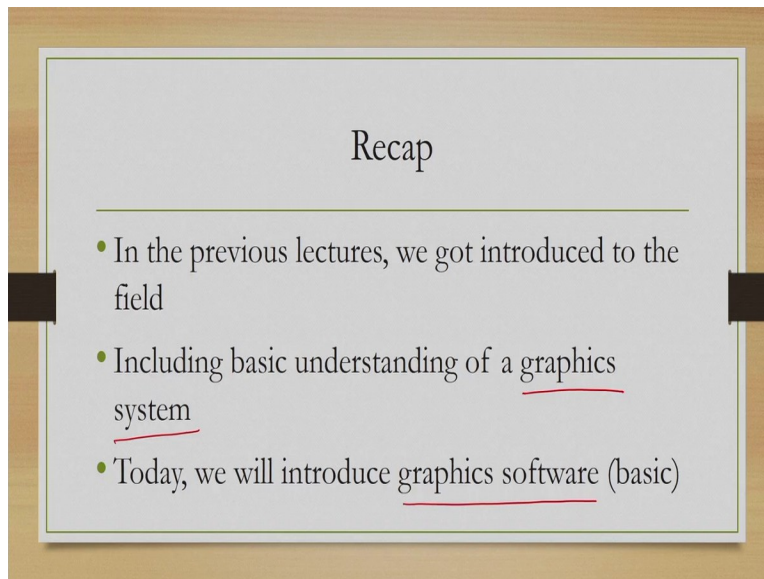


Computer Graphics
Dr. Samit Bhattacharya
Computer Science and Engineering
Indian Institute of Technology, Guwahati
Lecture 4
Introduction to 3D Graphics Pipeline

Hello and welcome to lecture number 4 in the course Computer Graphics.

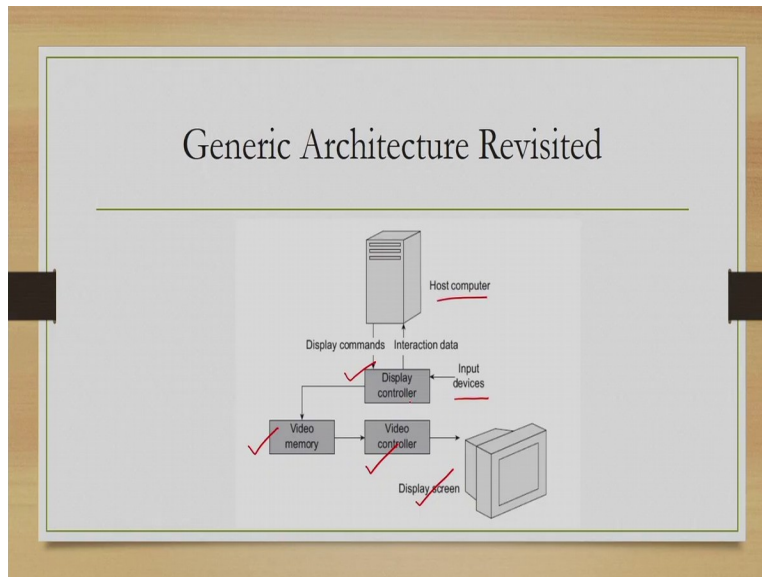
(Refer Slide Time: 00:39)



Before we start, we will briefly recap what we have discussed in the previous lectures. So we started with a basic introduction to the field where we discussed about the historical evolution as well as the issues and challenges that are encountered by the workers in this field. This was followed by a basic introduction to the graphics system. So whenever we talk about computer graphics implicitly we refer to some hardware platform on which some software works.

And the basic hardware structure or architecture of a graphic system has been introduced in one of the previous lectures. Today we are going to introduce the other component of the graphic system, namely the graphics software. Of course at this stage we will restrict ourselves to a basic introduction and the software stages will be discussed in details in the subsequent lecture.

(Refer Slide Time: 02:06)



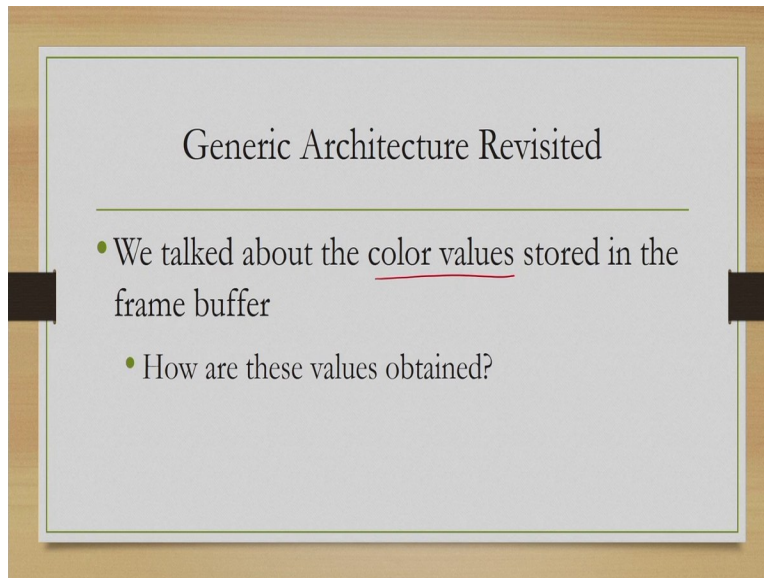
So let us recap what we have learned about a generic architecture of a graphic system. As we mentioned in one of our earlier lectures, so there are 3 unique components of a graphic system. One is the display controller, one is the video memory and the 3rd one is a video controller. What the display controller does? It essentially takes input from the host computer as well as from some external input devices which are used to perform interactive graphics. And based on that input, it creates a representation, a digital representation of a 2D image. That is the job of the display controller.

Now that representation, that the controller generates is stored in a memory which is called video memory. Now the content of the memory, video memory is given as input to the 3rd component that is the video controller which takes the memory content as input and then generates certain voltage levels to drive some electro-mechanical arrangements that are required to ultimately display the image on a computer screen.

As you may recollect, we also mentioned that most of the things are done separately without involving the CPU of the host computer. So typically computers come with a component which is called as graphics card which probably all of you have heard of which contains the video memory, the video controller and the display controller components. And the processing unit that is typically part of the display controller is known as the GPU or graphics processing unit, this is

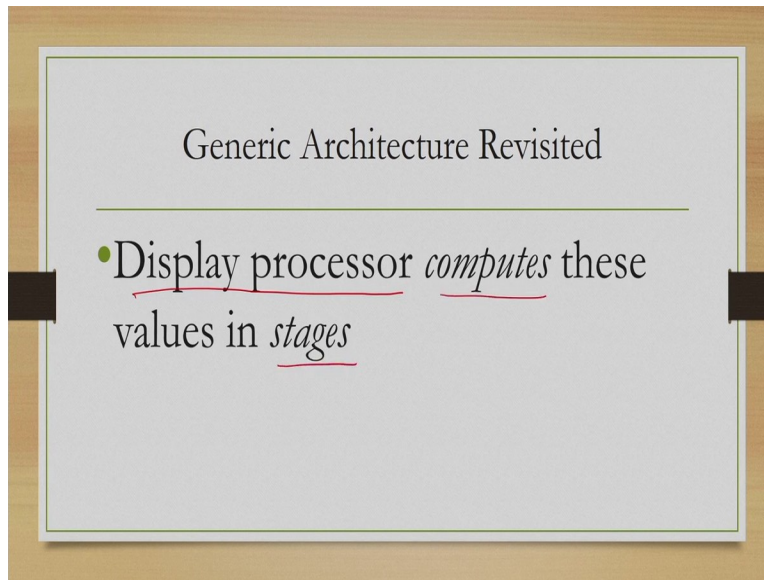
separate from the CPU or the main processing unit of a host computer. And the GPU is designed to perform graphical activities, graphical operations.

