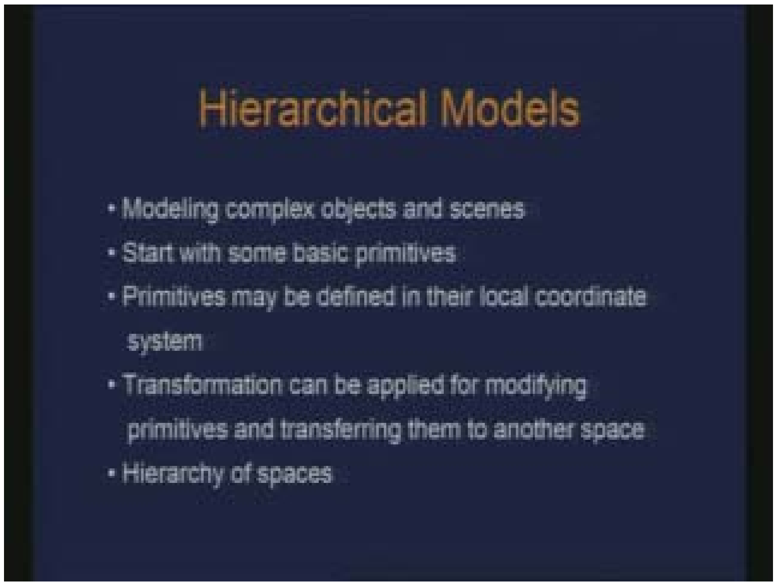
Introduction to Computer Graphics
Dr. Prem Kalra
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

Lecture # 20
Hierarchical Models

Today we are going to talk about modeling of some complex scenes or models using hierarchical modeling. We have looked at building models using surfaces, solids etc. Now our interest is how we make use of those individual models to build bigger models. Let us look at the hierarchical models.

(Refer Slide Time: 01:43)



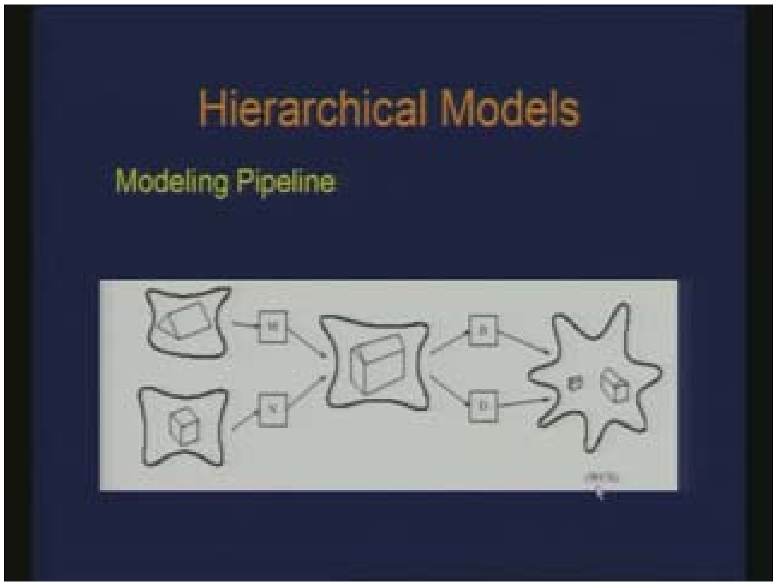
Hierarchical Models

- Modeling complex objects and scenes
- Start with some basic primitives
- Primitives may be defined in their local coordinate system
- Transformation can be applied for modifying primitives and transferring them to another space
- Hierarchy of spaces

Here the interest is that we would like to do modeling of complex objects and scenes. So again the idea is that we start with some basic primitives, individual models and these primitives may be defined in their own local coordinate system. So you can build a primitive or a model in its own coordinate system but then the question is about transferring that model or a primitive to the scene. For that we may require some transformations. There could be several roles of transformations that we are going to see.

So, at the end what we basically form is actually a hierarchy of spaces. So there is a space of model, there is a space of collection of some models or primitives and then there is a space of scene or world. So there is a hierarchical structure to the description of the scene. This is the underlying principle which you are also going to use for the assignment of building a car. Last time actually we looked at a sub pipeline of the complete graphics rendering pipeline known as modeling pipeline, so what does that constitute? That is what is basically captured in the hierarchical structure of the modeling.

