with respect to the light source. In some sense it is a visibility test with respect to the light source. So the fact that the object may come between the light source and this object would cause a shadow here. So shadow ray is nothing but a sort of secondary ray which starts from this point of intersection which I get from the primary ray to the light source. And if I observe that there is something intercepted for this ray which is from this object to the light source I declare that point to be in the shadow. Thus the computation of shadow is very easy. So these are also in some sense secondary rays just like we have reflection rays and transmission rays.

(Refer Slide Time: 00:40:00)

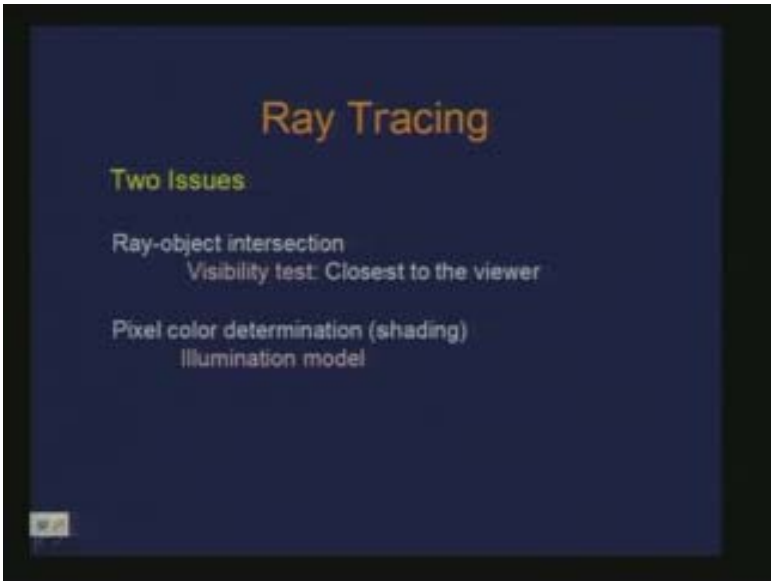

Actually we were trying to resolve two issues pertaining to rendering. One is visibility determination that we would like to render the point which is front most. Second is we want to compute the illumination at that point. So let us ignore the secondary rays and try to see what happens if we just consider primary rays.

Primary rays are nothing but the rays which start from the viewer or the camera with respect to each pixel on this viewing plane. So in some sense I am just casting these rays from the viewer with respect to each pixel here and then observe the intersection of this ray with respect to the environment or the objects I have in the scene that pick the closest point.

If I am talking about this ray, this will give you an intersection here, this will give you an intersection here and so on and I pick this point. This is the front most point and therefore this point needs to be rendered. So this is just a variation of what we are looking at ray tracing where it is just a single or one level of ray tracing just look at the ray emanating from the viewer passing through the pixel of interest and gives you the closest point and that is called ray casting.

If I am just trying solve the problem that what object needs to be displayed at this viewing plane that is the question I am trying to answer. So it is achieved by just casting rays from the viewer from all the pixels I have in the viewing plane and observe the point which is closest with respect this ray. So, if I am talking about this ray so this is point which is closest. I will get a point of intersection here I will get a point of intersection there and possibly with other objects but this is the point which I need to display. Therefore I will capture the color or the material properties of illumination with respect to this point and the rest are of no interest to me. I am basically solving the problem of visibility determination.

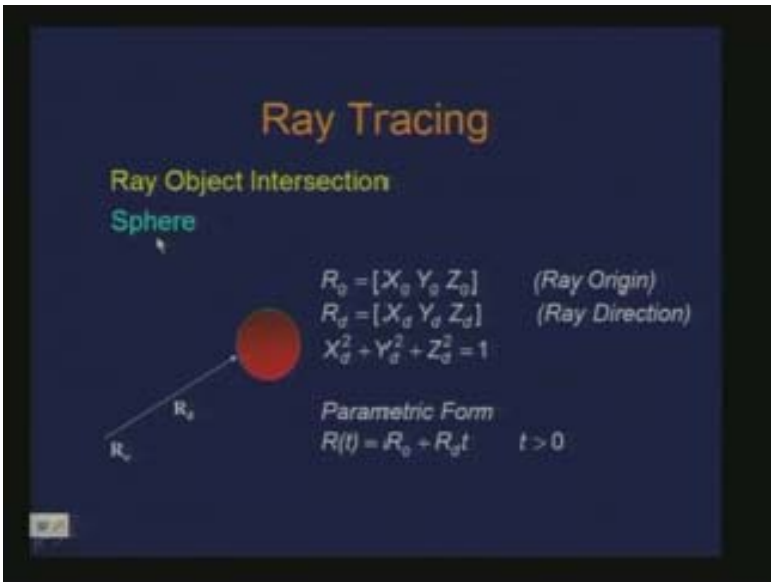
(Refer Slide Time: 00:40:54)




