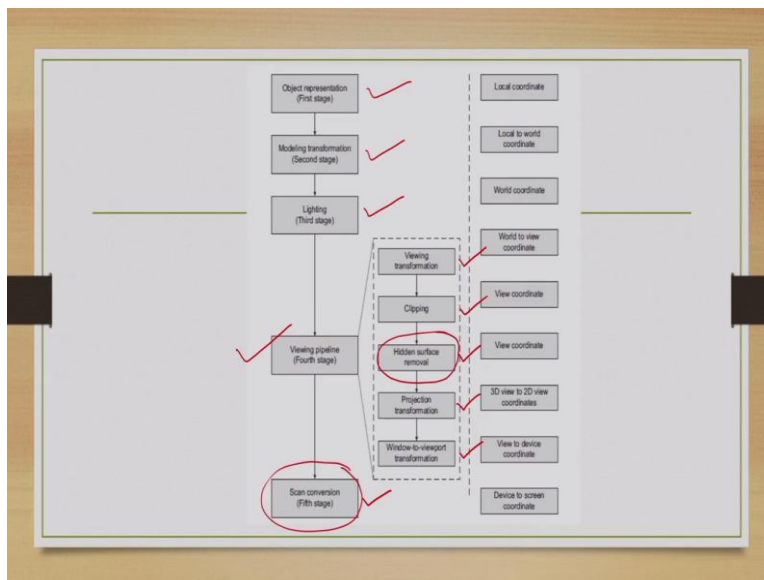


**Computer Graphics**  
**Professor Dr Samit Bhattacharya**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Guwahati**  
**Lecture No 15**  
**Shading Models**

Hello, and welcome to lecture number 15 in the course Computer Graphics. As usual, we will start with a quick recap of the pipeline stages that we are currently discussing.

(Refer Slide Time: 00:43)



So, as you may recollect, there are five stages in the graphics pipeline. The first stage is Object Representation; the second stage is Modeling Transformation. The third stage is Lighting or assigning color to the surface points. The fourth stage is the Viewing pipeline which itself consists of five sub-stages namely Viewing transformation, Clipping, Hidden surface removal, Projection transformation and Window to Viewport transformation.

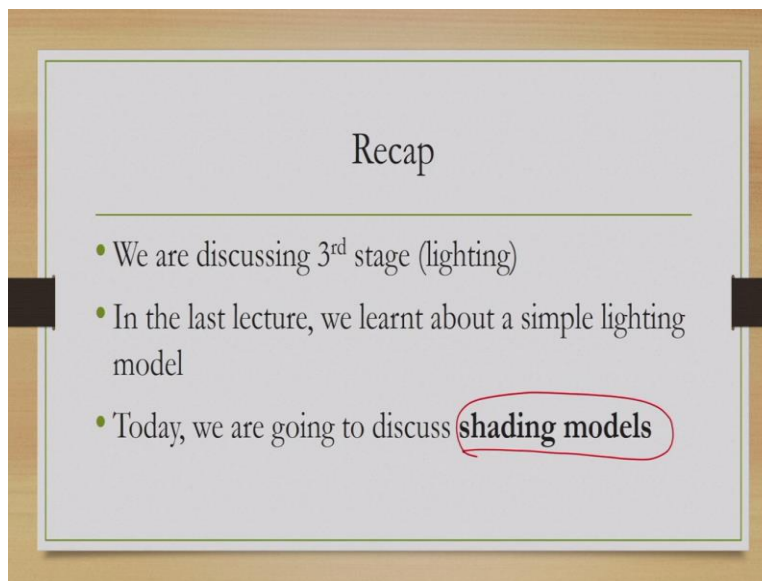
The fifth and final stage of the graphics pipeline is Scan comparison. I would like to emphasize here again the fact that although in this lecture or in this course, I will be following this sequence of stages, but in practice, it is not necessary to follow this exact sequence. So, when a graphics package is implemented, you may find that some stages are coming after other stages although in the sequence that I have discussed. They are actually before those other stages like Hidden surface removal may come after Scan conversion although we are discussing it as before Scan

conversion. So, this sequence is not a strict requirement. The basic concepts are what matters the most.

So, far we have completed our discussion on the first two stages namely Object representation and Geometric or Modeling transformers. Currently, we are discussing the third stage that is Lighting or assigning color to the surface points. In the Lighting stage, we have introduced the basic issues that are addressed in this stage. And in the previous lecture, we have gone through a simple Lighting model. If you may recollect, in the simple lighting model, we assume that the color is essentially a composition of three constituent colors or intensities.

Intensity due to ambient light, intensity due to diffuse reflection, and intensity due to specular reflection. And we have learned models for each of these components and how to combine those models in the form of a summation of these three individual components.

(Refer Slide Time: 03:23)



Today, we are going to discuss Shading models which is related to assigning colors to the surface points, but in a slightly different way. Now, as we have seen during the simple lighting model discussion, the model itself is computation intensive.

(Refer Slide Time: 03:57)

Basic Idea

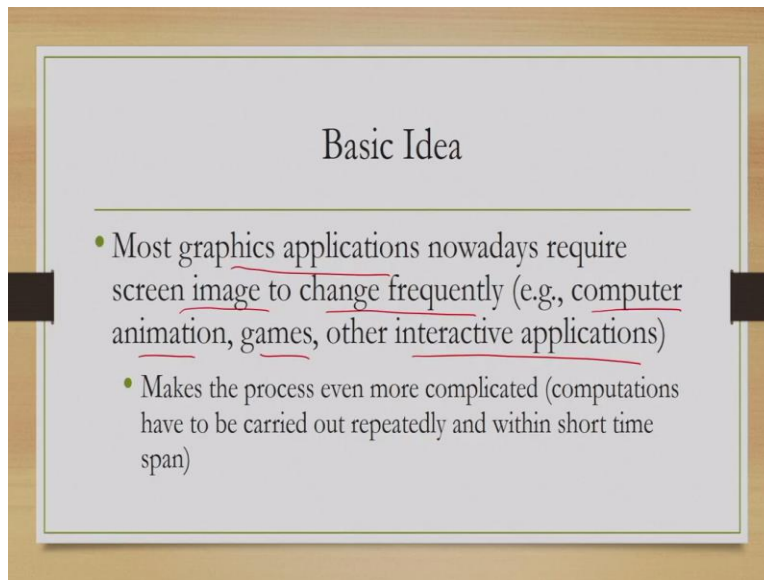
---

- With lighting model, we can compute color of a surface point in a 3D scene
- Computations involve lots of operations
- Consequently, generating image very expensive in terms of computing resources (processor, memory, etc.) and time

So, the calculation of color at a surface point in a 3D scene involves lots of operations. As a result generation of the image which includes assigning colors to the image is complex and expensive in terms of computing resources, what are those resources? Processor memory and so on. Also, it takes time. So, both are important resources and time. So, when we are talking of assigning colors or computing the colors, which is the job of the third stage.

What we are referring to is essentially the utilization of underlying computing resources. And in the Lighting model, we have seen that the utilization is likely to be very high because the computation involves lots of mathematical operations involving real numbers. Also, it is likely to take time.

(Refer Slide Time: 05:26)



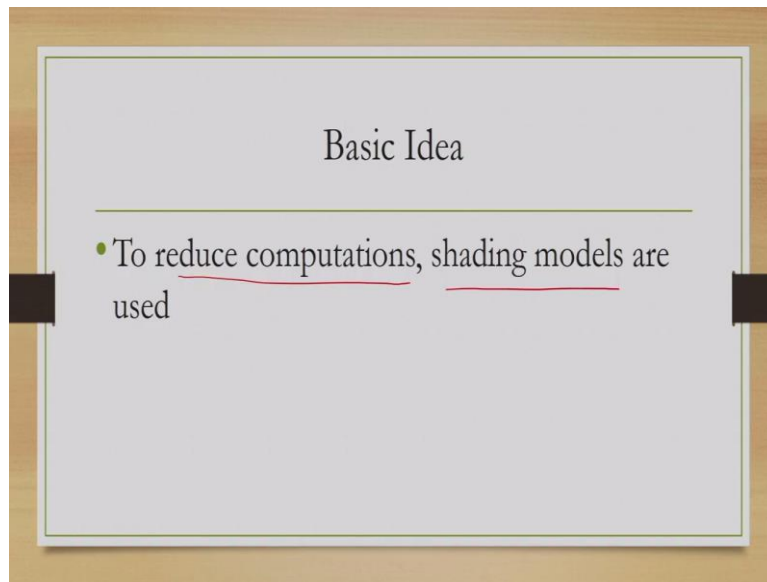
### Basic Idea

- Most graphics applications nowadays require screen image to change frequently (e.g., computer animation, games, other interactive applications)
  - Makes the process even more complicated (computations have to be carried out repeatedly and within short time span)

In practice, whenever we use some graphics applications, we may have noticed that the screen images change frequently. For example, if we are dealing with computer animation or computer games, or any other interactive application, so, screen content changes at a very fast rate. So, the requirement is that we should be able to generate newer and newer content and render it on the screen very quickly.

