Virtual Reality

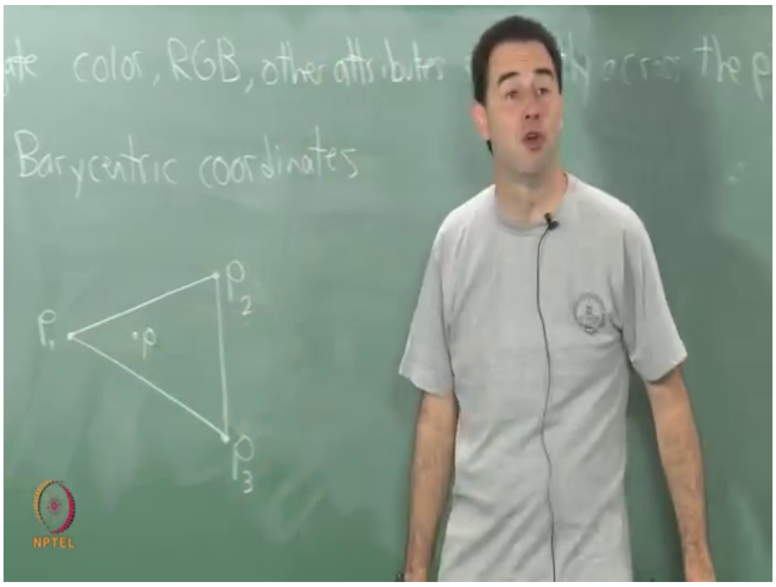
Prof. Steve Lavalle
Department of Multidisciplinary
Indian Institute of Technology, Madras

Lecture – 62
Visual Rendering (pixel shading)

Hello, welcome back. In the last lecture we went through on the basics of visual rendering. So, the rendering part generating the pixels the RGB values on the need to be presented on the display just from a general perspective that is common in computer graphics. I have also started to talk about the particular difficulties that arise in virtual reality when these pixels are rendered to a display that is mounted onto your head for virtual reality.

So, as I am continuing onward we were talking about object order rendering and so in object order rendering we are iterating over the triangles and for each triangle we are figuring out how to shade the pixels and so, I gave you a lot of the steps of that already now, I am continuing onward.

(Refer Slide Time: 01:13)



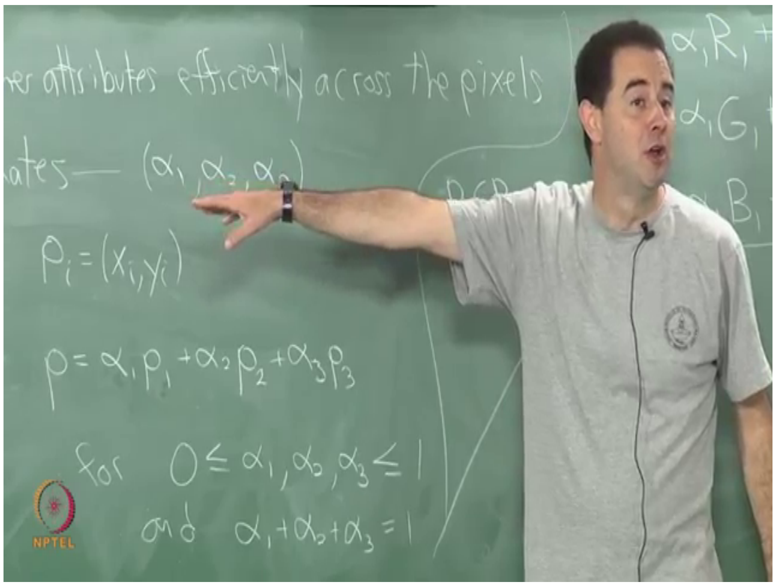
So, the part that I have not talked about is how to propagate color or more particularly RGB values or other any other attributes which may be surface normals or textures, I will give some examples of that efficiently across the pixels.

So, one very convenient way to do this which is commonly done is to use on a barycentric coordinates. Have you seen barycentric coordinates before they may not be completely common in terms of what you normally see in engineering I have seen them come up several times in my 20 or more years of research and they were my favorite coordinate systems?

So, the end of being useful in many settings particularly when you need to do some kinds of interpolations over 3 dimensional spaces or higher dimensional spaces it gives you a way to interpolate or combine information in a very natural way across a triangle in if it is a 2 dimensional surface or are more generally across to a D dimensional simplex using D plus one points.