(Refer Slide Time: 03:51)



Hierarchical Models

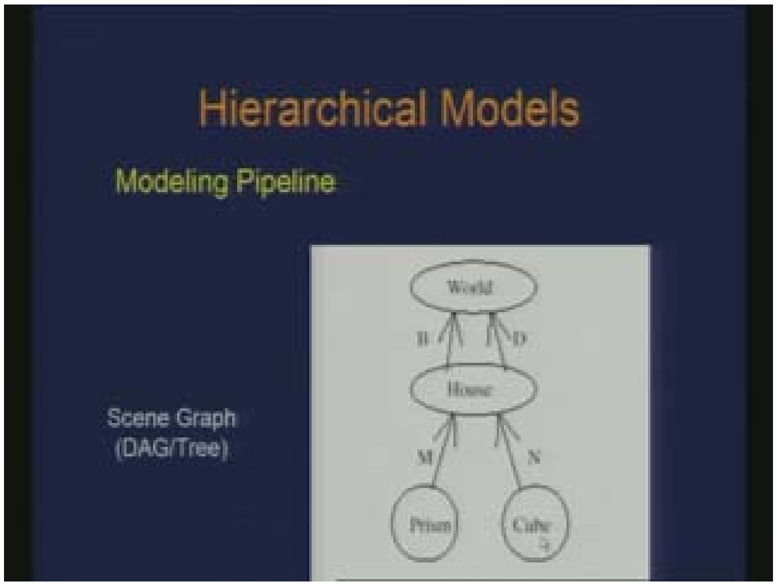
Modeling Pipeline

What we have here is that if you want to build a world or a scene it has its own coordinate system which is world coordinate system and then you have individual primitives or models and their own local coordinate system. Then there are certain transformations which are required either to instantiate these primitives or to assemble them and build another model.

A house could basically be built of the roof structure which is some sort of a prism and a cuboid structure which is the body of the house. So there are some transformations required say M and N which are needed to assemble these two primitives to form this. Further you may have set of transformations which may make use of this as a model now to have a bigger house or a smaller house and where to place those houses to build a scene of colony or some system.

Again you have certain other transformations like B and D which would be applied to this model to generate this world. That is the philosophy which is there in the hierarchical model. So all we need is models or primitives and set of transformation which are required for building your entire scene. Now as a representation of the entire scene I can look at this as some sort of a graph a scene graph which could be something like a directed acyclic graph or a tree.

(Refer Slide Time: 06:37)



Hierarchical Models

Modeling Pipeline

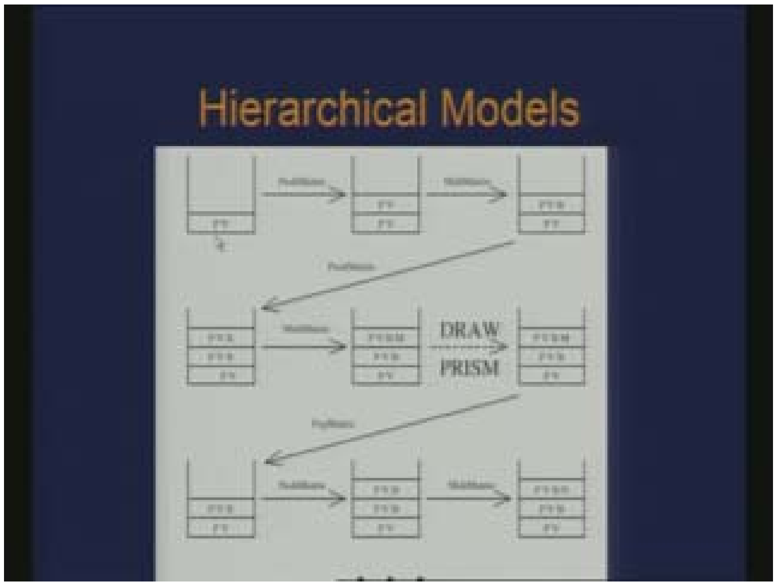
Scene Graph (DAG/Tree)

$$\begin{tikzcd}
 & \text{World} & \\
 & \text{House} \arrow[u, shift left, "D"'] \arrow[u, shift right, "B"] & \\
\text{Prism} \arrow[ur, "M"] & & \text{Cube} \arrow[ul, "N"']
\end{tikzcd}$$

You have a definition of world there, you have this definition of the house and these links are nothing but the transformations which get applied to this node of house and then this house has children of the primitives like prism and cube. So the description of your scene can be basically defined as a graph and we call it as scene graph.

Now the question is, when you are using a library like OpenGL or any other graphics rendering library or even your own library then what is the mechanism you can use for drawing or displaying these scenes. One of the ways in which you can think of this is some sort of a stack of matrices. If you are displaying a model like a house then let us assume that you already have the information about the transformation like the projection transformation and the viewing transformation.

(Refer Slide Time: 07:36)



Therefore, what we are trying to do here is basically build a stack of matrices and the building process is through commands like push matrix, push the matrix into the stack which actually means that I already have an element there in the stack and I copy that to have the top of the stack as the same as this, this is just a copying process through push. Then what I do is I multiply by a current transformation matrix which was B so this entry in the stack basically gets multiplied by the matrix which I pass through mult matrix. So the current top of the matrix becomes P V B.