But if we are getting bogged down with this lots of complex computations for assigning colors or as we shall see in subsequent stages for doing other pipeline stages, pipeline operations, then that requirement may not be fulfilled, we will not be able to generate images quickly.

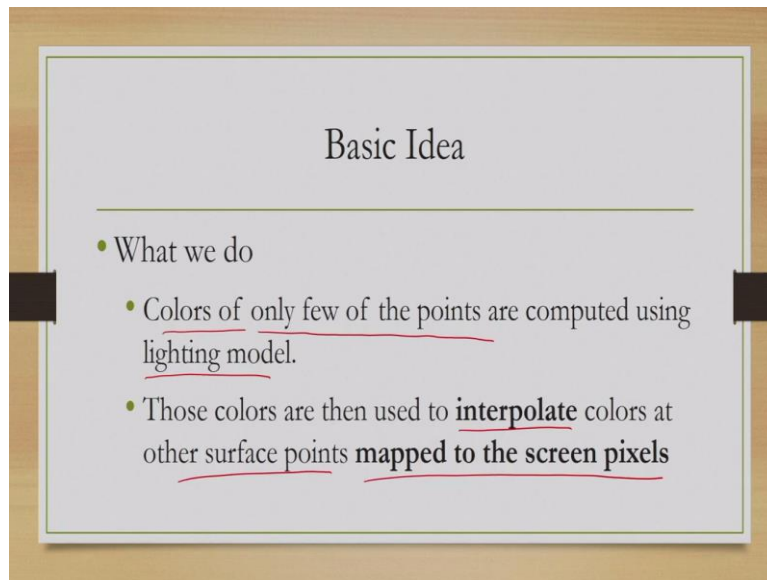
(Refer Slide Time: 06:32)



So, that may result in visible flickers, distortions which in turn may lead to irritation and annoyance to the user. And we certainly do not want such a situation to occur. In order to avoid such situations by reducing the number of computations involved or the amount of computations involved in assigning colors to surface points, we make use of Shading models.

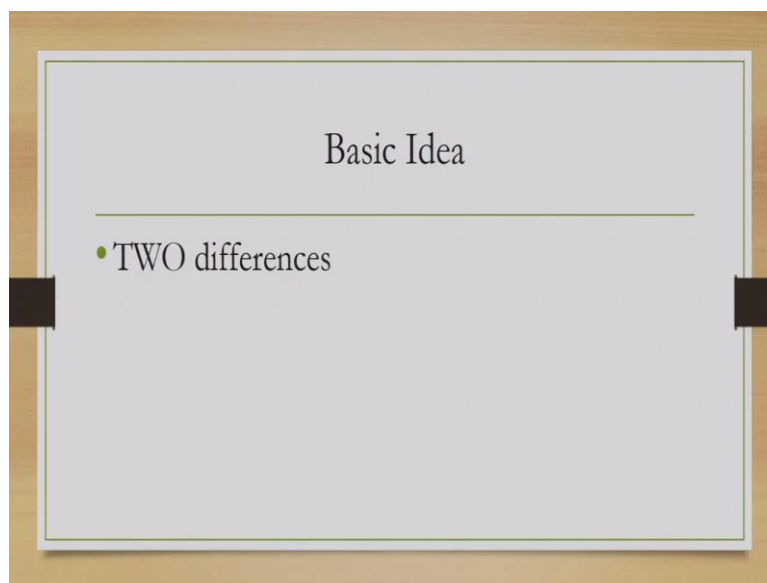
So, the idea of Shading models is that we have Lighting models, we can make use of it to find out or determine the color at a given point. However, if we do that for each and every point, then that is likely to be computation-intensive and time-consuming. To reduce computation we use some tricks in the form of Shading models.

(Refer Slide Time: 07:41)



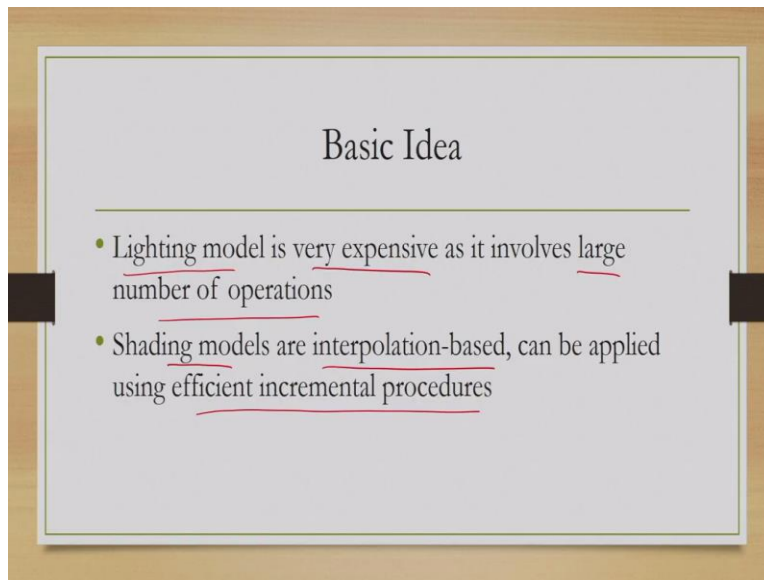
So, what do we do with a Shading model? First, we use the Lighting model to find out or compute colors of only a few of all the points that are there on the surface. Now, using those computed points, we perform interpolation and through interpolation, we assign color at other surface points which are mapped to the screen pixels. So, here Shading models are used when the surface points are already mapped to screen pixel. So, already rendering took place.

(Refer Slide Time: 08:38)



Now, between the Lighting model and Shading model, there are broadly two differences.

(Refer Slide Time: 08:50)



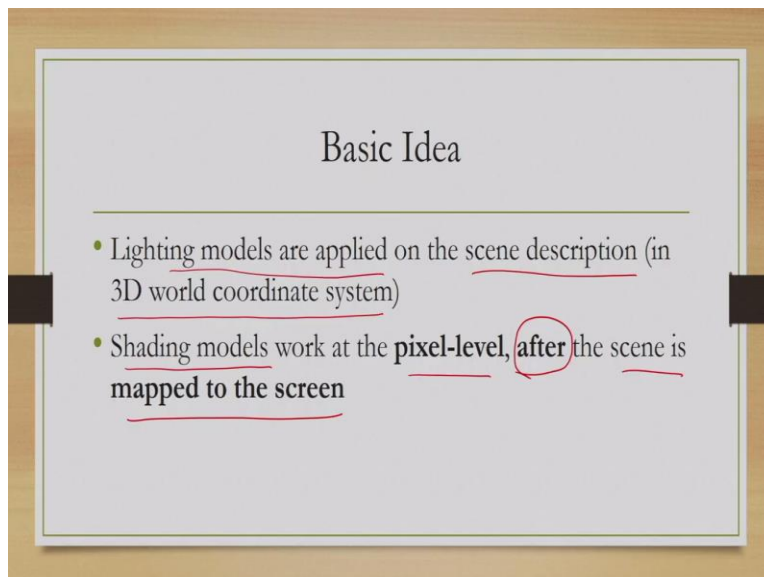
The slide is titled "Basic Idea" and contains two bullet points. The first bullet point states that lighting models are very expensive due to a large number of operations. The second bullet point states that shading models are interpolation-based and can be applied using efficient incremental procedures. The text in the slide is underlined.

### Basic Idea

- Lighting model is very expensive as it involves large number of operations
- Shading models are interpolation-based, can be applied using efficient incremental procedures

We have already mentioned that the Lighting model is very expensive because it involves large number of floating-point operations. In contrast, Shading models are interpolation-based. That means, we can come up with efficient incremental procedures to perform the computations rather than going for complex floating-point operations as we shall see in our subsequent discussions.

(Refer Slide Time: 09:28)



The slide is titled "Basic Idea" and contains two bullet points. The first bullet point states that lighting models are applied on the scene description in a 3D world coordinate system. The second bullet point states that shading models work at the pixel-level after the scene is mapped to the screen. The text in the slide is underlined, and the word "after" in the second bullet point is circled.