So, I am only going to give the case for triangles because that is what is relevant today. So, I am supposed we have a triangle that is formed by 3 points that correspond to its vertices. So, P 1 P 2 P 3 and now, I want to consider some interior point not necessarily at the center some interior point P and I want to give it a coordinate system that expresses the location of P in terms of the coordinates of P 1, P 2 and P 3.

(Refer Slide Time: 03:35)


er attributes efficiently across the pixels
ates — $(\alpha_1, \alpha_2, \alpha_3)$
$P_i = (x_i, y_i)$
$P = \alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3$
for $0 \le \alpha_1, \alpha_2, \alpha_3 \le 1$
and $\alpha_1 + \alpha_2 + \alpha_3 = 1$
$\alpha_1 R_1 +$
$\alpha_1 G_1$
B_1

Each one of these P 1, P 2 and P 3 if I write it as P I is going to have coordinates X i, Y i, so, right. So, it has standard on Cartesian coordinates and then I can represent the coordinates of P in the following way I can write P equals alpha 1 P 1 this is a 2 dimensional vector here and this is a scalar coefficient alpha. So, it is scalar times vector plus alpha 2 P 2 plus alpha 3 P 3, my alpha parameters all lie between 0 and 1 and they must sum to 1.

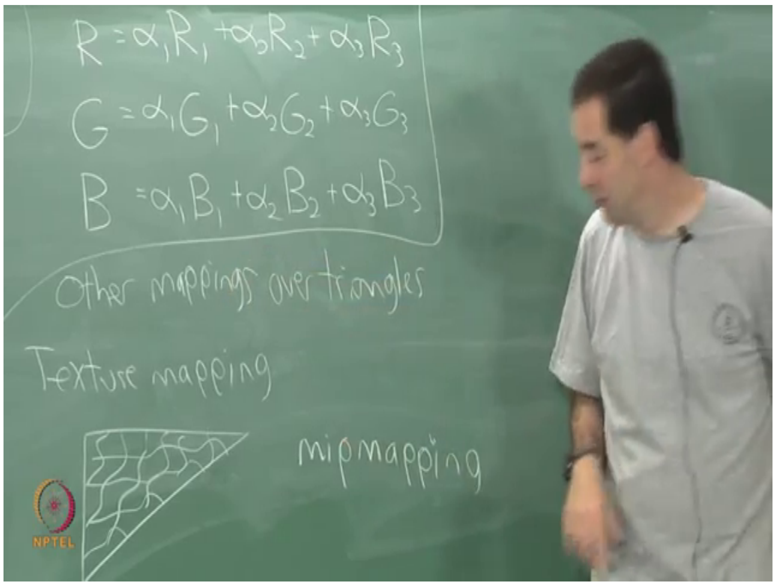
So, alpha 1 plus alpha 2 plus alpha 3 is equal to 1. These alpha values constraints on them should look familiar from probability theory. So, that we are not talking about probabilities here today, but it is the same constraint. So, you can imagine them as looking like probabilistic weights if you like. Some interesting special cases if we decide to write P in this way it looks like P is just a simple interpolation of the coordinates of these three points P 1, P 2 and P 3 and if we make alpha 1 equal 1 for example, then the other two coordinates have to be 0, right the coordinates are the alphas, ok.

So, those are the barycentric coordinates should maybe make that very clear on the barycentric coordinates are these alpha 1, alpha 2, alpha 3. So, the coordinate 1 0 0 would correspond to P 1, correct and 0 1 0 corresponds to P 2, 0 0 1 corresponds directly to P 3 what happens if I make the P 1 component 0 and I let the other two components be whatever we like as long as they add to 1 and are non negative. So, that should correspond to the edge between P 2 and P 3. So, they are beautiful properties of this, right.

If we pick a point in the interior that corresponds to all three of the coordinates being nonzero if we allow one of the coordinates to go negative it turns out that will send us outside of the triangle. So, if you do not satisfy the constraints your all coordinates go negative, you can actually parameterize the entire plane by allowing the quarters to go negative, but by having all three positive you know that we are we are picking a point that is inside of the triangle.

And, so, now, suppose that we have RGB values that we have calculated using our share shader right, we have shading methods that we have talked about we calculated RGB values at these particular vertices we like to figure out now how to color the pixels that have their centers inside of this triangle. So, these are particular values P and if I want to figure out now, what the value should be I can just do simple interpolation.

(Refer Slide Time: 07:03)


$R = \alpha_1 R_1 + \alpha_2 R_2 + \alpha_3 R_3$
$G = \alpha_1 G_1 + \alpha_2 G_2 + \alpha_3 G_3$
$B = \alpha_1 B_1 + \alpha_2 B_2 + \alpha_3 B_3$
Other mappings over triangles
Texture mapping
mipmapping

So, I get let us say R, G and B at some particular point P right I had like to figure out RGB values at P and again P is some particularly chosen point that happens to be the center of a pixel where I would like to render and figure out what the what the value should be.