Now what I am trying to do is if I am trying to build this world the question is how I use the information from the bottom of the primitives which are defined in the scene graph to come to this. So I am using a mechanism of stack. And these matrices or these transformation matrices are the entries to the stack of the matrix. This is where I get the appropriate matrix definition to be able to have the construction of the house. That is what B does as B brings the house to the scene.

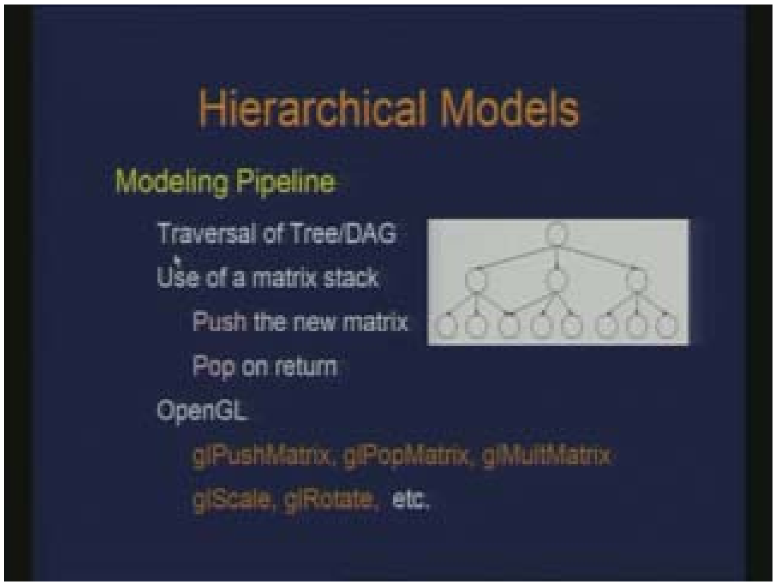
Now what I do is I push the matrix again so again it copies this entry and the current entry is again P V B because I am going to follow this push matrix with a multiplication. So I again multiply with the matrix M which is for transferring the primitive or the model of prism for making the house and that is the necessary transformation I require.

Therefore now I have the matrix or the stack of the matrix consisting of the transformation M which takes the primitive or the model prism to build a house. This B matrix is there to take the model of the house to the world and then I have these matrices for the winged transformation and the projection of the viewing. Now I can do a draw of prism. So all we are doing is basically using a stack of matrix and then in order to restore back what I use is a pop matrix. At this instance I can draw the model and then if pop it so I get back to a level below which is the current entry as P V B. So now I am at the level of house. Again I can push the matrix which copies this entry to this and I multiply the matrix which is necessary to take the cube to the house which is N. This whole process shows you how I can use the description through the scene graph to draw the world here.

This is what you will also be doing using commands like push matrix, pop matrix, mult matrix. Again we trying to say that we have the modeling pipeline which is represented as a scene graph or a tree and all I am basically referring to is a traversal of that graph or the tree which would build the scene. And I use a matrix stack in order to allow the various transformations embedded in the scene graph and I have the commands like push and pop.

Push basically embeds a new matrix or instead a new entry to the stack or allows to enter to the stack and pop is on return. Once I have done that job I get to the top of the stack. So with regards to OpenGL the corresponding commands you have are glPushMatrix, glPopMatrix, glMultMatrix and also you may have other transformation through matrices like glScale, glRotate which could also help in terms of getting an instance of a model.

(Refer Slide Time: 12:45)

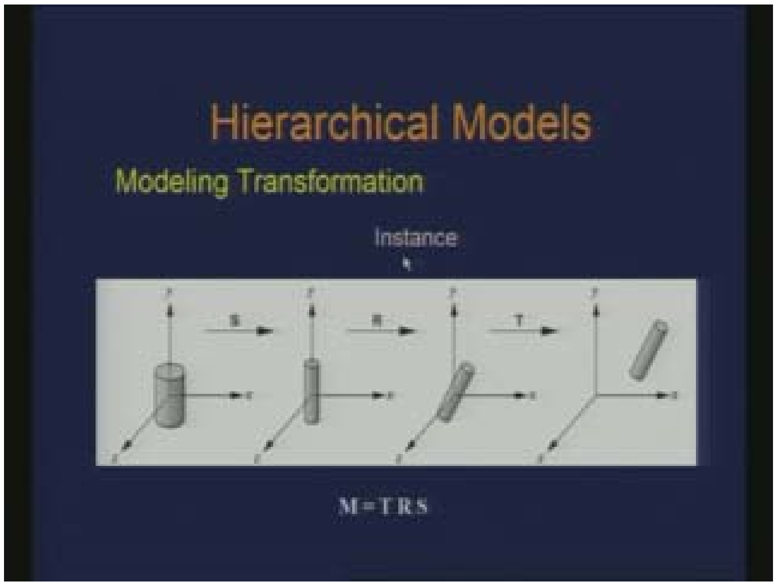

Hierarchical Models

Modeling Pipeline

- Traversal of Tree/DAG
- Use of a matrix stack
  - Push the new matrix
  - Pop on return
- OpenGL
  - glPushMatrix, glPopMatrix, glMultMatrix
  - glScale, glRotate, etc.