Thus, once we have done this then it is a matter of just applying illumination model to the point of intersection. So if I discard these secondary rays then the ray tracing to rate for one level or ray casting is just easy. Now if you look at the two issues m ray tracing or ray casting one is the ray object intersection that is the visibility test which gives the closest point of the object to the viewer and the other thing is the pixel color determination or the shading which is coming from the illumination model.

Ray object intersection:
First of all it depends on what kind of object you consider. For certain object is is a trivial thing to find out the ray intersection but for certain objects it may not be so easy. Here are some of the objects with which finding intersection is relatively easy. If I have the object as sphere, first of all I need to define the ray in its parametric form starting from some origin defined as $R_0$ and direction $R_d$ so this is the ray definition I have. So parametrically I have R(t) is equal to $R_0$ plus $R_d t$ and clearly I am going to consider t grater than 0.

(Refer Slide Time: 00:44:25)

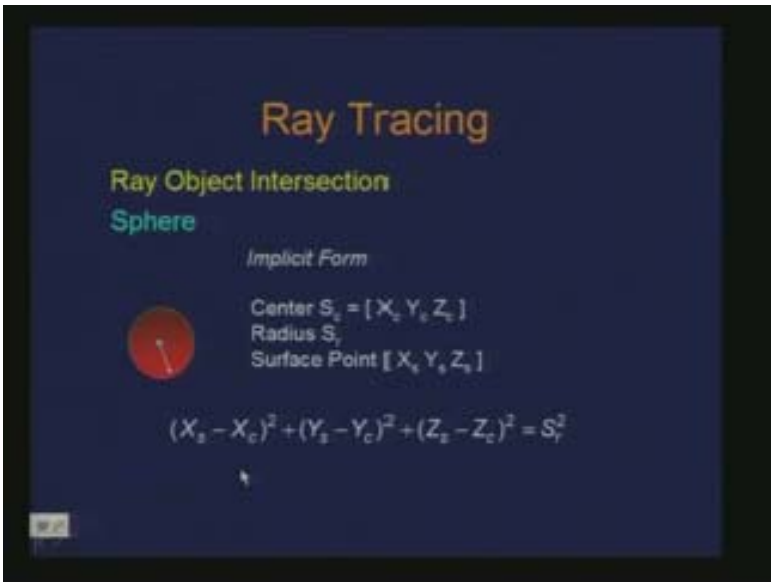



This is normally my viewer. Basically what we need to consider is the half ray definition like half planes we have. So here I am talking about the semi infinite line I am not concerned about this part because this is the starting point. If I am looking at also t less than 0 that means I am talking about the objects which are behind this behind the viewer I need not consider those. So this $R_0$ which is the viewer decides that I am going to see ahead of that therefore it is always t greater than 0 which I need to look at. Now for the purpose of convenience we can always define the Rd mod to be 1 so it is a unit vector so the direction vector of the ray is a unit vector.

Now, given the ray definition in parametric form we are going to find the intersection with the sphere. So what should be the representation of the form for sphere which is easy to use for finding the intersection?