### Basic Idea

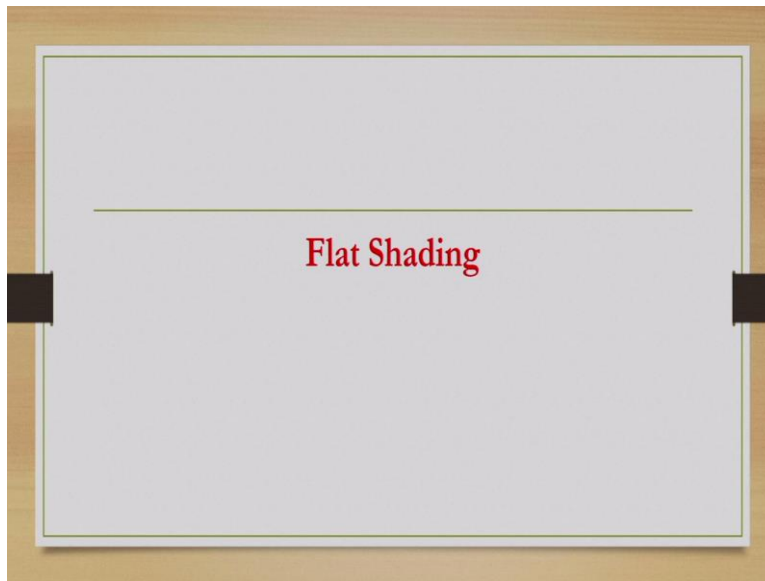
- Lighting models are applied on the scene description (in 3D world coordinate system)
- Shading models work at the pixel-level, after the scene is mapped to the screen

The other major differences, Lighting models are applied on the scene description that means, in a 3D world coordinate system whereas, as we have just mentioned, typically Shading models

work at the pixel level after the scene is mapped to the screen or after the rendering is done. That is the fifth stage of the pipeline is performed.

So, as I said at the beginning, it is not necessary that everything should work as per the sequence we have outlined. In practice things work with a slightly modified sequence, what is important is to know about the basic concepts rather than sticking to the exact sequence of pipeline stages.

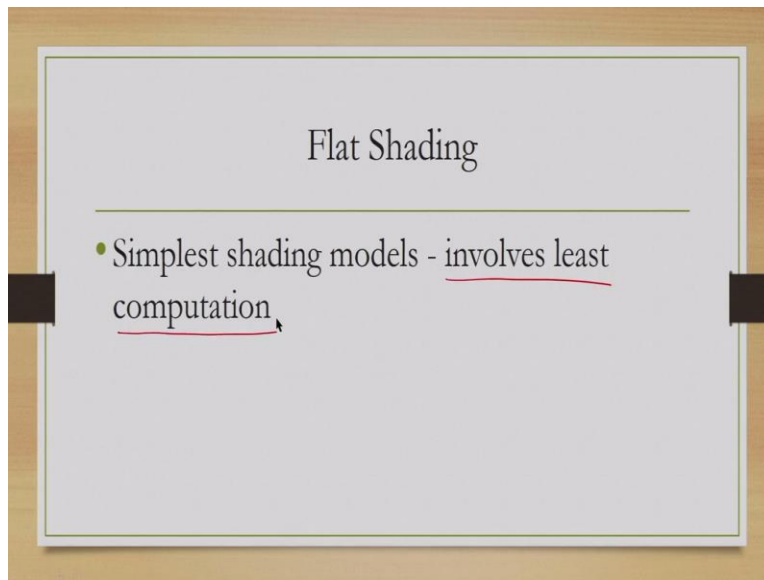
(Refer Slide Time: 10:23)



So, that is the idea of the Shading model and there are two major differences between Lighting and Shading models. Now, let us try to have a look and try to understand some Shading models briefly we will start with the simplest of the Shading models that is Flat Shading.

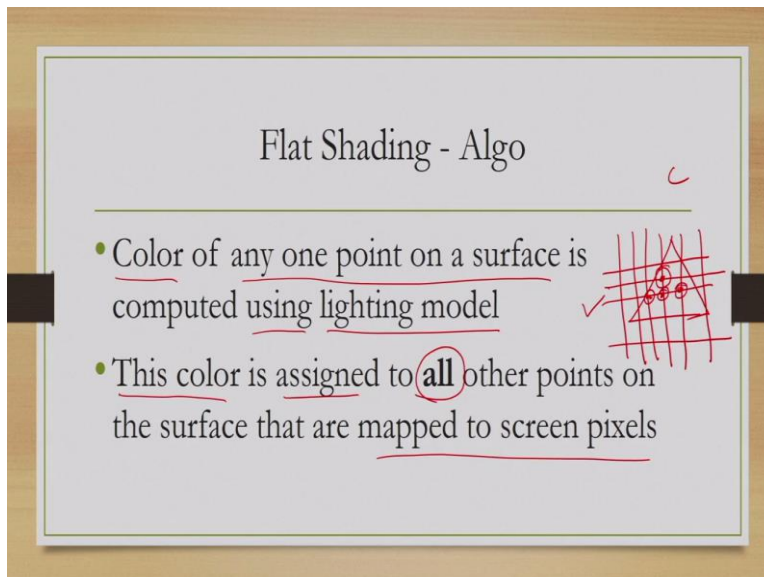


(Refer Slide Time: 10:51)



So, it involves the least amount of computation and what it does?

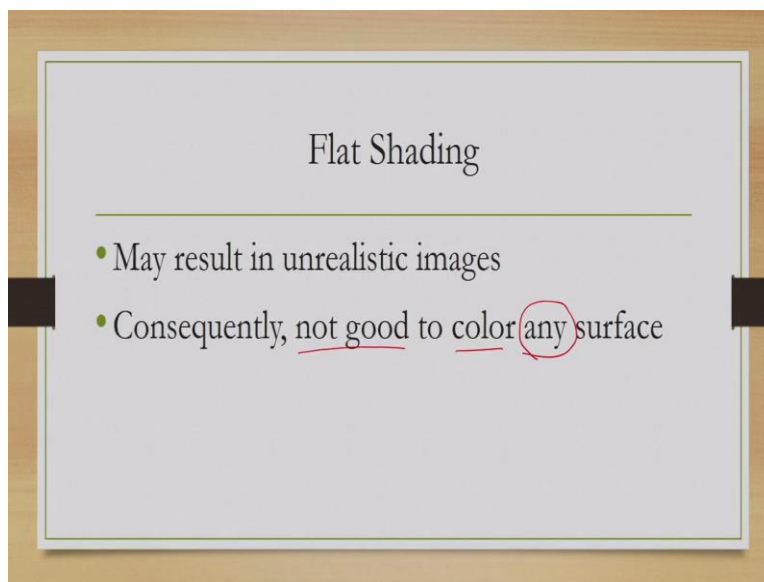
(Refer Slide Time: 11:00)



So, first in this Flat shading model, what we do first is, find out the color of any one point on a surface using the Lighting model. So, we apply the Lighting model and compute the color of any one point, a single point on a surface and then this color is assigned to all other surface points that are mapped to the screen pixels. So, suppose this is a surface and this is mapped. This is the pixel grid that I am drawing here.

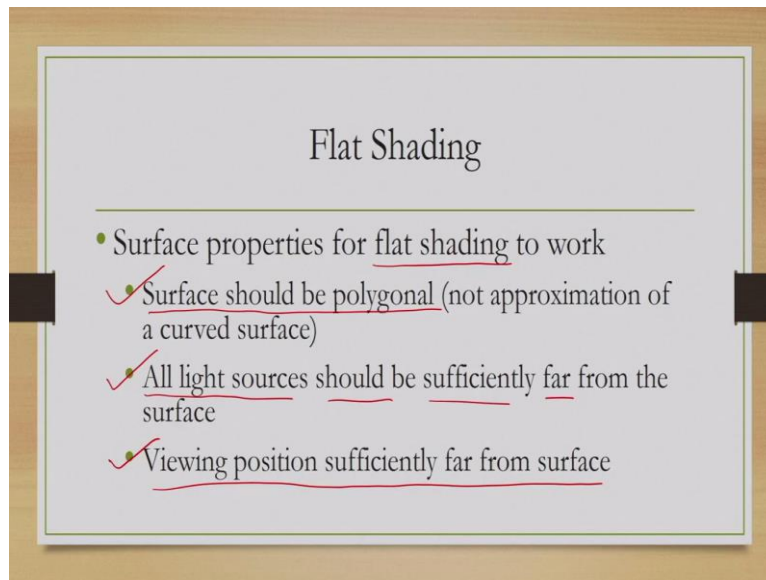
So, consider this scan line here. So, the pixels that are part of the surface are these three. Now, what we do in this Flat Shading model is that we choose any arbitrary point to apply the Lighting model and compute its color, color of that particular point in the 3D world coordinate system because we required to compute the vectors also, and then we use that to assign colors to all other pixels that are part of the surface. So, suppose we have computed color at this point say the color is C at this point, then we use this color to set color values of all other surface pixel points. For example, these three we set as C.

(Refer Slide Time: 13:03)



Clearly, this is a very simple scheme and it is likely to lead to unrealistic images unless we choose the application scenario properly. So, we must say that Flat Shading works in certain situations, but not in general good to color any surface. So, in general, we will not be able to use this particular Shading technique, because it may result in unrealistic images. So, when Flat Shading will be useful, there are a few conditions. What are those conditions? Let us see.