So, I just apply the barycentric coordinates and use the R, G and B values for the vertices these three vertex points and same for G and B. G 1. So, all we are doing is taking the RGB values that correspond to these vertices and just using the barycentric coordinate coefficients, right. So, these have to be calculator they are not very hard to find for a given point and then we use those coefficients to linearly interpolate in this sense, between the RGB values of the vertices to get any point in the interior.

So, that is a very quick way to propagate the information across a let us say rather large triangle without having to do shader calculations for each and every point that we sample inside, makes sense. Alright questions about that? Should also just remember barycentric coordinates is a very convenient way to do interpolation over triangles and I said 1 dimension higher you can you can you can interpolate over 3 dimensional volume using four points which ends up forming a tetrahedron of some kind and it works in higher dimensions as well. So, it is a very powerful idea very practical for a lot of engineering applications finite element and methods and all sorts of things.

There is other kinds of mappings you might want to do in addition to RGB. So, let me give some other examples and then I will go into some of the specific problems that virtual reality causes for these types of techniques; so other mappings over triangle. So, one common kind of mapping is texture mapping. So, it may be the case that over my triangle I may have some kind of textured pattern that needs to appear and we again can use the barycentric coordinates to index into an image of the texture and then fill in the pixel values according to the texture.

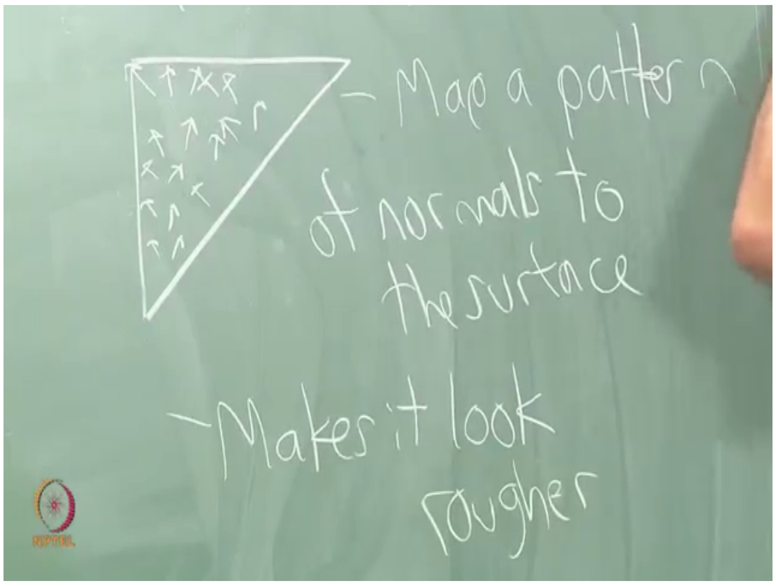
So, if you want to do implement is a kind of texture that looks like a let us say a carpeted floor then you would use an image of that and then propagate that across the triangles and then you have some delicate issues to worry about if you have a bunch of neighboring triangles they are all connected together in some way and you need the texture to correctly propagate across those. So, you also have to be very careful with regard to that because the texture patterns might not match up very well on the edges. So, there is some special techniques for handling that as well.

Another issue that comes up in texture mapping that is challenging is that what happens when I render this at different scales. So, it may be that there is some kind of texture or pattern appearing here and it has some kind of wavelength to it you may imagine and then if this triangle is very far away from the viewing location then the triangle be very small and then the pixelation may interfere with this textured pattern and so, there is a very well known and widely used technique called mipmapping which involves storing these texture images at different resolutions and optimizing them for each resolutions and then when they are rendered in practice different-different levels of this resolution are selected that are appropriate based on the essentially the number of pixels that are that you have at your disposal for rendering the image.

So, the appropriate resolution of the texture will be selected and some kind of a sophisticated interpolation between the different resolutions of textures that have been stored will be used. Generally, there is a logarithmic number of these resolutions in other words there will be a full scale resolution of the textures stored and then a half scale and then a quarter scale and so forth down to smaller ones. So, mipmapping I am not going to go into the details of it, but it is a very powerful technique for overcoming some of the difficult aliasing problems that happen when you have a very small number of pixels to represent a texture, right. So, that the pixel wavelength if you like gets close to the texture wavelength there ends up being some interference between them.