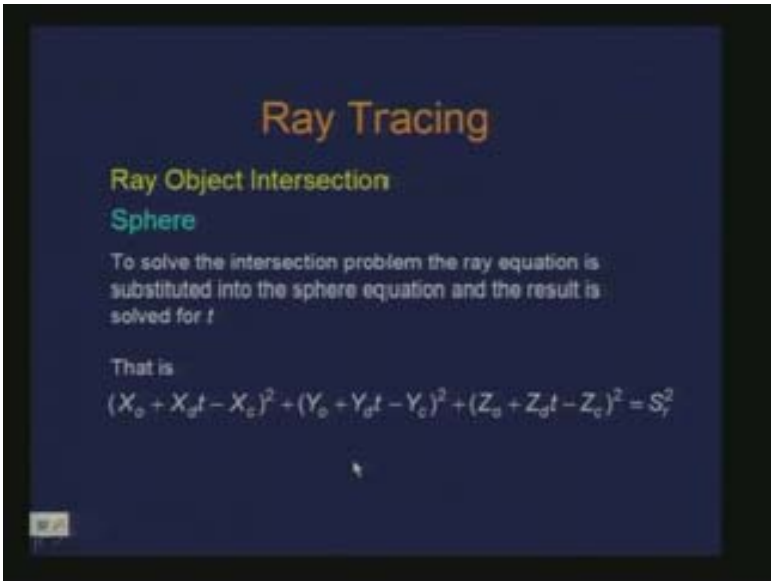
Devising this as a parametric form of ray my ultimate goal is to be able to solve for only for the parameter. How can I do that? If I have the sphere equation in implicit form where I just substitute the equation of the ray and solve for t. If I have a parameter definition of sphere I am in trouble, you have more variables. Basically we consider an implicit form of the sphere which is given through the center of the sphere somewhere here $S_c$, the radius is $S_r$ so for any surface point $X_s$, $Y_s$, $Z_s$ it is in the equation.

(Refer Slide Time: 00:46:23)



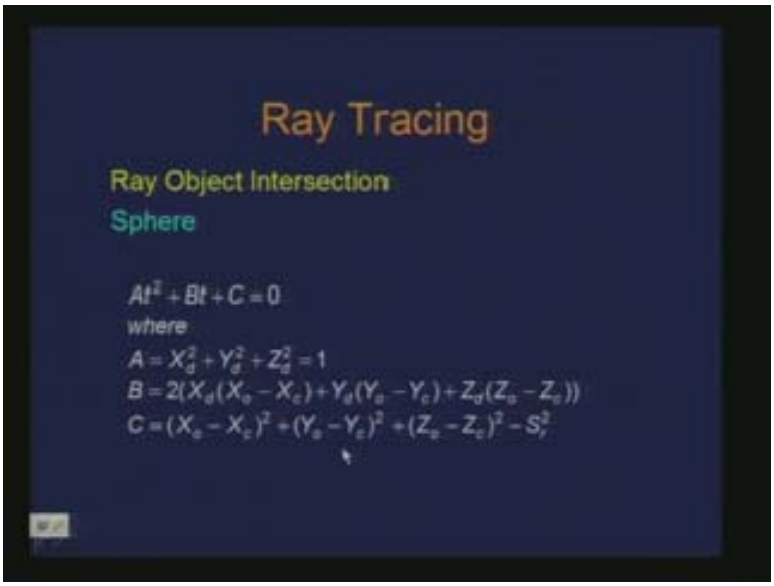

Now it is just a matter of substituting for the $X_s$ and solve for t. We substitute for $X_s$ as $X_0$ plus $X_{dt}$ for all other components which gives a quadratic equation and we try to solve for t.

(Refer Slide Time: 00:46:55)



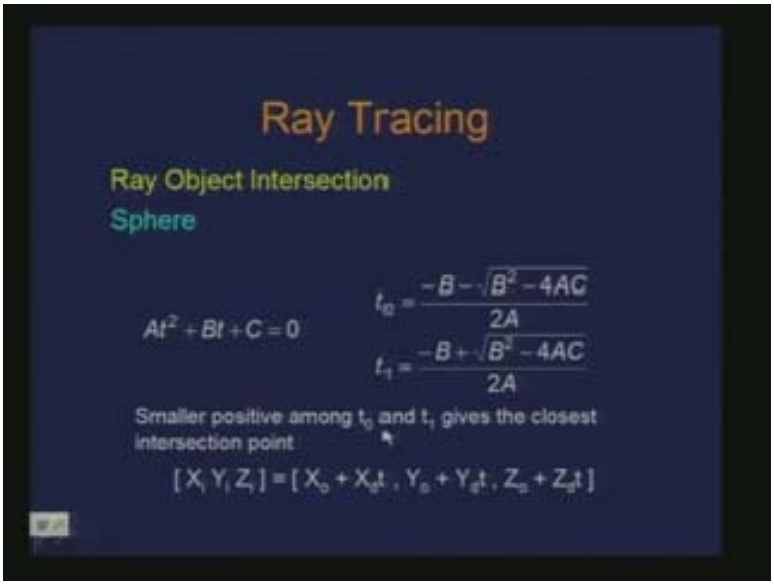

I have the equation in this form now as; At power 2 plus $B_t$ plus C is equal to 0 where I can find out A B C in terms of $X_d$ $X_0$ $X_c$ and so on which gives me two values of t as $t_0$ and $t_1$ solving this equation.