I can have transformations which I call it as modeling transformations which are basically doing two kinds of jobs. One is to have an instance of a primitive and the other is the transform from one coordinate system to another coordinate system.

(Refer Slide Time: 14:45)


Hierarchical Models

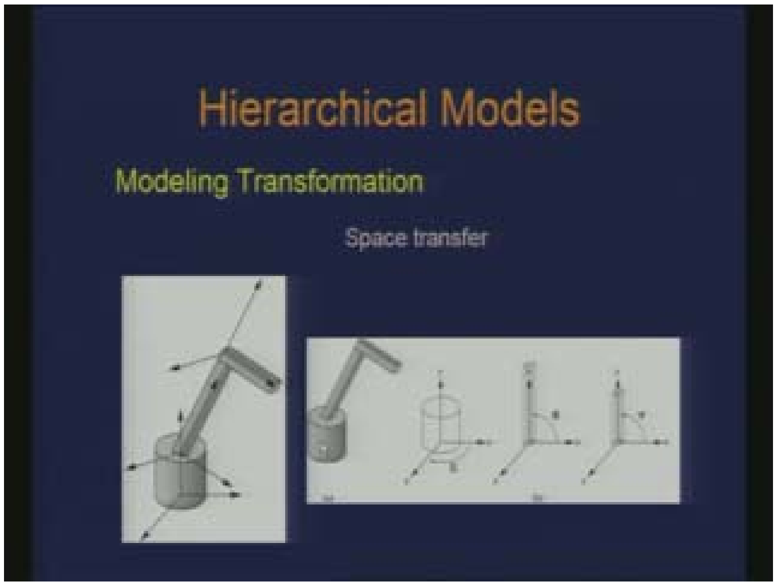
Modeling Transformation

Instance

M = T R S

So here if I have a canonical or a generic description of the primitive as a model of cylinder here and my modeling requirement is that I should have a cylinder in this way so I need a series of transformations which would transform this cylinder to this which is nothing but transformations like scaling, rotation, translation and so on. Therefore all it says is that there is a modeling transformation M as a composition of T, R and S which would take this model and give me an instance K. So this is one way of using the modeling transformation.

(Refer Slide Time: 15:45)


Hierarchical Models

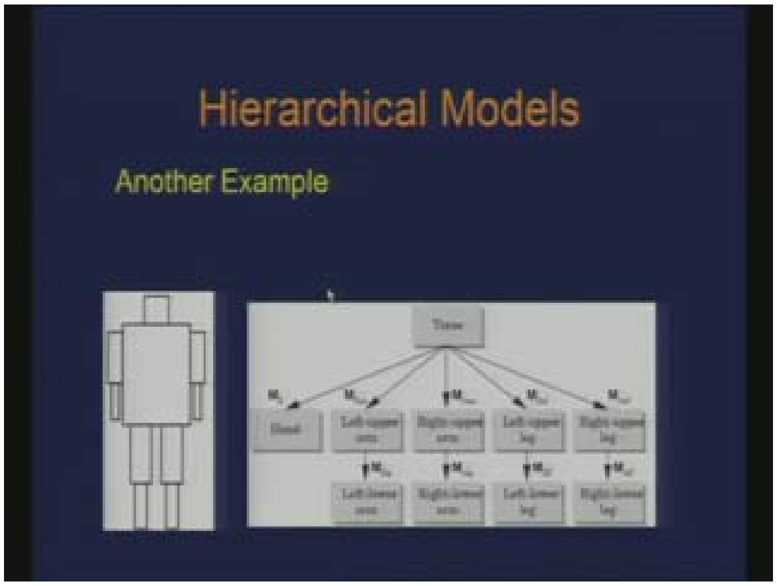
Modeling Transformation

Space transfer

The other thing is some sort of a space transformation where, if I am interested in building some kind of a robot or arm of the robot so I could have a necessary definition of each of the components of this robot, this is the body here, this is the upper arm, this is the lower arm. And each of these could be defined in its own local coordinate system. And eventually I want to build something like this. So all it requires is a necessary transformation which would take this to here, in this case I may actually keep this as my reference coordinate system to build on the top of the rest of the model and take this to have a model here inserted in this coordinate system. All it requires is some kind of a transformation from here to here, similarly from here to here. So there could be manipulations done at this level then I would again require a transfer of information from here to the rest so there is a notion of hierarchy.