(Refer Slide Time: 13:46)

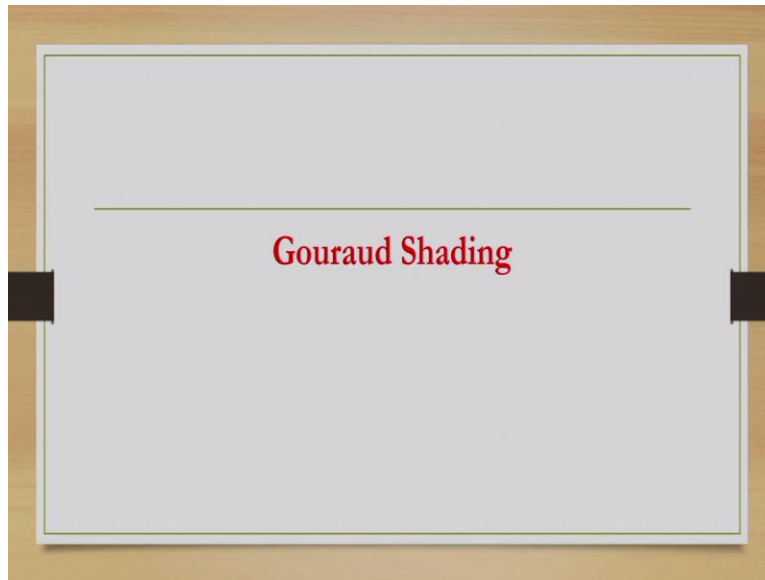


So, in order to make the particular Shading method work, we have to assume three things. First, the surface should be polygonal. Second, all light sources should be sufficiently far from the surface. So, the Shading effects sets of different intensities or colors are not applicable. And the third Viewing position is also sufficiently far from the surface. It may be obvious that if we are assuming that the light source is very far away and the viewer is also looking at the scene from a very far distance.

Then the minute differences between colors at neighboring regions may not be perceivable to the viewer, and accordingly whatever color we assign will look like uniform. So, in that case, Flat Shading may work and these three conditions restrict the use of the Flat Shading algorithm.

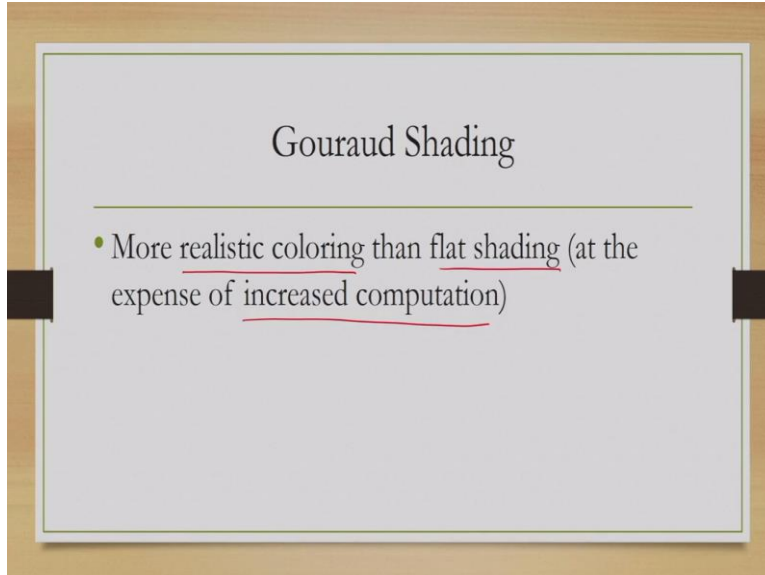
I repeat again in order to make the Flat Shading work there should be three conditions satisfied, first the surface must be polygonal in nature. All light sources should be sufficiently far from the surface and the viewing position should be sufficiently far from the surface. If these three conditions are not met, then the resulting colored surface may look unrealistic.

(Refer Slide Time: 15:56)



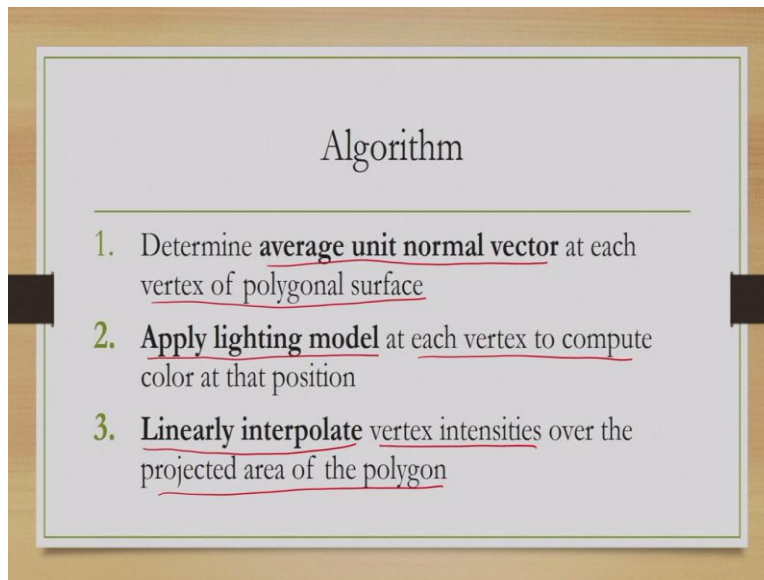
To avoid the problems that are associated with Flat Shading, an improved Shading model is there that is called Gouraud Shading. Let us try to understand Gouraud Shading.

(Refer Slide Time: 16:18)



It gives us a more realistic coloring effect than Flat Shading. But, at the same time, it is having more computation. So, the improvement is at the expense of increased computation.

(Refer Slide Time: 16:37)



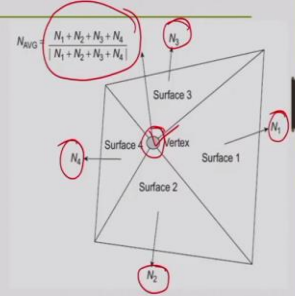
What happens in this Shading method, first, we determine the average unit normal vector at each vertex of a polygonal surface. We will soon see what we mean by the average unit normal vector. Then using that vector we compute color by applying a Lighting model at each vertex of the surface. Then we Linearly interpolate the vertex intensities over the projected area of the polygon.

So, three stages are there or three steps are there in the first step, we compute average unit normal vector, in the second step, we compute color at the vertex positions by considering the average unit normal vector and in the third stage, we Linearly interpolate the color that we have computed at the vertices of the surface. To assign color to other pixels that are part of the surface.

(Refer Slide Time: 17:56)

### Explanation

- **First step** implies vertices may be shared by more than one surface
  - For each vertex, calculate average unit normal vector - average of the unit normals of the surfaces sharing the vertex



$$N_{avg} = \frac{N_1 + N_2 + N_3 + N_4}{|N_1 + N_2 + N_3 + N_4|}$$

Now, let us try to understand the stages in detail. So, in the First step what we do, we compute the average unit normal vector. It essentially implies that a vertex of a surface may be shared by more than one surfaces. For example, consider this vertex here. Now, this vertex is shared by all the four surfaces in this figure. So, in that case, when we are trying to compute color at this vertex, which surface normal I should use?

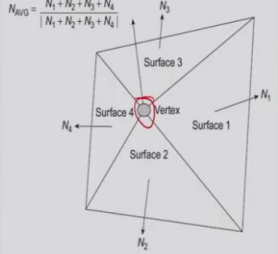
So, there is confusion. In order to avoid that Gouraud Shading tells us to compute the average unit normal vector. This is essentially the average of the unit normals of the surfaces sharing the vertex. So, in this particular example, the vertex here is shared by four surfaces, each will have its own normal vector. Say for Surface 1 it is  $N_1$ , Surface 2, Surface 3  $N_3$ , Surface 2  $N_2$ , Surface 4  $N_4$ .

We take the unit normal vectors then compute the average using the simple formula. So, this is a vector addition divided by a scalar quantity which is the modulus of the four-unit vectors. So, at that particular shared vertex, we use or we compute the average unit normal.

(Refer Slide Time: 19:43)

### Explanation

- **Second step** – average normal used in simple lighting model (instead of *normal*) to compute color
  - Performed for all vertices

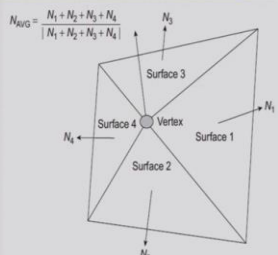

$$N_{avg} = \frac{N_1 + N_2 + N_3 + N_4}{|N_1 + N_2 + N_3 + N_4|}$$

Then in the second step with the average normal, we compute the color at this vertex using the Simple Lighting model. So, if you may recollect from our discussion on the Simple Lighting model to compute color components for diffuse reflection and specular reflection we had to use surface normals. So, instead of that regular surface normal, we use average surface normal to compute color. And this will do for all the vertices of the surface. So, it takes one surface at a time and compute colors for all vertices that define that particular surface.