(Refer Slide Time: 04:48)



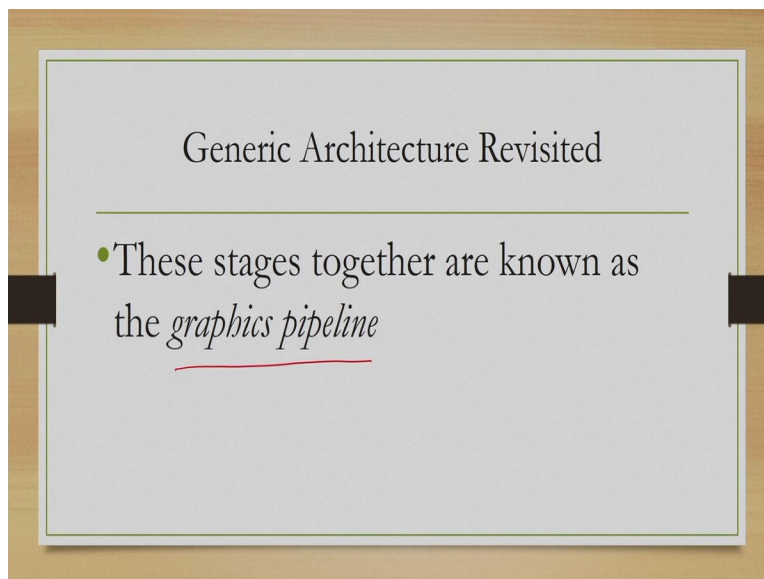
Now in this generic architecture as we said, display controller generates representation of an image. So what that representation contains? It contains some color values or intensity values in a specific format which ultimately is used to generate the particular sensation of color on the screen. Now from where these color values are obtained? Let us try to go into a some details of the process involved in generating these color values.

(Refer Slide Time: 05:29)



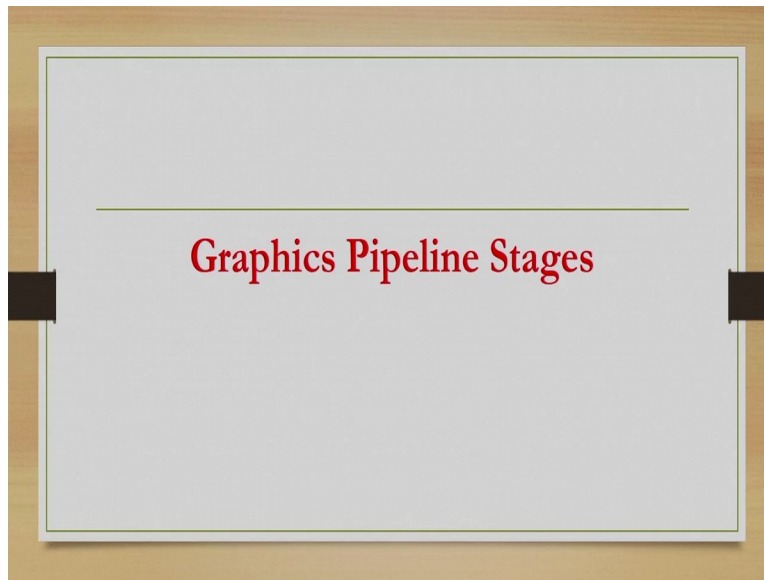
Now these color values are obtained by the display processor through some computations that are done in stages. So there are a series of computations and these computations ultimately result in the generation of the color values.

(Refer Slide Time: 06:00)



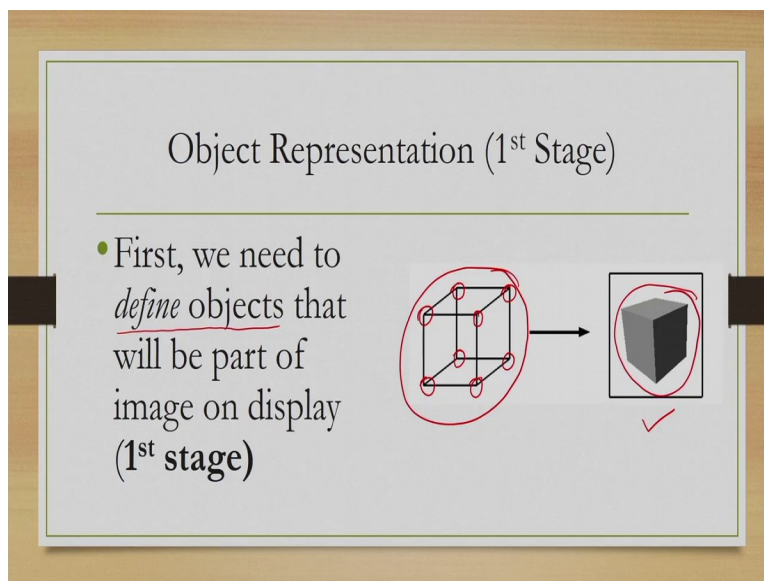
Now these stages or the series of steps that are involved in the generation of color values are together called the graphics pipeline. This is a very important terminology and in our subsequent lectures we will discuss in details the stages of the pipeline, that actually will be the crux of this course.

(Refer Slide Time: 06:30)



But today we are going to introduce the pipeline for our benefit, so that we can understand the later discussion better. So let us get some introductory idea on the pipeline and its stages.

(Refer Slide Time: 06:47)



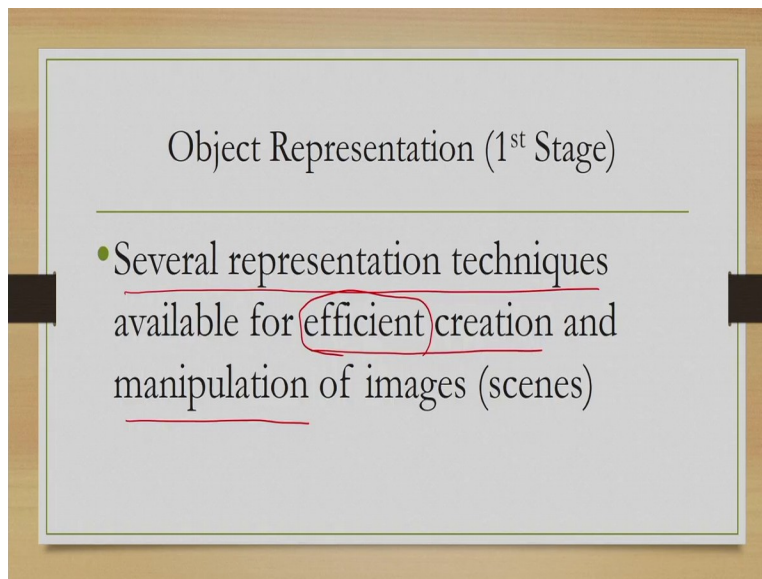
There are several stages as I mentioned, first stage is essentially defining the objects. So when we talk of creating a scene or an image, it contains objects. Now there needs to be some way to represent these objects in the computer. That activity where we define objects which are going to be the parts of the images constitute the first stage of the pipeline which is called object

representation stage. For example, as you can see in this figure on the screen we want to generate the image of a cube with color values as shown on the right hand part of the screen.