For instance, if I want to move only the lower arm here these may not move but the moment I move this then the rest will move. So there is a notion of hierarchy. The body is at the top and when I move the body everything else also moves with it. Here the other application of modeling transformation is some sort of a space transfer from one coordinate system to another coordinate system. Here you have some kind of a humanoid structure. There is a torso of a humanoid. Again that is the root of the tree which you are building and then the other parts like head, left upper arm, right upper arm, left upper leg, right upper leg and so on.

(Refer Slide Time: 18:09)

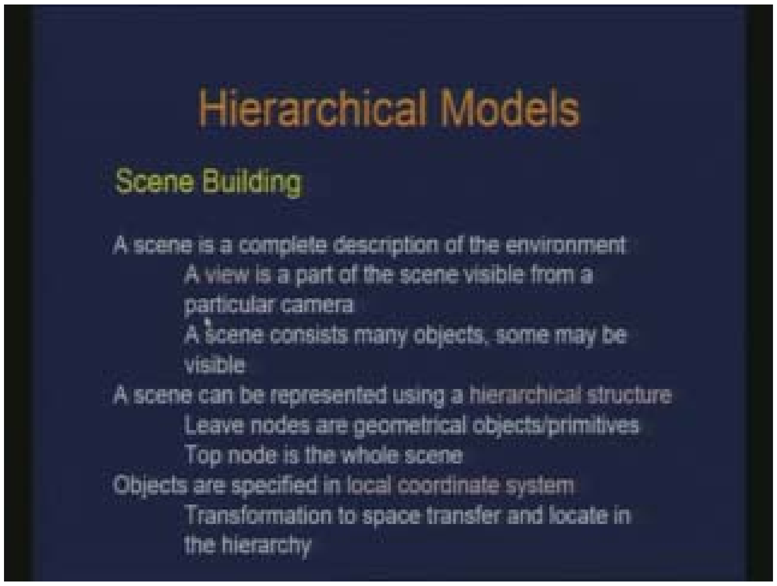


Hierarchical Models

Another Example

And each of these links are basically capturing the transformation I need going from here to there. This is the hierarchy of all the models I want to compose to build a structure like a humanoid. We are interested in building complex scenes or environments so we describe the scene as a description of the environment. We consider the scene as a description of the environment or the world which we are building and a view is a part of the scene visible from a particular camera. So you build the entire scene and then you insert the viewing transformation with respect to the camera you specified. So a view of that scene is just a scene which is visible with respect to the camera you insert and that is what gets determined by transformations like B and T.

(Refer Slide Time: 19:23)



Hierarchical Models

Scene Building

A scene is a complete description of the environment
- A view is a part of the scene visible from a particular camera
- A scene consists many objects, some may be visible

A scene can be represented using a hierarchical structure
- Leave nodes are geometrical objects/primitives
- Top node is the whole scene

Objects are specified in local coordinate system
- Transformation to space transfer and locate in the hierarchy

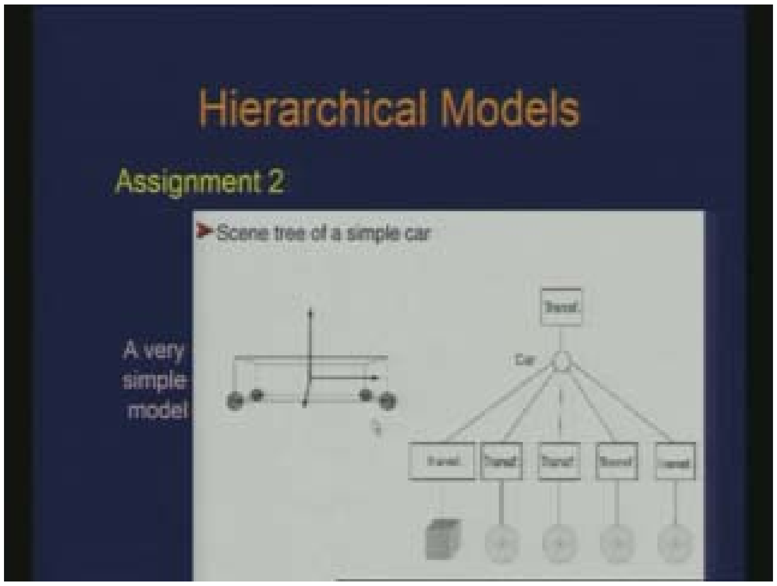
Then the scene in fact consists of many objects. Some may be visible some may not be visible with respect to the camera you incorporate. Hence, as far as the part of determining the visibility is concerned you are just giving that to some kind of a graphics engine in your case OpenGL to look after the visibility. So you are not explicitly determining the visibility of the individual objects with respect to the other objects that is being done by OpenGL. But it requires you to handle the visibility.