(Refer Slide Time: 20:39)

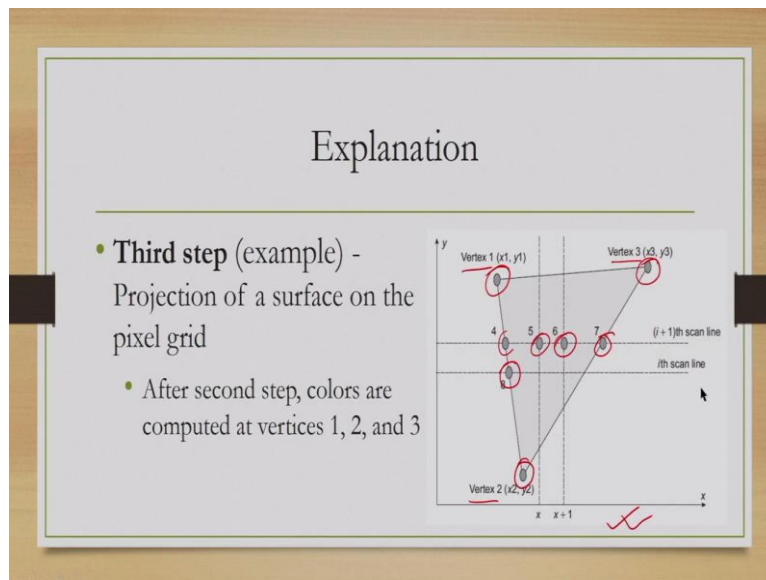
### Explanation

- **Third step** – use vertex colors to interpolate colors of the pixels that are part of the projected surface


$$N_{avg} = \frac{N_1 + N_2 + N_3 + N_4}{|N_1 + N_2 + N_3 + N_4|}$$

In the third step, which is the final step, we use these vertex colors to linearly interpolate the colors of the pixels that are part of the projected surface. So, we are assuming here that the surface is already projected on screen through the final stage of rendering and we already know the pixels that are part of the surface. Since we have computed the vertex colors in the first two stages, we use these colors to linearly interpolate and assign colors to other pixels that are part of the surface.

(Refer Slide Time: 21:24)



Let us try to understand in terms of one example. So, in this figure, we have shown a projected surface defined by three Vertices, Vertex 1, Vertex 3, Vertex 2. So, if we apply Gouraud Shading after the second step, we have already computed the colors of these three vertices by using the Simple Lighting model as well as the average unit normal vector at these vertex locations.

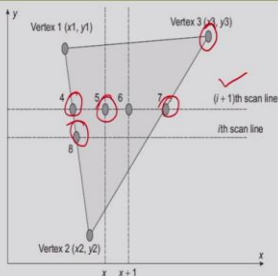
Now, we are interested to assign or find out the colors of the pixels that are part of the surface, but not vertices. For example, there are Pixels 4, 5, 6, 7 these are all part of the surface, also 8 and many more. 4, 5, 6, 7 belong to the same Scan line, 4 and 8 belong to two consecutive Scan lines.



(Refer Slide Time: 22:47)

## Explanation

- We use these colors to interpolate
- Colors at 4, 7 (two edge intersection points on a scan line)
- 5 (a pixel inside the projected surface on the same scan line)



$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

$$I_7 = \frac{y_7 - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_7}{y_3 - y_2} I_2$$

$$I_5 = \frac{x_7 - x_4}{x_7 - x_4} I_4 + \frac{x_5 - x_4}{x_7 - x_4} I_7$$

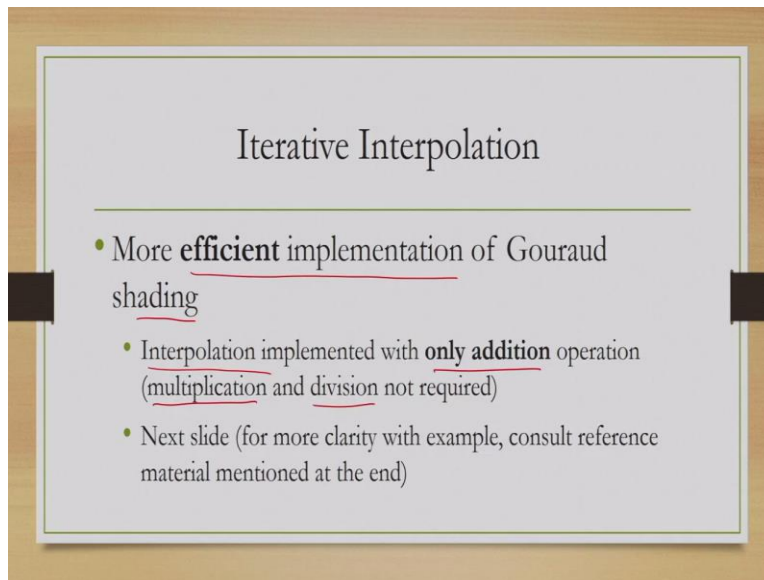
So, what we do, we perform linear interpolation in terms of the colors that are already computed for the vertices. So, we take one scan line at a time. For example, we have taken the  $(i+1)^{\text{th}}$  scan line. So, we compute the color at 4 and 7 which are two edge intersection points on the scan line which means, they are the intersection points between the edges of the surface and the scan line.

And we apply interpolation where  $I_1$  and  $I_2$  denote the intensity or the color value that is already computed at Vertex 1 and Vertex 2. So, for  $I_4$  we required these two values for  $I_7$  we require  $I_3$  and  $I_2$  where  $I_3$  is the vertex color at 3 here and this  $y_4$ ,  $y_2$  these are all y coordinates of those pixels.

So, we first compute colors for  $I_4$  and  $I_7$  on the same scan line and then using  $I_4$  and  $I_7$  we compute  $I_5$  which is here, which is inside the projected surface on the same scan line. So, the interpolation is shown here  $I_5$  is computed in terms of  $I_4$  and  $I_7$  note that here we are using the x coordinates of the pixels. In order to compute  $I_4$  and  $I_7$ , we used y coordinates.

But in order to compute  $I_5$  we are using x coordinates of the corresponding pixels. That is about the same scan line what happens when we want to compute the color of subsequent scan lines say in terms of previous colors, we want to compute the color for  $8^{\text{th}}$  pixel, the point 8.

(Refer Slide Time: 25:27)



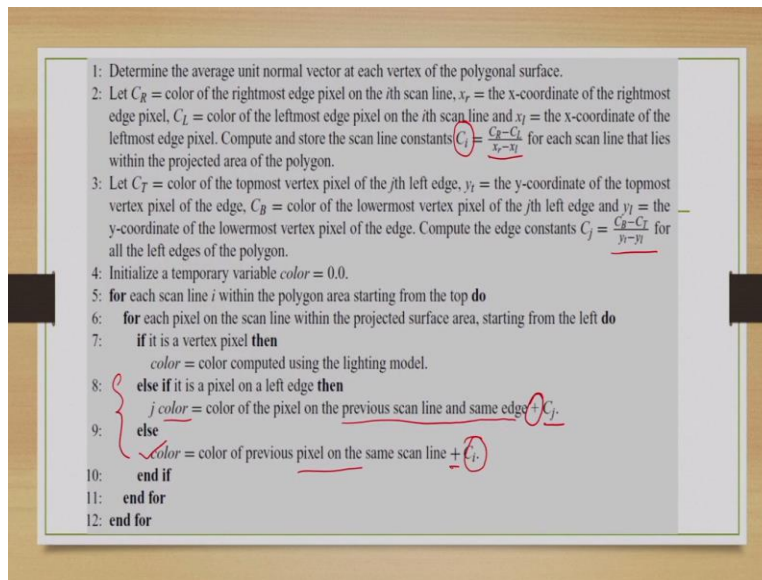
The slide is titled "Iterative Interpolation" and contains the following text:

- More **efficient** implementation of Gouraud shading
  - Interpolation implemented with **only addition** operation (multiplication and division not required)
  - Next slide (for more clarity with example, consult reference material mentioned at the end)

That is also possible. Actually, the equations or the formula that I have shown in the previous slide are not what is implemented in practice. There is a more efficient implementation of Gouraud Shading where we do not necessarily always compute the ratios and multiply it with the color values as we have seen in the previous slide. Instead, we perform interpolation with only addition, the multiplication and division are not required.

However, for more details on this incremental approach of interpolation, you may refer to the reference material mentioned at the end of this lecture. We will quickly have a look at the corresponding algorithm.