Now this image contains an object which is a cube and on the left hand side here we have defined this cube. So when we talk of defining what we mean essentially as we can understand intuitively, defining the cube involves specifying the vertices or edges with respect to some reference frame that is the definition in this simple case that is what are the vertices or what are the edges as pair of vertices.

Ofcourse cube is a very simple object, for more complex objects we may require more complex definitions, more complex way of representing the objects.

(Refer Slide Time: 08:53)

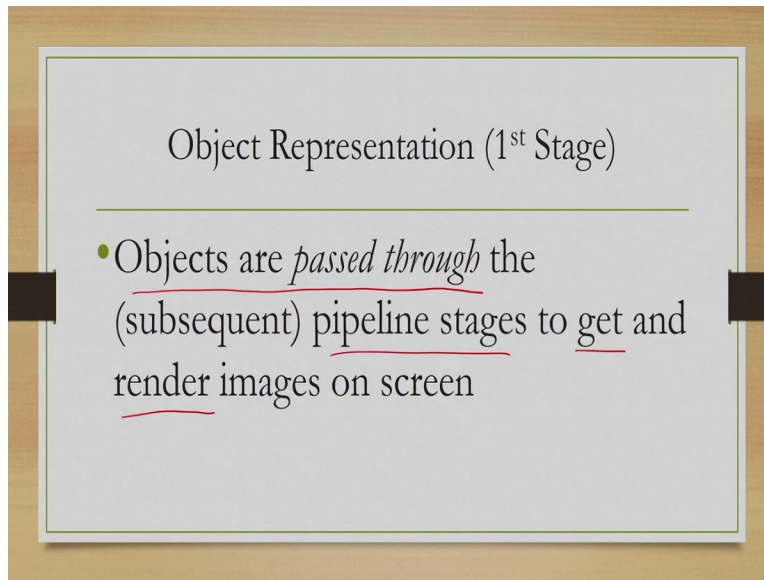


Accordingly, several representation techniques are available for efficient creation and efficient manipulation of the images. Note here on the term efficient, so when we talk of this term efficient, essentially what we refer to, we refer to the fact that the displays are different, the underlying hardware platforms are different. So whatever computational resources we have to display something on a desktop or a laptop are likely to be different with respect to whatever we have to display something on a small mobile device or on a wearable device screen.

Accordingly, our representation techniques should be able to utilize the available resources to the extent possible and should be able to allow the users to manipulate images in an interactive

setting. So the efficiency is essentially with respect to the available computing resources and the way to make optimum use of those resources.

(Refer Slide Time: 10:32)

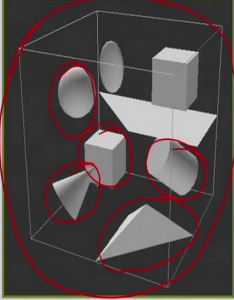


Now once we define those objects, these objects are then passed through the subsequent pipeline stages to get and render images on the screen. So the first stage is defining the objects and the subsequent stages we take these object definitions as input and generate image representation as well as render it on the screen.

(Refer Slide Time: 11:00)

Modeling Transformation (2nd Stage)

- Objects defined in their own (local) coordinate system
- Need to put them together to construct image, in its own coordinate system (world coordinate)



What are those subsequent stages? First one is modeling transformation which is the 2nd stage of the pipeline. Now as I said when we are defining an object where considering some reference frame with respect to which we are defining the object. For example, the cube that we have seen earlier. To define the cube, we need to define its coordinates but coordinates with respect to what? There we will assume certain reference frames.

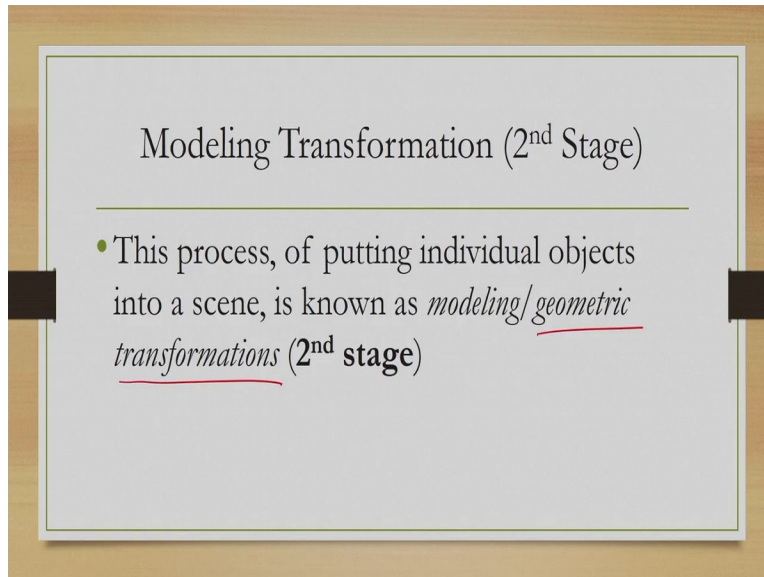
Now those reference frames with respect to which the objects are defined are more popularly called local coordinate of the object. So the objects are typically defined in their own or local coordinate system. Now multiple objects are put together to create a scene, so each object is defined in its own or local coordinate system and when we are combining them we are essentially trying to combine these different reference frames.

By combining those different objects, we are creating a new assemble of objects in a new reference frame which typically is called world coordinate system. Take the example shown on this figure. So here as you can see there are many objects, some cubes, spheres and other objects, cylinders. Each of these objects is defined in its own coordinate system.

Now in this whole scene, consisting of all the objects, this is the whole scene, here we have assembled all those objects from their own coordinate systems. But here again we are assuming another coordinate system in terms of which this assembling of objects is defined. So that coordinate system where we have assembled them is called the world coordinate system. So

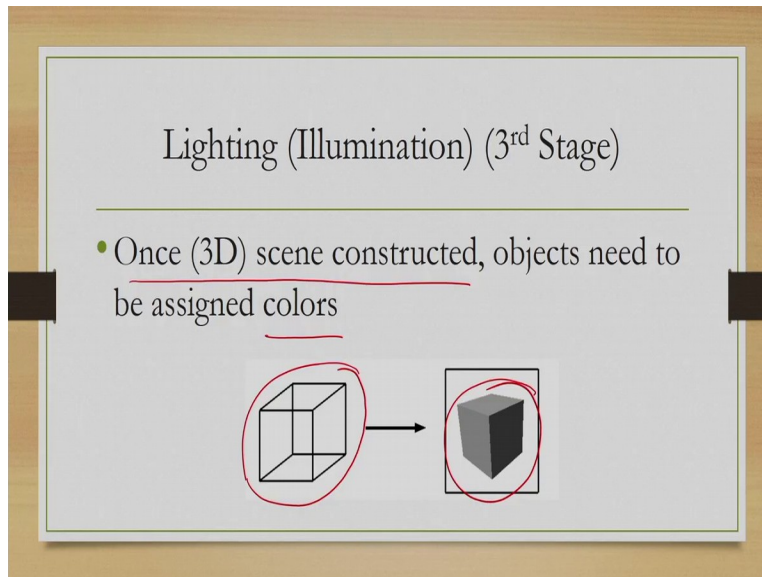
there is a transformation, transforming an object from its own coordinate system to the world coordinate system. That transformation is called modeling transformation which is the 2nd stage of the graphics pipeline.