So, you can map anything you want you can put text as well right on to a surface you can even use texture mapping in your in some game engines, you can map an entire video onto a surface if you like if that will just have the texture keep changing from frame to frame. So, you can make a virtual a TV screen in virtual reality. So, a couple more examples and then I will start to talk about these virtual reality challenges.

(Refer Slide Time: 13:00)



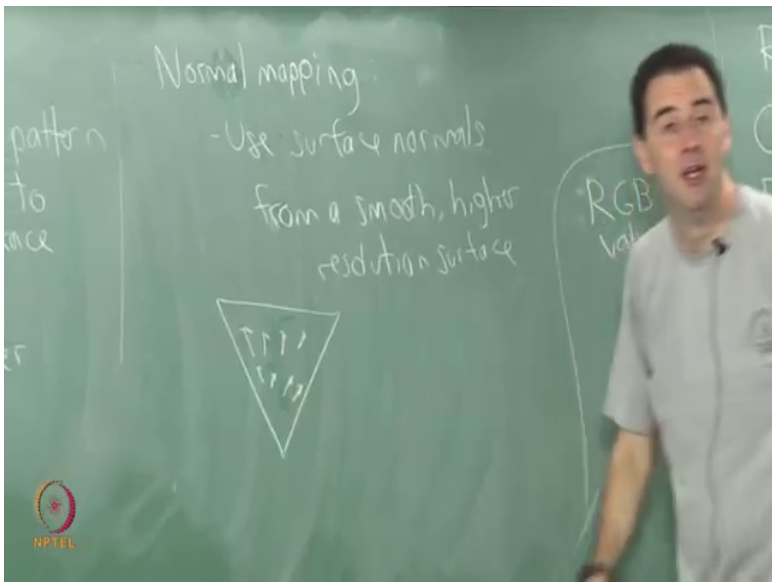
Another technique is called bump mapping. In this case and here is my triangle. You can play certain kinds of tricks or implement certain kinds of tricks by varying the surface normals across the surface in some way. So, it is another kind of hack or trick that will generate a desired effect. So, in this case bump mapping generates an irregular pattern of normals across the surface.

So, if it is a single triangle the normal the normal should be the same across the entire triangle correct, but you can instead make an artificial normal just make some perturbation to it just do something let us say random perturbation or do it in some systematic pattern if you like as you move across the surface. So, in terms of the barycentric coordinates you could have some kind of pattern for the normals, if you do this it will make the surface look rough. So, map a pattern irregular or a regular pattern of normals to the surface and then if you remember these shading models that I talked about in the last lecture they depend on the surface normal, right.

So, if I make a kind of fictitious surface normal then that will affect the shading and then that can make what would be a flat surface look rough even though I did not change the geometry. So, I have not changed the depth of the pixels in any way it is still a flat surface, but just by understanding the way shading works because it is using the star product with the normal I will affect that and make a surface look artificially rough. So, map a pattern of normals to the surface or to the triangle surface and then it will make it look rougher.

For example, I may be able to take a smooth sphere and then triangulate it and then across the triangles I may make a rough kind of patterns so that the sphere looks bumpy and may look like an orange for example, if some kind of citrus fruit with a very rough texture around it. So, you may be able to play a trick like this by varying the normals. So, that is bump mapping.

(Refer Slide Time: 15:30)



Related to this is more generally normal mapping I suppose I can consider bump mapping as one special case of normal mapping, but I am going to give a different sort of particular case I guess they are all playing tricks with the normals bump mapping or this example of normal mapping that I will give and what I can do is use surface normals from a smooth higher resolution surface.

So, for example, I may have a triangle we were always talking about and perhaps, this triangle in terms of the depth it needs to be a linear patch, but I could imagine that this triangle is approximating a curved surface, it is coming out of the board maybe it is part of a sphere in which case I can just keep track of the fact that the normals should be mapped along a sphere. They should correspond to the normals of a spherical patch. I am not sure exactly how to draw that here because I do not have enough dimensions, but you know they were just drawn in some nice way so that this appears to be curved if I do that I can make a flat surface look like it is smoothly curving, does that make sense?

So, I could generate what looks like a spherical patch, but it may be done with a very small number of triangles. So, this is a way to hide some of the limitations of the geometry I mean when you use a small number of triangles for efficiency, but I make it look better by actually keeping the normals correct I have the depth incorrect, but I have the normals, correct? Alright, so then I can use Latvia Lamberson shading model for example, across either of these bump mappings or normal mappings to give this effect where that is one of the depth cues that you have, right is shading across the surface. So, that is giving you a sense of the depth that is artificial here. So, it is relying on that fact that you will be fooled by that. Questions on this?