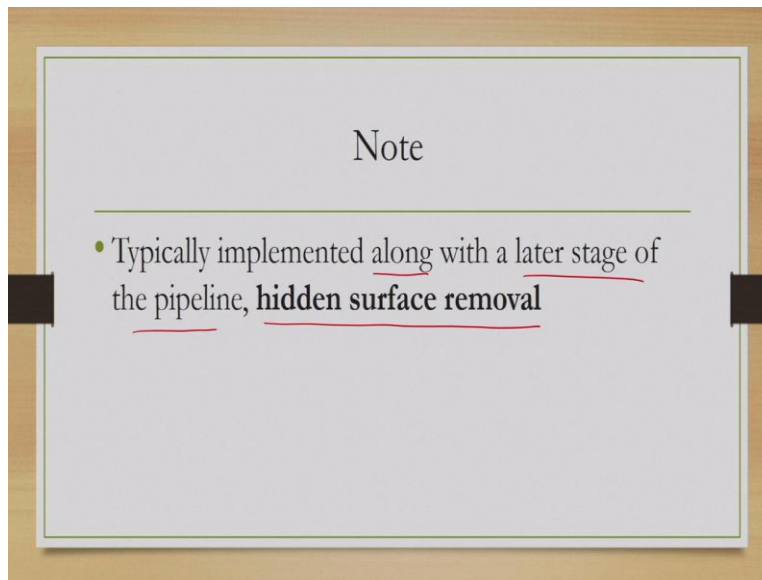
(Refer Slide Time: 26:37)



The incremental approach is encapsulated here. In these two lines, as you can see, color can be found out by simply considering the color already computed plus some age constants which are predetermined. Similarly, in this stage also in this stage, we can use simple addition to compute color where the addition is between previously computed color and some constant which is already pre-computed as shown in this line 2.

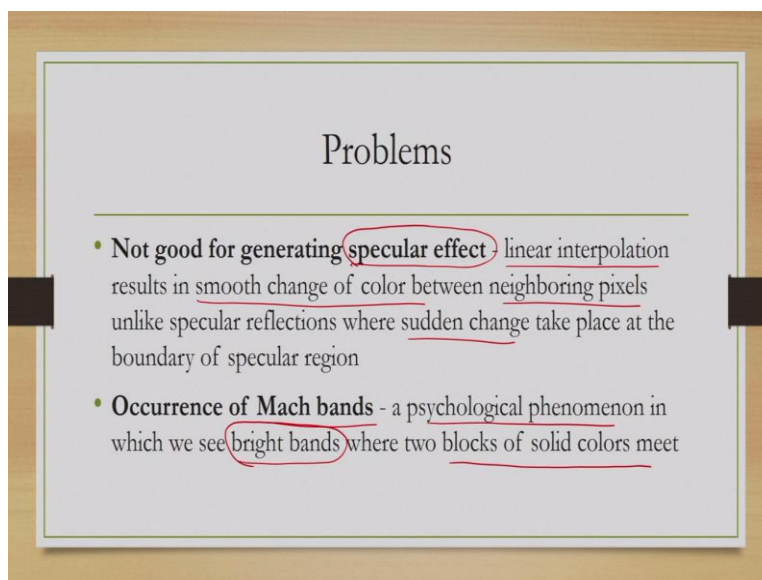
For more explanation on this algorithm, you may refer to the material that will be mentioned at the end. The basic idea is that this linear interpolation can be computed using simply addition rather than multiplication and division that is required if we are trying to do it in a classical way. So, this is a more efficient implementation of the stage three of Gouraud Shading.

(Refer Slide Time: 27:59)



And one more thing we should note here is that this particular Shading technique Gouraud Shading is implemented along with a later stage of the pipeline, which is part of the fourth stage it is called hidden surface removal. So, we will discuss about it later. So, Gouraud Shading assigns colors, but it is typically implemented along with a later stage of the pipeline that is a sub-stage of the fourth stage hidden surface removal.

(Refer Slide Time: 28:41)

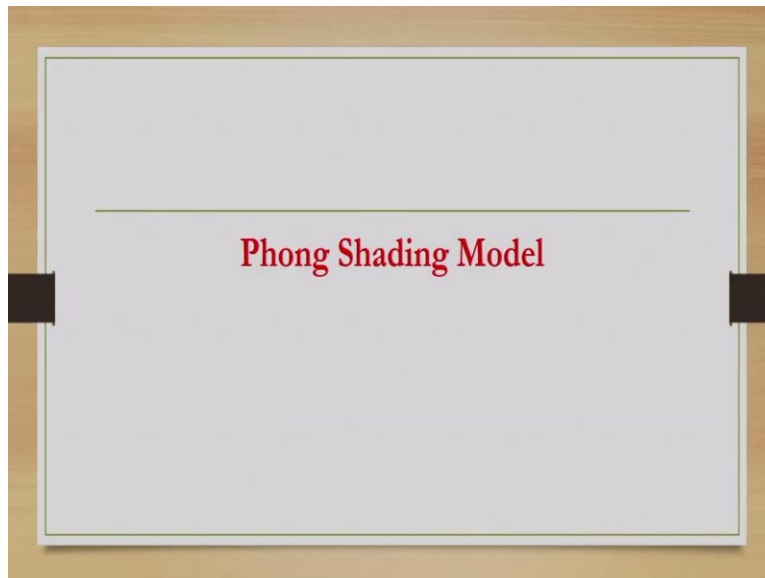


There are problems with Gouraud Shading as well, although it generates more realistic images compared to Flat Shading, but there are two major problems, one is it is still not good to generate

a specular effect that is that shiny surface or the bright spots that we get to see on the surface. This is primarily because this linear interpolation results in a smooth change of color between neighboring pixels which is not what happens in the specular reflection where there is a sudden change between neighboring pixels.

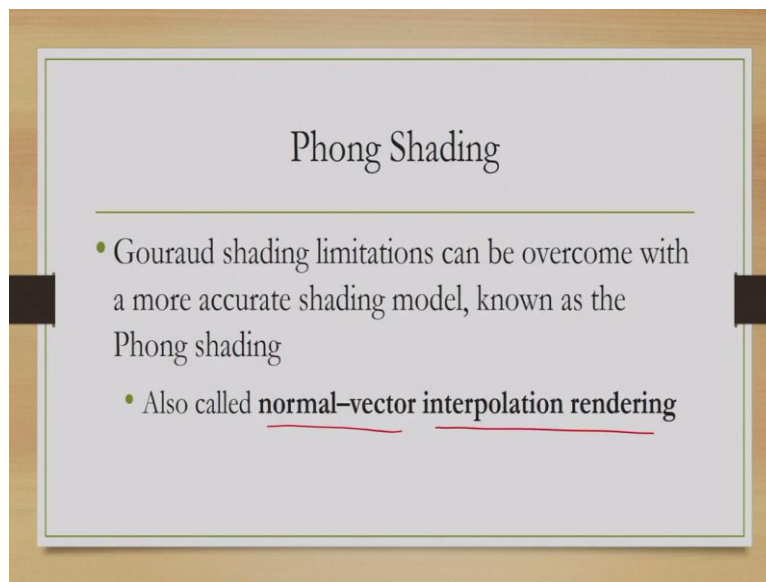
Secondly, what Gouraud Shading suffers from this problem of occurrence of Mach bands is kind of psychological phenomena in which we see bright bands when two blocks of solid colors meet, so, if two constitutive surfaces are assigned different colors, then at their joining point we may get to see some band like things, which is a psychological phenomenon known as Mach banding effect. And this may result if we apply Gouraud Shading.

(Refer Slide Time: 30:13)



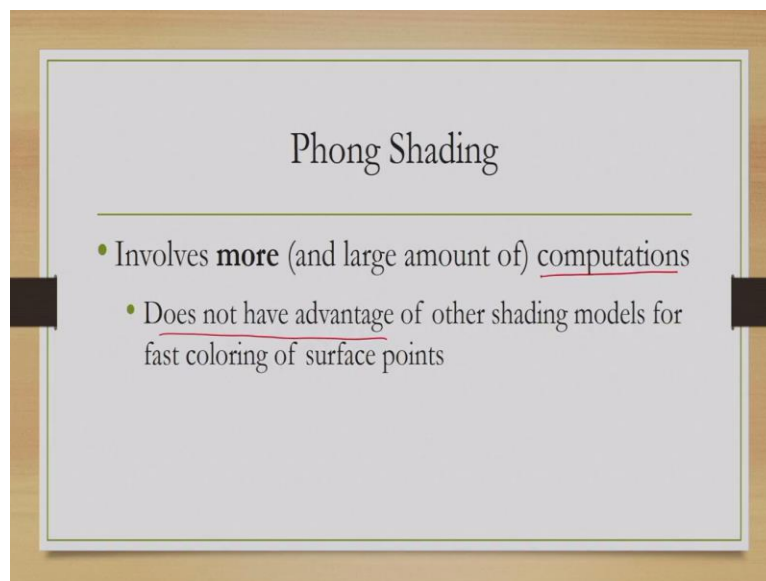
There is a third Shading method, which is quite advanced and it eliminates all problems that we have discussed so far with Flat Shading and Gouraud Shading.

(Refer Slide Time: 30:26)



But, it is heavily computation-intensive and requires huge resources as well as time. We will just learn the basic idea and we will not go into the details. So, this Phong Shading is also known as Normal vector interpolation rendering.