(Refer Slide Time: 13:58)



So in the first stage we define the objects, in the second stage we bring those objects together in the world coordinate system through modeling transformation which is also sometime known as the geometric transformation. So both the terms are used either modeling transformation or geometric transformation that is the 2nd stage of the graphics pipeline.

(Refer Slide Time: 14:14)



Now once the scene is constructed, the objects need to be assigned colors which is done in the 3rd stage of the pipeline called lighting or illumination stage. Take for example the images shown here. In the left figure we have simply the object, in the right figure we have the color. So the, we have applied colors on the object surfaces. Now as you can see the way we have applied colors, it became clear which surface is closer to the viewer and which surface is further.

In other words, it gives us a sensation of 3D, whereas without colors like the one shown here, that clarity is not there. So to get realistic image which gives us a sensation of 3D, we have to assign colors. Assignment of colors is the job of the 3rd stage which is called lighting or illumination stage.

(Refer Slide Time: 15:36)

Lighting (Illumination) (3rd Stage)

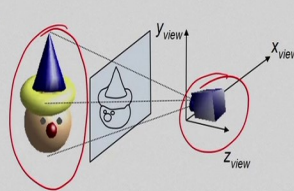
- Color is a psychological phenomena linked to the way light behaves (i.e., the laws of optics)
- Need methods that mimic optical laws – **done in the 3rd stage**

Now as probably you are aware of color is a psychological phenomenon and this is linked to the way light behaves or in other words, this is linked to the laws of optics. And in the 3rd stage, what we do? We essentially try to mimic these optical laws, we try to mimic the way we see color or we perceive color in the real world and based on that we try to assign colors in the synthesized scenes.

(Refer Slide Time: 16:17)

Viewing Transformation (4th Stage)

- Next task (4th stage) - map (colored) 3D scene to 2D *device coordinate*
- Similar to taking a snapshot of a scene with a camera

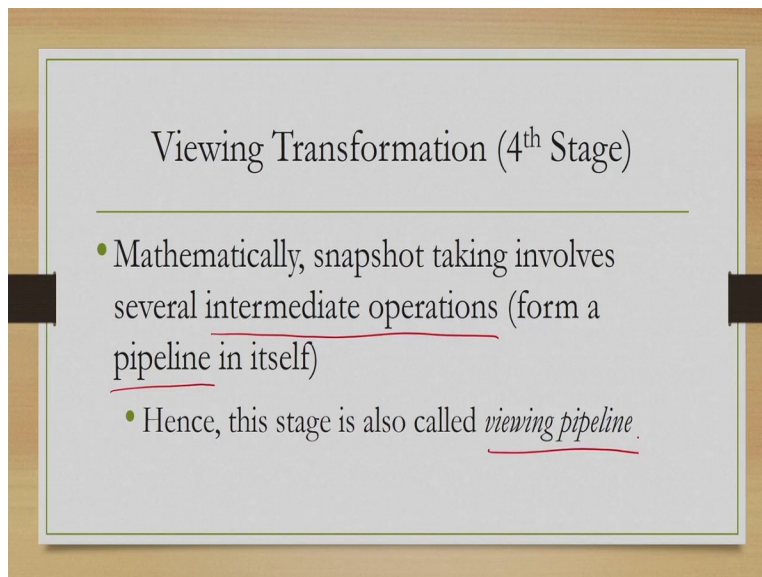


So first we define an object, 2nd we bring objects together to create a scene, 3rd stage we assign colors to the object surfaces in the scene. Now till this point, everything we were doing in 3D

setting in the world coordinate system. Now when we get to see an image, the computer screen is 2D, so essentially what we require is a mapping from this 3D world coordinate scene to 2D computer screen. That mapping is done in the 4th stage that is viewing transformation.

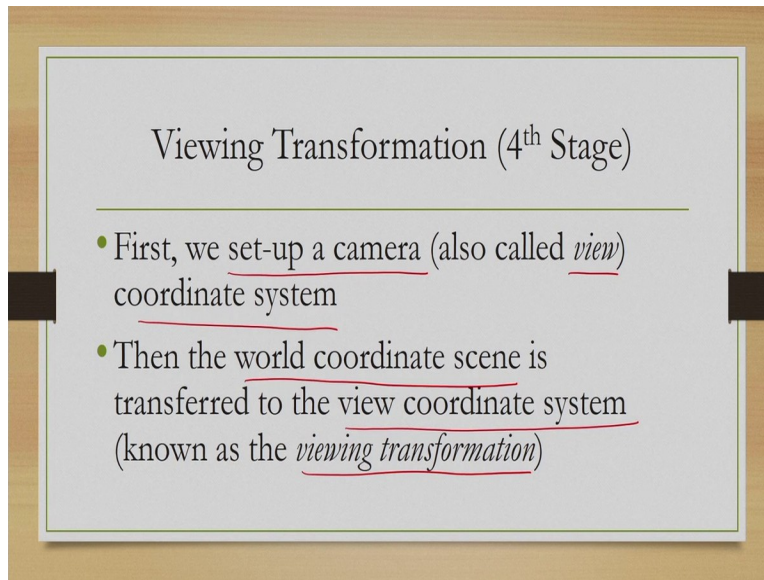
Now this stage we perform several activities which is similar to taking a photograph. Consider yourself to be a photographer, you have a camera and you are capturing some photo of a scene. What you do? You place the camera near your eye, focus to some object which you want to capture and then capture it on the camera system and also this is followed by seeing it on the camera display or camera screen, if you are having a digital camera.

(Refer Slide Time: 18:01)



Now this process of taking a photograph can be mathematically analyzed to have several intermediate operations which in itself forms a pipeline, which is a pipeline within the broader graphics pipeline. So the 4th stage viewing transformation itself is the pipeline which is a part of the overall graphics pipeline. Now this pipeline where we transform a 3D world coordinate scene to a 2D view plane scene is called viewing pipeline.

(Refer Slide Time: 18:50)



Viewing Transformation (4th Stage)

- First, we set-up a camera (also called view) coordinate system
- Then the world coordinate scene is transferred to the view coordinate system (known as the viewing transformation)

Now in this pipeline what we do? We first setup a camera coordinate system which is also referred to as a view coordinate system. Then the world coordinate scene is transformed to the view coordinate system. This stage is called viewing transformation. So we have setup a new coordinate system which is a camera coordinate system and then we transformed the world coordinate scene to the camera coordinate scene.

(Refer Slide Time: 19:30)

Viewing Transformation (4th Stage)

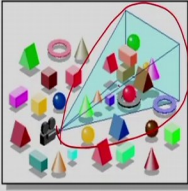
- From there, we transfer the scene to the 2D view plane (the projection transformation).

From there we make another transformation, now we transfer the scene to a 2D view plane. Now this stage is called projection transformation. So we have viewing transformation followed by projection transformation.