So what we have seen is that the scene can be represented as a hierarchical structure where the leave nodes are the geometrical objects of the primitives and the top node is the entire scene which you want to build and individual objects are specified in local coordinate system. You may add transformation to transfer from one coordinate system to another coordinate system to locate individual objects in the hierarchy of the scene. That is what the hierarchical modeling does for building a scene.

So it helps you in many ways because if you want to do a manipulation of individual components then you can do it. If you have a monolithic description of the scene, if you just consider collection of points or collection of triangles then you would not be able to manipulate the components because you will have to build those components. That is the advantage here we have with hierarchical models. You can embed the manipulative or the functional requirements to the hierarchy. Basically your assignment is going to make use of hierarchical modeling.

Your assignment basically deals with building a model of a car. In this case you can consider the scene to be a car or you may actually consider again a scene of one level higher than the car where you have some road or some other environment where you insert the car. But the primary concern is that you build the model of the car in some sort of a hierarchical fashion. If I consider a very simple model of the car where I define a body something like a cuboid and then there are these four wheels here.

(Refer Slide Time: 24:03)



Hierarchical Models

Assignment 2

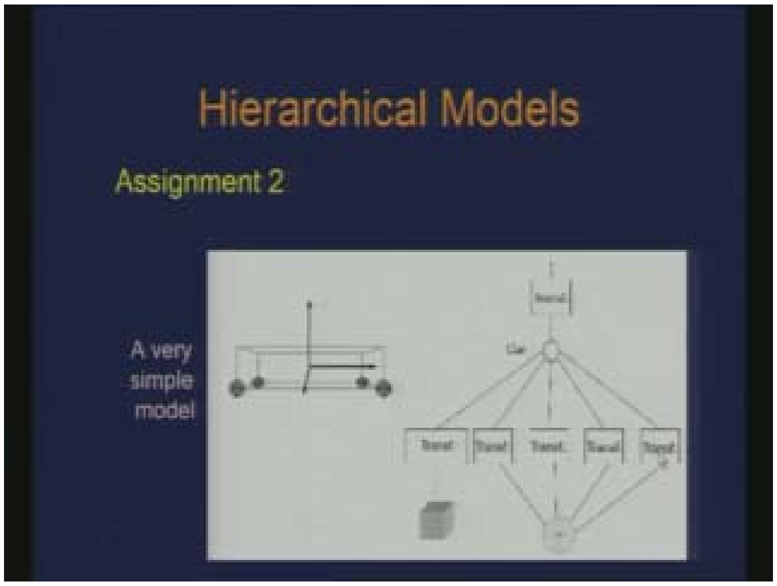
- Scene tree of a simple car

A very simple model

If I am interested in getting the corresponding tree or the graph of the scene with respect to such a model that is what is getting captured here. So you have the body here which is some sort of a cuboid, some transformation of this would make the body of the right aspect ratios and size and would place it in the coordinate system where I want to build the car and then there are primitives to define the individual wheel. A wheel is undergoing some transformation to get the left front wheel or the right front wheel and so on. This is the corresponding tree of such a model. But I definitely want you to have a better model in this. But that is just to show you the structure in a simple model as to how the hierarchy is basically built.

Now in fact once when you see these four individual models of a wheel and if you consider that each of the wheel is actually the same model it is a congruent element then what I can do is I can just take the instance of that and build the individual wheel. So I can modify this as something like this where this wheel is here now and I have the necessary transformation which would in turn define whether it is a left front wheel or the right front wheel and so on.

(Refer Slide Time: 26:00)