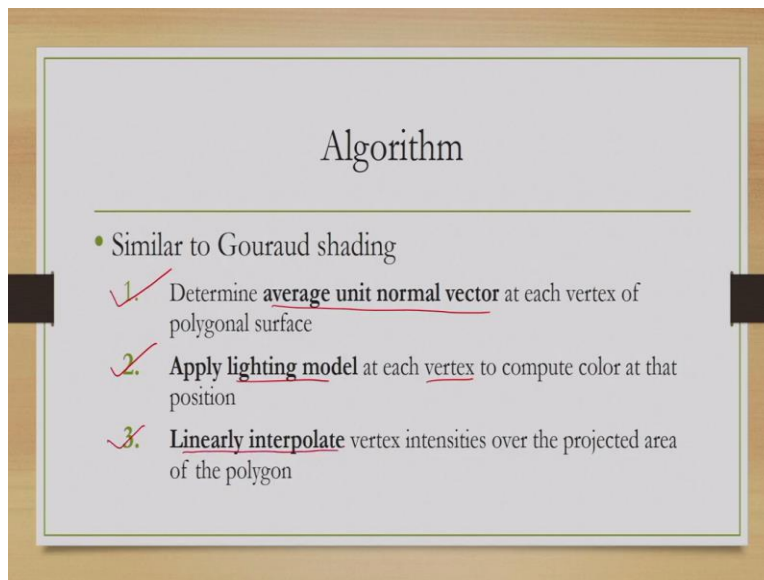
(Refer Slide Time: 30:51)



Now, in this, we actually compute color at each point where we find out the normal vectors in a different way. So, there is actually no interpolation involved, interpolation only in terms of finding out vectors, but not computing colors, it takes much more time as expected and it does not have the advantage of other Shading models in terms of reduction in computations.

So, it gives us a very realistic image because the coloring effect is closer to reality due to the very sophisticated approach, but for the same reason, it cannot compute colors with reduced computations, which are the advantages of Shading models. So, it is not having the main advantage, but it gives us more realistic images. We will not go into the details of it, it is quite complex. And if you are interested you may refer to the reference material that will be mentioned at the end of this lecture.

(Refer Slide Time: 32:12)



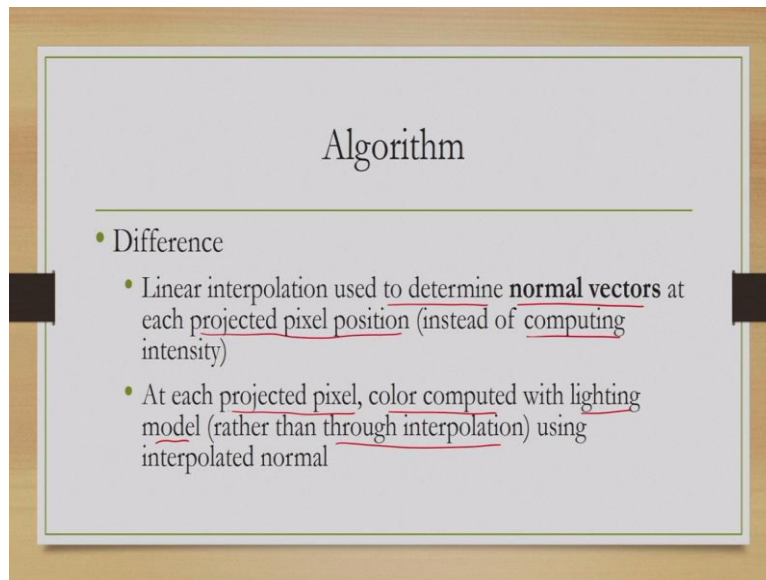
**Algorithm**

---

- Similar to Gouraud shading
  1. Determine average unit normal vector at each vertex of polygonal surface
  2. Apply lighting model at each vertex to compute color at that position
  3. Linearly interpolate vertex intensities over the projected area of the polygon

I will just mention the three steps. In the first stage, we compute the average unit normal vector-like in Gouraud Shading. In stage two we apply the Lighting model at each vertex to compute color and in stage three we apply interpolation but in a different way.

(Refer Slide Time: 32:43)



What is that difference? Instead of interpolating colors we now interpolate to determine normal vectors at each projected pixel position. Remember that normal vectors assume that we are in a 3D world coordinate system, whereas the projected pixel position assumes that we are already in the device coordinate system which is 2D. So, we need to calculate normal vectors to actually apply the lighting model which involves the use of normal vectors.

We do that here in Phong Shading. So, the interpolation is not used to compute intensity instead it is used to determine normal vectors. Once that is done at each projected pixel we know the normal vectors through interpolation, we compute color using the Lighting model. So, here we are computing color using the Lighting model, but not through interpolation only difference is that in order to compute color with the Lighting model, we need a normal vector that we are finding out through interpolation.

So, essentially in this case, if we summarize the surface is projected we identified a set of pixels that constitute the surface, at each pixel location we are applying the Lighting model. Before that, we are using interpolation to find out the normal vector at that pixel location and then we are using the Lightning model. So, we are using the Lightning model repeatedly, which increases the computation and time.



For more details, you may refer to the material that will be mentioned at the end. We will just outline and we will stop here on the discussion on Phong Shading. Now, let us try to understand the idea of Shading in terms of one illustrative example.

(Refer Slide Time: 35:01)

### Putting Everything Together - Example

\* Consider a cubical object with vertices  $A(0, 0, 2)$ ,  $B(2, 0, 2)$ ,  $C(2, 2, 2)$ ,  $D(0, 2, 2)$ ,  $E(0, 0, 0)$ ,  $F(2, 0, 0)$ ,  $G(2, 2, 0)$ , and  $H(0, 2, 0)$ . We want to create a scene of a room, in which the object is treated as a 'shelf attached to a wall', keeping the relative positions of the corresponding vertices same as in the original object. The wall is parallel to the  $XZ$  plane, cutting the positive  $Y$ -axis at a distance of 1 unit. The length of the sides of the shelf is half that of the original object. The surface of the shelf  $CD'H'G'$  corresponding to the surface  $CDHG$  of the object is on the wall, with its lower left corner at  $(1, 1, 1)$ .

The cube in 3D world coordinate after transformation

Let us consider a cubical object with the vertices given A, B, C, D, E, F, G, and H. Now, with this object we want to create a scene of a room in which the object is treated as a shelf attached to a wall keeping the relative positions of the corresponding vertices same. So, the relative position will be the same and there is some specification about the wall also it is parallel to the  $XZ$  plane cutting the positive  $Y$ -axis at a distance of 1 unit.

And the length is reduced by half and we also mentioned the corresponding vertices in the shelf with respect to the original vertices. So, after the specified transformation, this figure shows the 3D scene with the shelf attached to the wall as specified in the problem.

(Refer Slide Time: 36:39)

### Putting Everything Together - Example

- The shelf as it looks like after projecting onto the pixel grid, along with some of the mapped vertices and one vertex location

We also have to know its projection in order to be able to apply Shading. Now, that is mentioned here the shelf looks something like this as shown here with the vertices specified each of which corresponds to the corresponding vertex in the original scene. So, F', belongs to F', E double' belongs to E' and so, on. And in the projected scene, we have mentioned one vertex coordinate so that other coordinates can be derived.

For example, here we have mentioned the vertex coordinate of 4 7 then we can derive E to be, X will remain the same Y will be reduced by 1 2 3 4 5, so Y will be 2 and so on for other vertices. In that way, we can derive the locations.

(Refer Slide Time: 37:57)

### Putting Everything Together - Example

- Let the room have a monochromatic point light source at  $(1, -1, 3)$  with  $I_s = 2$  units
- Further assume ambient light intensity  $= 1$  unit and the object have  $k_a = 0.5$ ,  $k_d = 0.25$ ,  $k_s = 0.25$ , and  $n_s = 10$  specular exponent = 10
- You are looking at the shelf from a location  $(3, -1, 3)$

Now, assume that the room has a monochromatic point light source at a given location with intensity of 2 units and also assume there is an ambient light with the intensity of 1 unit and the reflective coefficients or reflectivities for the 3 components  $k_a$  for ambient light,  $k_d$  for diffuse reflection due to direct light and  $k_s$  for specular reflection due to direct light are specified. And the specular exponent is also specified as 10 and the viewer is located at this position.

(Refer Slide Time: 38:48)

### Putting Everything Together - Example

- Compute intensities at the pixels  $P_1$ ,  $P_2$ , and  $P_3$  assuming flat shading

Assuming this setting let us try to compute the colors at the pixels  $P_1$ ,  $P_2$ , and  $P_3$  assuming the simplest of all Flat Shading. So, this is  $P_1$ , this is  $P_2$  and this is  $P_3$ , how we can do that?

(Refer Slide Time: 39:13)

Putting Everything Together  
- Example

- Coordinates of some of the projected vertices and the pixels of interest ( $P_1$ ,  $P_2$ , and  $P_3$ ) are as follows.
  - $D'' = (7, 10)$ ,  $C'' = (7, 15)$
  - $P_1 = (8, 8)$ ,  $P_2 = (10, 8)$ ,  $P_3 = (10, 6)$
  - Also  $P_4 = (5, 8)$

So, we first determine the coordinates of the projected vertices which should be easy.