(Refer Slide Time: 19:49)

Viewing Transformation (4th Stage)

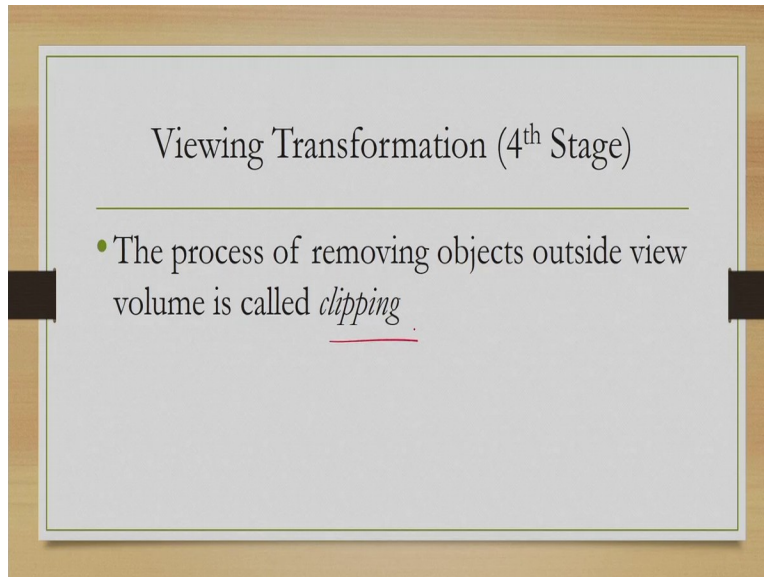
- For projection, we define a region in the viewing coordinate space (called view volume)
- Objects inside the volume are projected
- Objects that lie outside are not projected



For projection, we define a region in a viewing coordinate space which is called view volume. For example, in the figure shown here, as you can see this frustum is defining a view volume, the frustum shown here is defining a view volume. So we want to capture objects that are present within this volume, outside objects we do not want to capture. That is typically what we do when

we take a photograph, we select some region on the scene and then we capture it. So whichever object is outside will not be projected and whichever are there inside the volume will be projected.

(Refer Slide Time: 20:48)

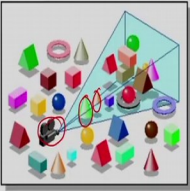


So here we require one additional process, a process to remove objects that are outside the view volume. Now those objects can be fully outside or can be partially outside. So in both the cases we need to remove them. So when an object is fully outside we completely remove it and when an object is partially outside we clip the object and keep only the part that is within the view volume, the outside part we remove. The overall process is called clipping.

(Refer Slide Time: 21:22)

Viewing Transformation (4th Stage)

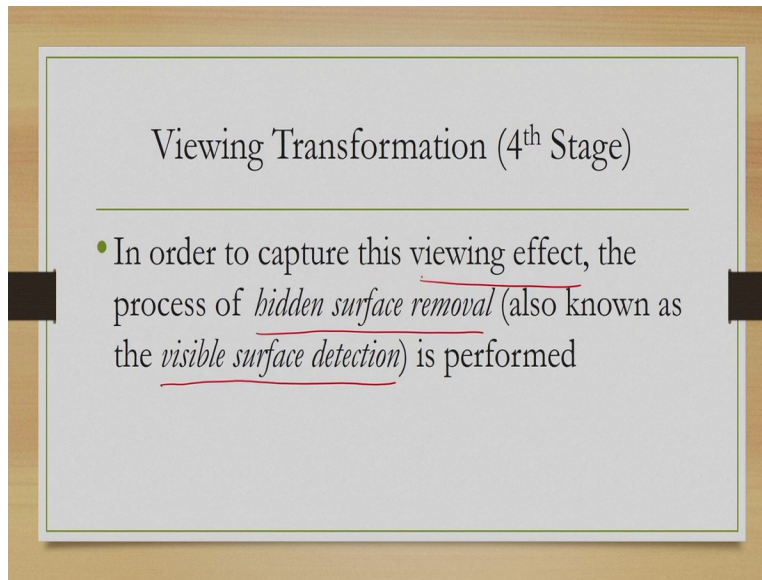
- When we project, we consider a viewer position
 - Some objects will be fully visible, some partially while some will be invisible, although all are within the same volume



Also when we are projecting, we consider a viewer position where the photographer is situated and in which direction he or she is looking at. Based on that position, some objects may appear fully visible, some may appear partially visible, whereas the other objects will become invisible. But all of them may be within the same volume.

For example, with respect to this particular view position, some objects may get, like this object if it is behind this object then it will be invisible. If it is partially behind, then it will be partially visible and if they are not aligned in the same direction, then both of them will be fully visible. So you take care of this fact also before projection which requires some further operations, computations.

(Refer Slide Time: 22:32)

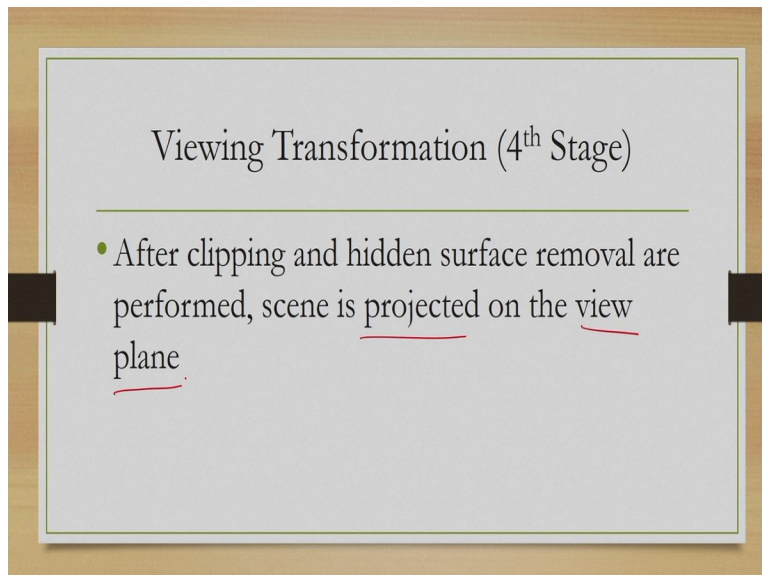


Viewing Transformation (4th Stage)

- In order to capture this viewing effect, the process of hidden surface removal (also known as the visible surface detection) is performed

So to capture this viewing effect, the operations that we perform are typically called hidden surface removal operations or similarly visible surface detection operations. So to generate realistic viewing effect along with clipping what we do is we perform the hidden surface removal or visible surface detection operations.

(Refer Slide Time: 23:06)

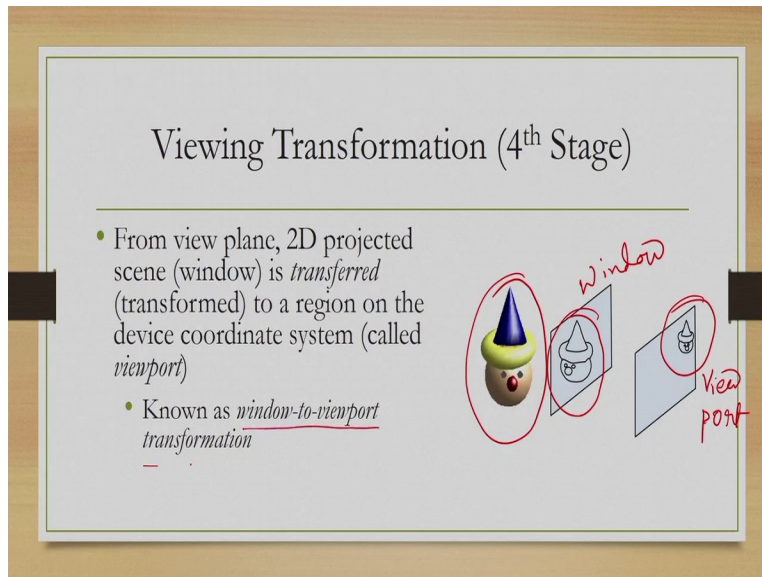


Viewing Transformation (4th Stage)

- After clipping and hidden surface removal are performed, scene is projected on the view plane.