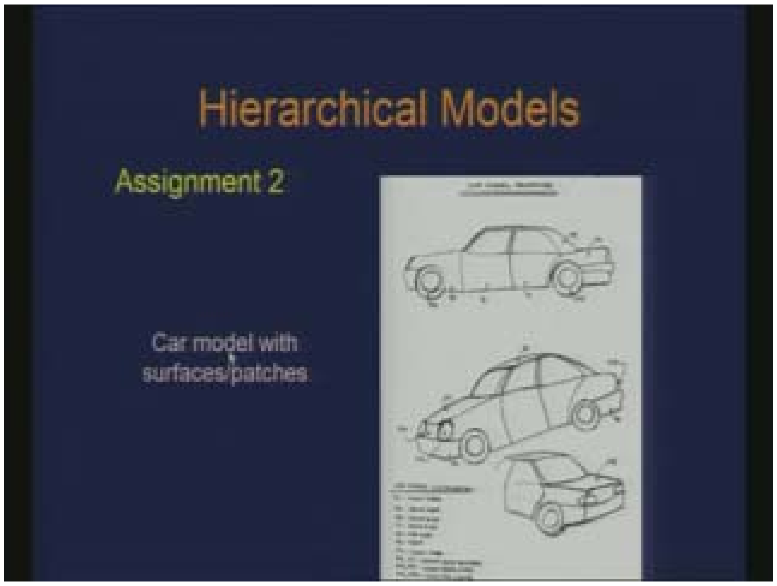

Hierarchical Models

Assignment 2

A very simple model

If I am trying to say that the motion of this wheel should get converted into the motion of a car so I have to link that. The rotational motion of the wheel should actually get linked with the linear motion of the car and that is what happens functionally. So you need a transformation which is getting originated by the rotation of the wheel as a translation of the whole body. This was just a simple illustration of a car model.

(Refer Slide Time: 27:33)


Hierarchical Models

Assignment 2

Car model with surfaces/patches

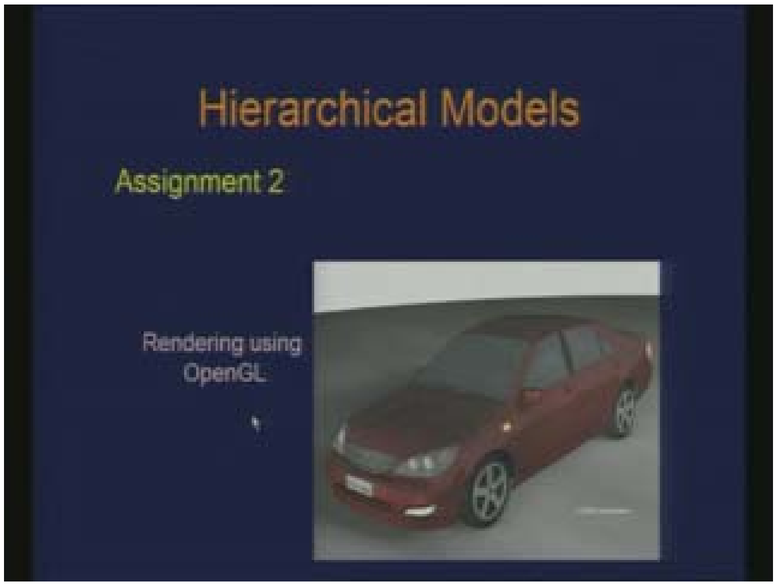
The model for your assignment is a prototype model given through a sketch. You have a front body here, doors, windows, wheels etc so there are nearly fourteen or fifteen surfaces on the patches. There we considered a car body just like a cuboid but here we want you to construct the body as a collection of surfaces and patches. And there you are free to choose what kind of patch or surface you want to construct. It could be Bezier surface, it could be Coon's patch, it could be B-Spline surface etc. You have complete flexibility and freedom to choose from.

In fact there is also a reference given to the blue prints of certain car models. You can even observe the dimensions for the various parts of the car and make your model consistent to that. Therefore you can build a car of your choice. But how do you take car of the hidden surface part? You have already used the OpenGL capability of doing hidden surface elimination using Z buffer. So all you have to do is enable your Z buffer. So Z buffer should be on, then OpenGL takes care of the rest, you do not have to do anything. Then the other thing which you will also require to use is double buffers.

Double buffer does something like this that you have two buffers so you keep writing on one of the buffers and then swap so you do not see the flicker of the change of the buffers. It is a continuous seamless display and is also required for animation. As far as the rendering is concerned you can have various modes of rendering. One of the modes could be wireframe display. Thus, all you have to do is render the piecewise lines which are present for describing this model. OpenGL also has nurves, OpenGL also has other surface primitives etc.

You should build that surface and for the purpose of rendering you can use primitives of line, primitives of triangle from OpenGL or quad because it should not be that just use gluts cylinder and get a cylinder or glut nurves and so on. Therefore you have to create the surface. You may create one surface and take instances of that surface in the hierarchical fashion. Start with some simpler schemes, may be take two or three patches in the side, two patches in the front and the back, one for the top of the vehicle and have all the wheels. This is as far as the modeling is concerned and a simple rendering. You should use different modes of rendering. One mode is the wireframe, you just have to draw lines or the polygons in lines.