(Refer Slide Time: 39:35)

Putting Everything Together  
- Example

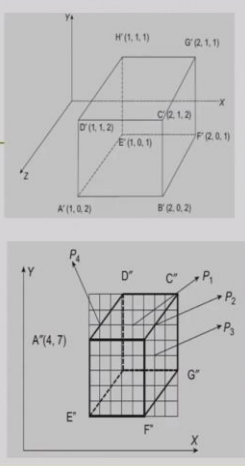
- light source is above  $A'B'C'D'$  surface and on the left side of the plane which contains the surface  $B'F'G'C'$ 
  - Will illuminate  $A'B'C'D'$  but will not contribute anything towards the illumination of  $B'F'G'C'$

Then, we have to compute the color at any given point on the surface. Note that as per the problem description light source is above the surface  $A'$ ,  $B'$ ,  $C'$ ,  $D'$ , and on the left side of the plane which contains the surface  $B'$ ,  $F'$ ,  $G'$ ,  $C'$ . Thus, it will illuminate this surface, but will not contribute anything towards the illumination of the other surface. So, this is the first observation of the problem description.

(Refer Slide Time: 40:17)

Putting Everything Together  
- Example

- We can calculate color at any point on the surface and use the same value throughout the surface (in flat shading)
- Let's calculate color at B'



Now, in order to compute color, we can calculate color at any point and then use the same value throughout the surface in Flat Shading. So, let us calculate color at this vertex B'.

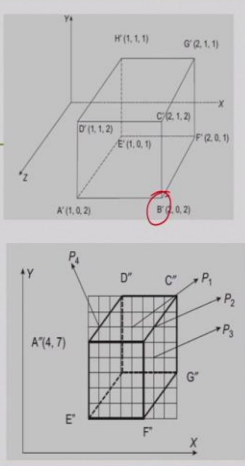
(Refer Slide Time: 40:37)

Putting Everything Together  
- Example

$\hat{N} = (0, 0, 1)$

$\hat{L} = \left(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$  ✓

$\hat{V} = \left(\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$  ✓



If we see the scene and the object description in the scene, then we know that surface normal at B' and the unit surface normal will be this. Now, we know the light source, so the unit vector towards the light source can be computed in this way and unit vector towards the viewer because we know viewer location can be computed in this way.

(Refer Slide Time: 41:17)

### Putting Everything Together - Example

$$\hat{N} \cdot \hat{L} = \frac{1}{\sqrt{3}} \approx 0.58$$

$$\hat{V} \cdot \hat{R} = \hat{V} \cdot [2\hat{N} \cdot \hat{L} \hat{N} - \hat{L}] = \frac{2}{3\sqrt{3}}(\sqrt{3} - 1) \approx 0.28$$

Therefore the color at B'

$$= (0 \times 5) \times 1 + 0 \times 25 \times 2 \times (0.58) + 0 \times 25 \times 2 \times (0 \times 28) \times 10$$

$$= 0.79 \text{ unit}$$

Then with these values we can get the dot product as something like this and also this second dot product for the specular component as something like this and with these values and using the reflectivity coefficients we can get the three components added up to get the overall color value to be 0.79 unit at B'.

(Refer Slide Time: 42:06)

### Putting Everything Together - Example

- P1 and P2 both are part of the same surface containing B'
- Color at P1 = color at P2 = 0.79 unit

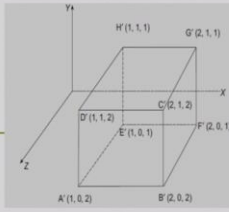
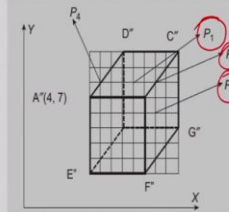
Now, we know that P<sub>1</sub> and P<sub>2</sub> both are part of the same surface containing B'. P<sub>1</sub> is part of the B' and P<sub>2</sub> is part of the surface containing the B'. Now, if we are using Flat Shadings, so, we have

already computed the color at B' so, we will simply assign these colors to all the surface points that mean to P<sub>1</sub> and P<sub>2</sub>. So, the values color values of P<sub>1</sub> and P<sub>2</sub> will be 0.79 units.

(Refer Slide Time: 42:47)

### Putting Everything Together - Example

- However, light source do not contribute in the illumination of **B'F'G'C'**
  - Thus, color decided by ambient light only
  - Since P<sub>3</sub> part of B'F'G'C', color at P<sub>3</sub> = **0.51 = 0.5 unit**

And we have also noted that the light source does not contribute to the illumination of this other surface B', F', G', C'. So, in that case, there will be no contribution due to the direct light source. So, those two components due to diffuse reflection and specular reflection due to direct light source will be 0 and it will be illuminated only by the ambient light which is computed using this expression  $k_a I_a$ , where  $k_a$  is the coefficient value and  $I_a$  is the intensity and we get this value.

So, these are the values that we have computed using Flat Shading P<sub>1</sub>, P<sub>2</sub> and P<sub>3</sub>. Note here that we did not use color model or the Lighting model to compute values at P<sub>1</sub> and P<sub>2</sub> instead we computed the value only at B, B' and use that to assign color to P<sub>1</sub> and P<sub>2</sub>. And similarly, we have done for P<sub>3</sub>.

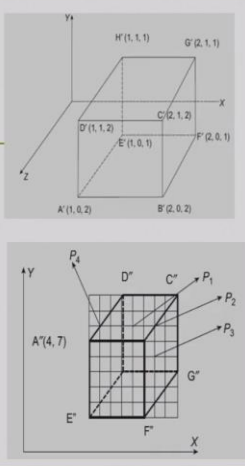
So, here we have reduced the usage of the Simple Lighting model and by that, we have reduced the amount of computations required. However, as I said before since we are using Flat Shading the colors that are computed may not look realistic when they are rendered on the screen if, the distances of the source, as well as the viewer from the surface, are not sufficiently large.



(Refer Slide Time: 44:34)

Putting Everything Together  
- Example

- Same results can be arrived at by considering the vectors only rather than such informal reasoning
  - Left as an exercise

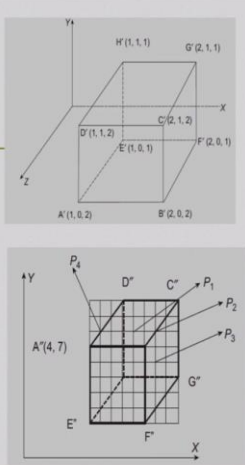


Now, here also it may be noted that we have done some informal reasoning to come to the conclusion of the color values. But if we simply apply the algorithms, then also we will get the same result. We do not need to actually do any informal reasoning but that you can try on your own. We will not work that out here.

(Refer Slide Time: 45:00)

Putting Everything Together  
- Example

- Now assume Gouraud shading to compute color
  - Left as an exercise



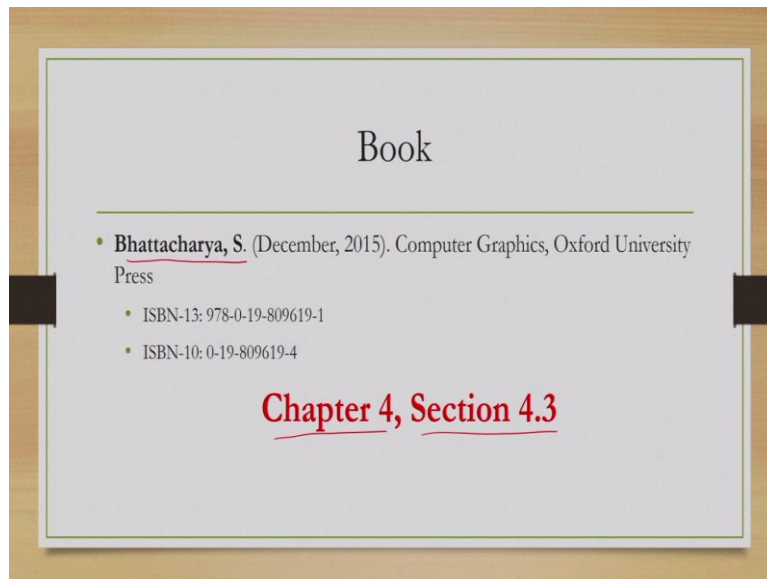
And also I would like to request you to use the Gouraud Shading algorithm to perform the same computations for the three points. I leave it as an exercise for all of you to do. And then you can compare the amount of computation as well as the end of values that you are getting and from



there, you can get some informal idea of the effect that results in the application of these different Shading models.

So, we have come to the end of our lecture today. To quickly recap, we learned about the idea of Shading and its difference with the Lighting model. Then we discussed in detail, Flat Shading model and Gouraud Shading models, and just outline the idea of Phong Shading models. With the illustrative example, I hope, you could get some idea of the application of the Shading models and its advantages over-application of only the Lighting model to compute colors. With that, I would like to end today's lecture.

(Refer Slide Time: 46:38)



For more details, including the ones that are mentioned at different points of the lecture you may like to refer to this book. Please have a look at Chapter 4, Section 4.3 for the details on all the topics that I have covered today. Thank you and goodbye.