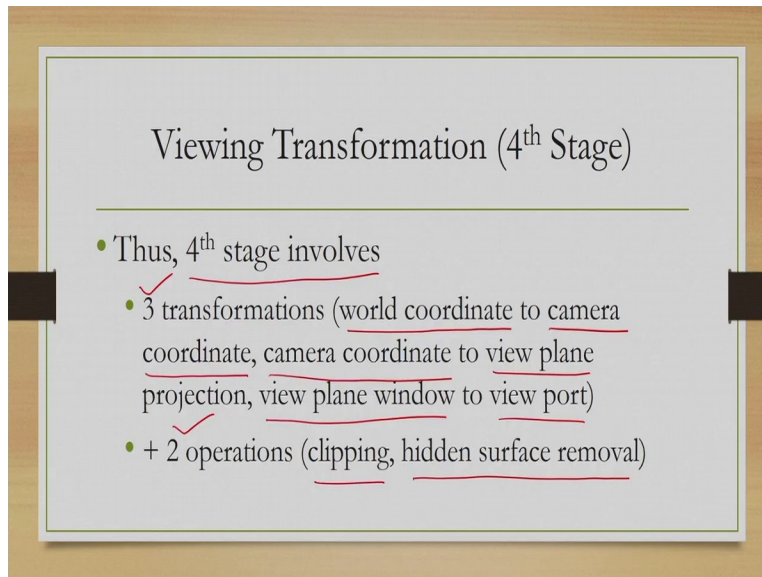
So after clipping and hidden surface removal operations, we project the scene on the view plane. That is a plane define in the system, in the view coordinate system.

(Refer Slide Time: 23:21)



Now, there is one more transformation. Suppose in the right hand figure, suppose this is the object which is projected here in the view plane. Now the object may be displayed on any portion of a computer screen, it need not to be exactly at the same portion as in the view plane. For example, this object may be displayed in a corner of the display. So we will differentiate between two concepts here; one is the view plane which is typically called a window, other one is the display region on the actual display screen which we call viewport. So one more transformation remains in the viewing pipeline that is transferring the content from window to the viewport. So this is called the window-to-viewport transformation.

(Refer Slide Time: 24:44)



So in summary what we can say is that, in the 4th stage there are 3 transformations. What are those transformations? First we transform from world coordinate scene to camera or view coordinate scene. Then from camera coordinate scene, we perform the projection transformation to view plane, then the view plane window is transform to the viewport. So these are the 3 transformations.

Along with those there are 2 major operations that we perform here; one is clipping that is clipping out the objects that lie outside the view volume and the other one is hidden surface removal which means creating a realistic effect, viewing effect with respect to the viewer position. So that is the 4th stage.

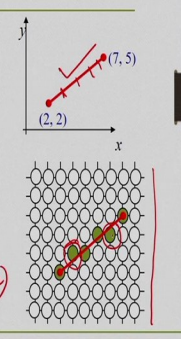
So first we defined objects in the first stage, in the 2nd stage we combined those objects in the world coordinate scene, in the 3rd stage we assigned colors to the object surfaces in the world coordinate scene, in the 4th stage we transformed in the world coordinate scene to the image on the viewport through a series of transformations which form a sub-section-pipeline within the overall pipeline.

And those sub-pipeline stages are viewing transformation, projection transformation and window-to-viewport transformation. This sub-pipeline is called viewing pipeline which is part of the overall graphics pipeline and in the 4th stage along with these viewing pipeline we also have to more operations performed that is clipping and hidden surface removal.

(Refer Slide Time: 27:17)

✓ Scan Conversion (5th Stage)

- Device coordinate is a continuous space - display contains discrete space (pixel grid)
 - Need to transfer viewport on the (continuous) device coordinate to (discrete) screen coordinate system
- Called *scan conversion* (also called *rasterization*)
 - 5th stage



One more stage remains that is the 5th stage which is called scan conversion or rendering. Now we mentioned earlier that we transform to a viewport. Now viewport is an abstract representation of the actual display. In the actual display if you recollect our discussion on our raster displays, we mentioned that the display contains a pixel grid.

So essentially the display contains locations which are discrete, we cannot assume that any point can have a corresponding point on the screen. For example, if in our image we have a vertex at location 1.5 and 2.5, on the screen we cannot have such a location because on screen we only have integer values as coordinates due to the discrete nature of the grid. So we have either a pixel located at 2, 2 or 3, 3 or 1, 1 or 1, 2 something like that rather than the real numbers 1.5, 2.5.

So we cannot have a pixel location at say 1.5, 2.5 but we can have pixel locations only at integer value say 1, 1; 2, 2 and so on. So if we get a vertex in our image located at 1.5, 2.5 then we must map it to these integer coordinates. That stage where we perform this mapping is called the scan conversion stage which is the 5th and final stage of the pipeline. For example, consider these lines shown here, the end points are 2, 2 and 7, 5. Now all the intermediate points may not have integer coordinate values but in the final display, in the actual display we can have pixels, these circles only at integer coordinate values.

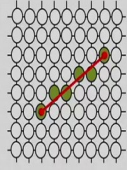
So we have to map these non-integer coordinates to integer coordinates. That mapping is the job of this 5th stage or scan conversion stage which is also called rasterization. And as you can see it

may lead to some distortion because due to the mapping we may not get the exact points on the line, instead we may have to satisfy ourselves with some approximate points that lies close to the actual line. For example, this pixel here or this pixel here is not exactly on the line but the closest possible pixel with respect to the line.

(Refer Slide Time: 30:19)

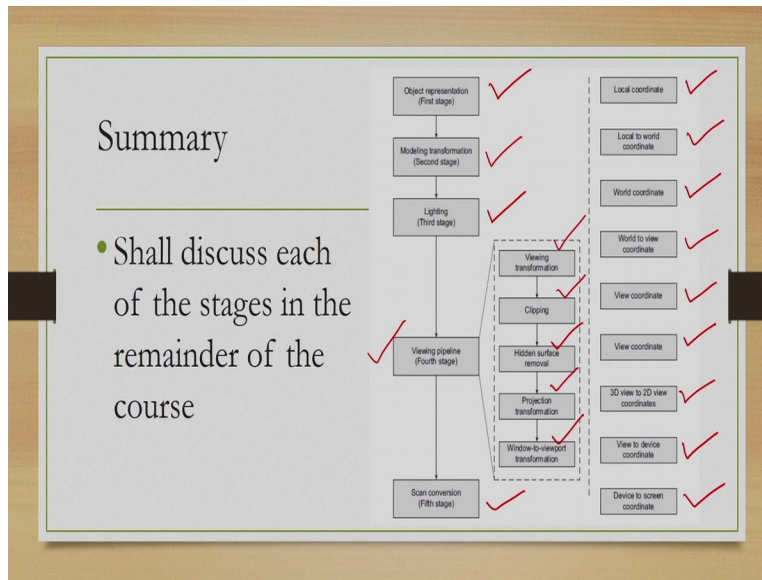
Scan Conversion (5th Stage)

- Concern - how to minimize distortions (called aliasing effect) that result from the transformation from continuous to discrete spaces?
- Solution - anti-aliasing techniques (used in this stage)



So what is the concern? How to minimize the distortions? Now these distortions has a technical name which is called aliasing effect, from where this name originated we will discuss later. So our concern is to eliminate or reduce the aliasing effect to the extent possible so that we do not get to see too much distortions, we do not get to perceive too much distortions. To address this concern, several techniques are used which are called anti-aliasing techniques. These are used to make the image look as smooth as possible to reduce the effect of aliasing.