(Refer Slide Time: 00:47:17)



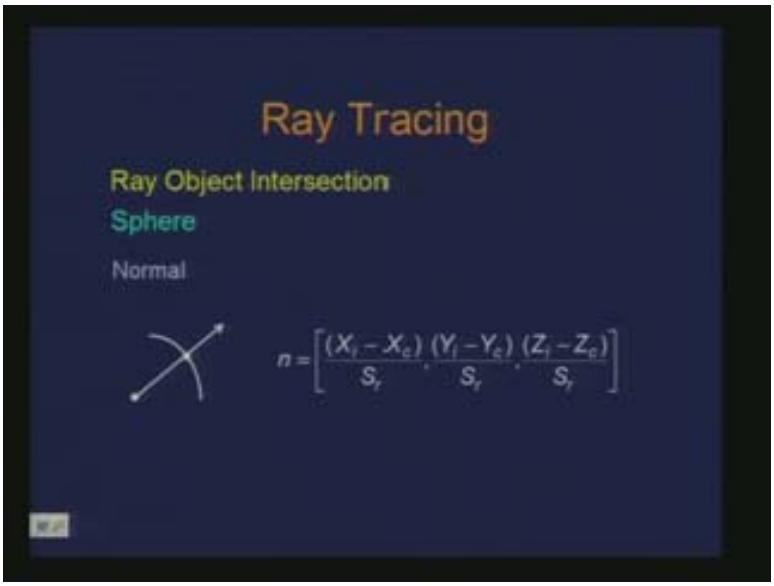

So obviously I am going to consider the smaller positive value as I am looking for the closest point. So the minimum of these $t_0$ and $t_1$ should be the solution for t and just substitute the value of t there to get the point of intersection which is $X_i$ $Y_i$ $Z_i$.

(Refer Slide Time: 00:48:35)



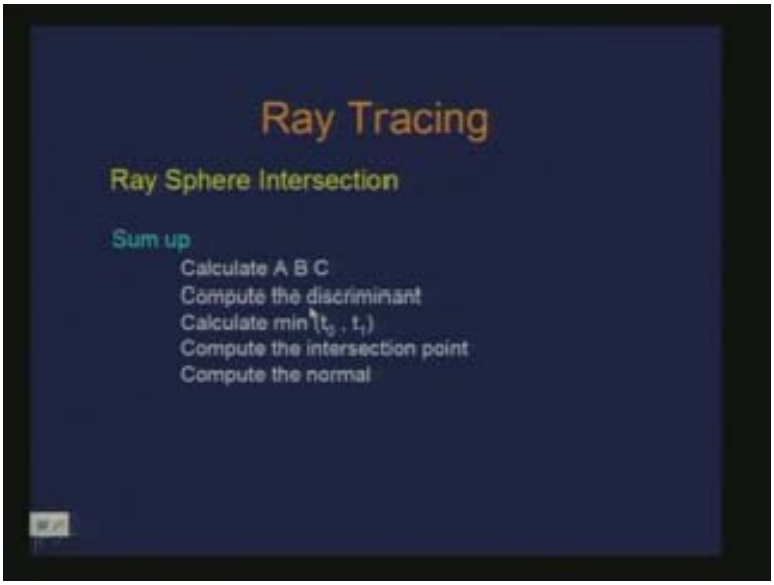

Now the question is, having found the point of intersection on the surface that locates the point which I will possibly be rendering. I will also be interested in finding out normal at that point because I need to do the computation of illumination.

(Refer Slide Time: 00:49:23)




The normal is again trivial. This is the sphere I have, this is the point of intersection which has been obtained and this is the center of the sphere then the normal at this point is nothing but this. It is a vector along the radius. So once I have the found out the normal I can do the rest of the computation for illumination. So just to sum up finding out the intersection with the sphere it is a quadratic equation which gets for coefficients as A B C so I can already define those terms which I use then I compute the discriminant and then calculate the minimum of these $t_s$ compute the intersection point then compute normal.

(Refer Slide Time: 00:49:48)



Having seen this computation with sphere now we can also do the similar thing for other minor projects. This is like one level ray tracing or ray casting. We are ignoring the secondary rays then we will incorporate the secondary rays and see how these computations are done. So basic computation of course is ray object intersection. The rest is how do you propagate the illumination or accumulate the illumination from all the intersections which are there while tracing the ray.