(Refer Slide Time: 34:37)



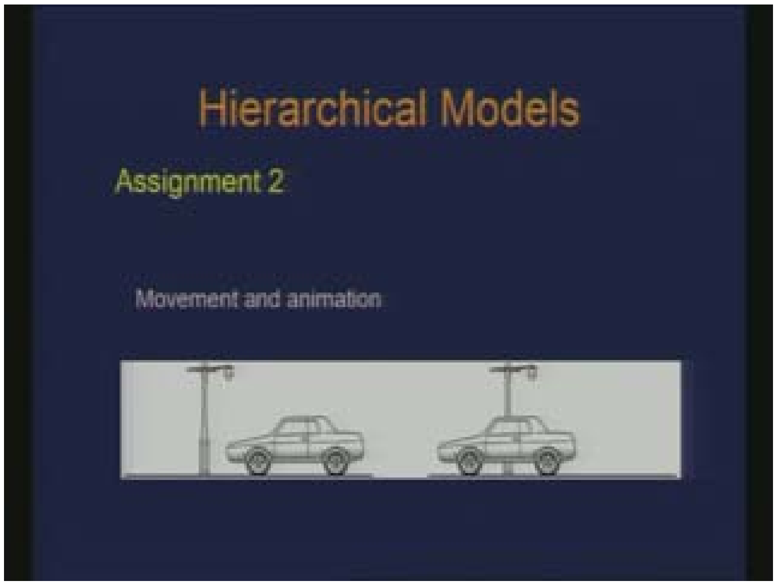
Hierarchical Models

Assignment 2

Rendering using OpenGL

The other rendering modes are using shades. Those are the models which will give you the shade of different polygons or different primitives as an interaction with the light you want to model. So there is some light source in the environment with respect to that light source you will have the elimination of the various polygons or triangles of the model. That is done very efficiently and in a straightforward manner in OpenGL. Hence for that you do not have to explicitly write your own algorithm. All you have to do is use GL shading functions. The only thing you may require to do is compute the normals for the various primitives. Eventually you may have your car looking like this. This is with respect to rendering.

(Refer Slide Time: 37:52)



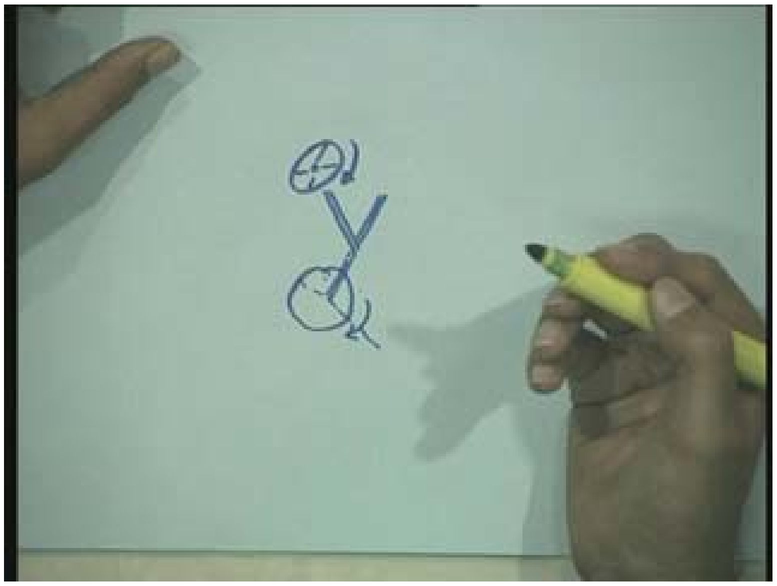
Hierarchical Models

Assignment 2

Movement and animation

So far what we have is a car which is static. Now the next stage is, we want some movement. So at some point of instance we would want the car to move so it will pass the lamp post. Here you can actually choose a very simple model. You may relate the rotational speed to the linear speed of the car. You may give some RPM to the wheel and that gets translated to the linear speed of the car. But here what we need is the turning of the wheels. It could be an explicit model or it could be an internal model you may procedurally code. Suppose you have a steer wheel that steer wheel has to connect to the front wheel. Then I rotate the steer wheel in one direction clockwise and the car wheels turn left. And when I rotate the steer wheel in the other direction the car wheels turn in the other direction. I rotate steer wheel in this direction so this would also turn in this direction. Let us not worry about differential gears.

(Refer Slide Time: 40:29)



You have a mechanism of mapping this rotation to the rotation of this and some way you then turn the car. That is up to you how you want to turn the entire car without necessarily modeling the differential. There is a notion of speed which you give. Either you give the rotatory motion of the wheels and then give the linear speed of the car or you give the speed of the car and find out the rotations of the wheels. Let that be specified by the user. You have a model inside which is a procedural model but how does it sound? You can always construct some kind of a road there so your car is actually going this way or you can specify a trajectory of a road in this manner.

(Refer Slide Time: 43:46)