(Refer Slide Time: 31:21)



So let us summarize what we have discussed so far. We mentioned the graphics pipeline, it contains the 5 stages; 1st stage is object representation, 2nd stage is modeling transformation, 3rd stage is assigning colors or lighting, 4th stage is the viewing pipeline which itself has sub-pipeline involving viewing transformation, clipping, hidden surface removal, projection transformation and window-to-viewport transformation and the final stage is scan conversion which is the 5th stage. So there are broadly 5 stages involved.

Now each of these stages has its own reference frames, own coordinate system. So in stage 1 we deal with local coordinate system of the objects, in stage 2 we deal with world coordinate system. So essentially it transforms from local to world coordinate system. Stage 3 again we deal with world coordinate. So when we are assigning color, we essentially assuming that the objects are defined in the world coordinate system.

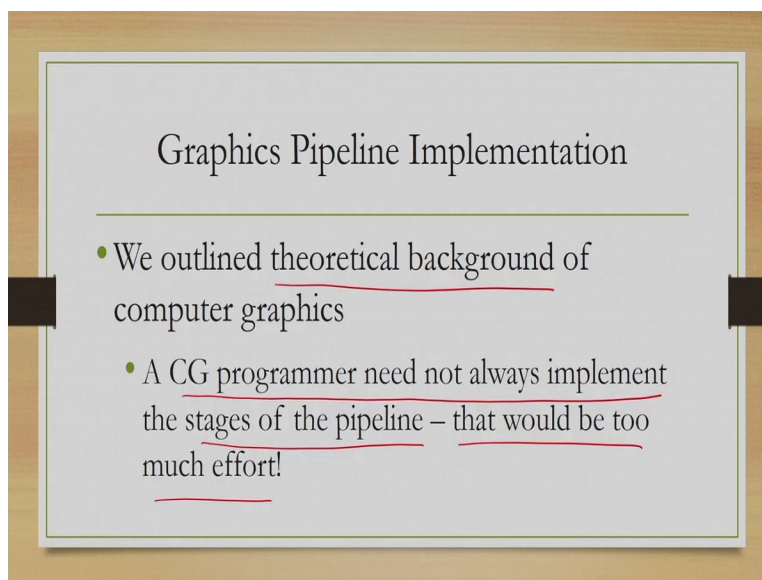
In stage 4 again different coordinates are used, so first transformation that is viewing transformation, involve transformation from world coordinate to view coordinate system or the camera coordinate system. Clipping is performed on the view coordinate system, hidden surface removal is also performed in the view coordinate system. Then we perform a transformation, projection transformation which transforms the content of the view coordinate system to 2D view coordinate system.

So we have 3D view coordinate system from there we transfer the content to a 2D view coordinate system. And in the window-to-viewport transformation, we transfer from this 2D view coordinate system to device coordinate system. And finally in the 5th stage what we do, we transfer from device coordinate to actual screen coordinate system. Note that device coordinate is an abstract an intermediate representation, whereas the screen coordinate is the actual pixel grid.

So device coordinate contains continuous values, whereas screen coordinate contains only discrete values in the form of a grid. So this is in summary what is there in the graphics pipeline. So display controller actually performs all these stages to finally get the intensities values to be stored in the frame buffer or video memory. Now these stages are performed through software, of course with suitable hardware support.

For a programmer of a graphic system, of course it is not necessary to learn about the intricate details of all these stages, they are quite involves lots of theoretical concepts, lots of theoretical models. Now if a graphics programmer gets brought down with all this theory, models then most of the time will be consumed by understanding the theory rather than actually developing the system. So in order to address this concern of a programmer what is done is essentially development of libraries, graphics libraries.

(Refer Slide Time: 35:17)



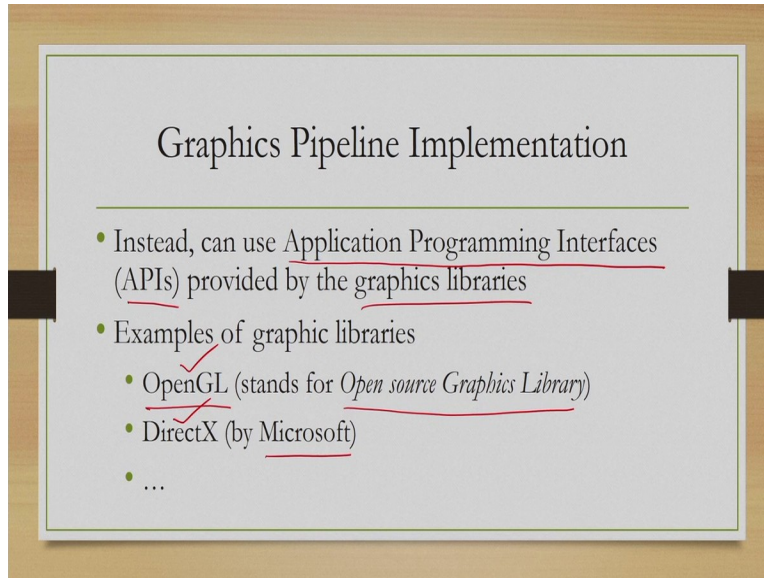
Graphics Pipeline Implementation

- We outlined theoretical background of computer graphics
- A CG programmer need not always implement the stages of the pipeline – that would be too much effort!

So there is this theoretical background which is involved in generating 2D image. The programmer need not always implement the stages of the pipeline to fully implement the

theoretical knowledge, that would be of course too much effort and major portion of the development effort will go into understanding and implementing the theoretical stages.

(Refer Slide Time: 35:52)

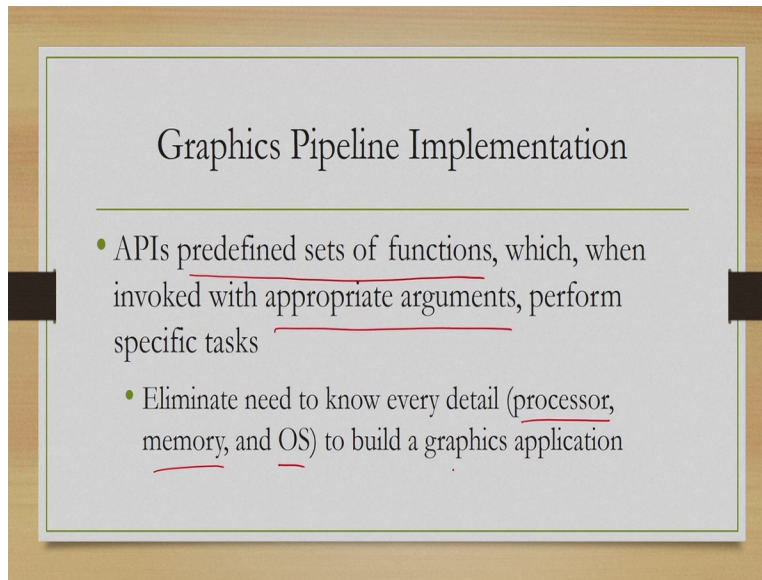


Graphics Pipeline Implementation

- Instead, can use Application Programming Interfaces (APIs) provided by the graphics libraries
- Examples of graphic libraries
 - OpenGL (stands for Open source Graphics Library)
 - DirectX (by Microsoft)
 - ...

Instead the programmer can use what is called application programming interfaces or APIs provided by the graphics libraries. Where these stages are already implemented in the form of various functions and the developer can simply call those functions with arguments in their program to perform certain graphical tasks. There are many such libraries available, very popular ones are mentioned here OpenGL which is an open source graphics library which is widely used. Then there is DirectX by Microsoft and there are many such other commercial libraries available which are proprietary but OpenGL being open source is widely accessible and useful to many situations.

(Refer Slide Time: 37:00)

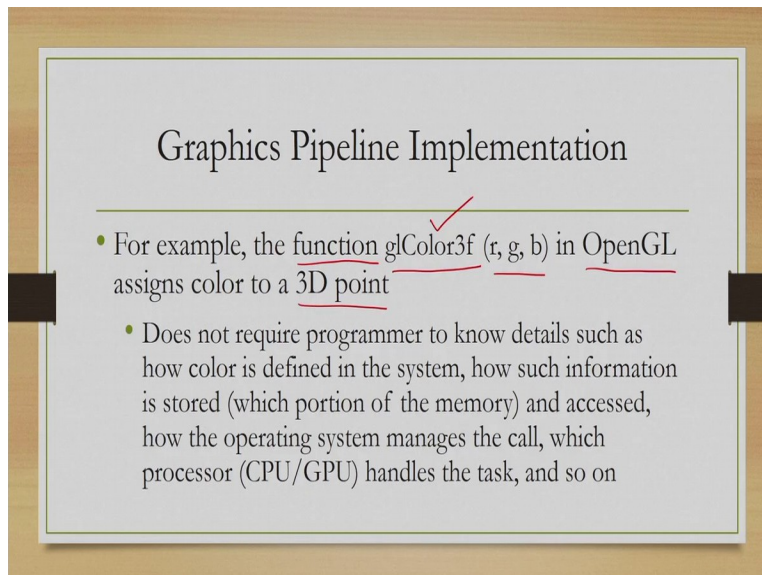


Graphics Pipeline Implementation

- APIs predefined sets of functions, which, when invoked with appropriate arguments, perform specific tasks
- Eliminate need to know every detail (processor, memory, and OS) to build a graphics application

Now what these libraries contains? They contain predefined sets of functions, which, when invoked with appropriate arguments, perform specific tasks. So the programmer need not know every detail about the underlying hardware platform namely processor, memory and OS to build a application.

(Refer Slide Time: 37:29)



Graphics Pipeline Implementation

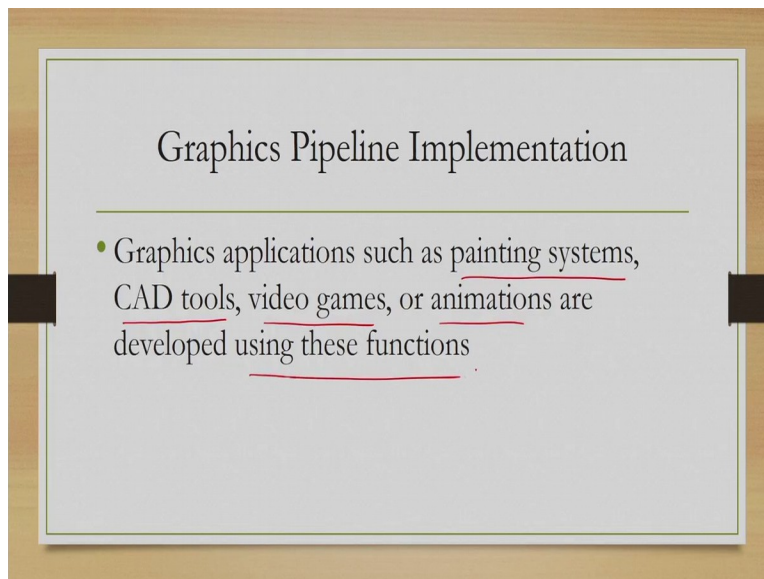
- For example, the function glColor3f (r, g, b) in OpenGL assigns color to a 3D point
- Does not require programmer to know details such as how color is defined in the system, how such information is stored (which portion of the memory) and accessed, how the operating system manages the call, which processor (CPU/GPU) handles the task, and so on

For example, suppose we want to assign colors to an object we have modelled. Do we need to actually implement the optical laws to perform the coloring? Note that this optical law implementation also involves knowledge of the processors available, the memory available and

so on. So what we can do is instead of having that knowledge, we can simply go for using a function `glColor3f` with an argument `r`, `g`, `b`. So this function is defined in OpenGL or the open graphics library which assigns a color to a 3D point.

So here we do not need to know details such as how color is defined in the system, how such information is stored, in which portion of the memory and accessed, how the operating system manages the call, which processor CPU or GPU handles the task and so on. So all these complicated details can be avoided and the programmer can simply use this function to assign color. We will come back to this OpenGL functions in a later part of the lecture where we will introduce OpenGL.

(Refer Slide Time: 38:57)

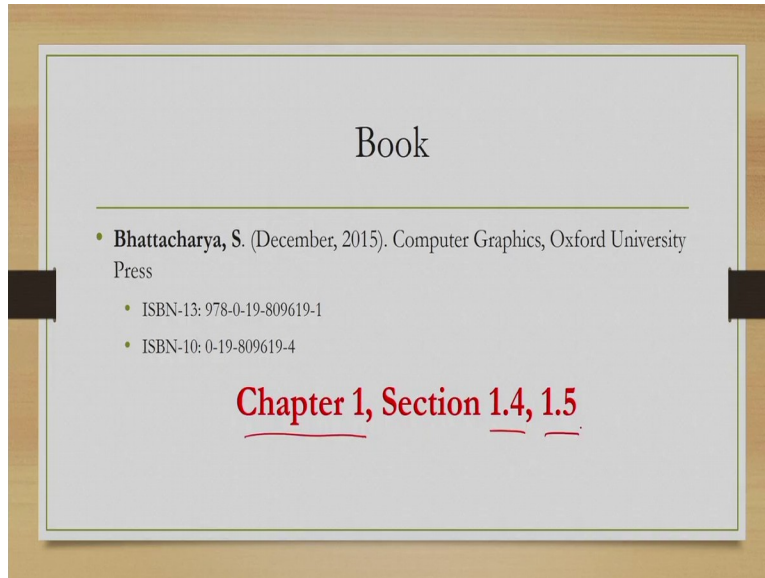


Now graphics applications such as painting systems which probably all of you are familiar with, CAD tools that we mentioned in our introductory lectures earlier, video games, animations, all these are developed using these functions. So it is important to have an understanding of these libraries if you want to make your life simpler as a graphics programmer. And we will come back later to this library functions, we will discuss in details some functions popularly used in the context of OpenGL.

So in summary today we have learned some idea of 3D graphics pipeline and also got some idea, introductory idea to the graphics libraries. In subsequent portion of the course, we will discuss in

details all the stages as well as some more details of the graphics libraries and the graphics hardware.

(Refer Slide Time: 40:26)



That is all for today, whatever I have discussed today can be found in chapter 1 of the book mentioned here. You are advised to refer to section 1.4 and 1.5. Thank you and good bye.