Animation

CSE606: Computer Graphics

T K Srikanth, IIIT Bangalore

April 2025

Animation

CS 606 Computer Graphics

Term 2, 2022-23

T K Srikanth

IIIT Bangalore

Agenda

Overview

- Animation
- Kinematics
- Keyframe Animation
- Procedural Animation
- Relative and Constrained Motion
- Hierarchical Models
- Scene Graphs

Animation: Evolution

Steamboat Willie – Disney (1928)

Computer Ballet (1965), full-video

Geri's Game – Pixar (1997)

Phase-Functioned Neural Network for Character Control, Holden et al, ACM Transactions on Graphics (2017)

Real-time character control using diffusion model (SIGRAPH 2024)

Beyond Rigid bodies:

Meshless Deformations Based on Shape Matching (SIGGRAPH 2005)

Particles and fluids

What part of Geri’s game is computer generated?

1. Geri and the furniture
2. … and the chess pieces
3. Everything except the background
4. The entire scene and animation

Animation

- Animating a model involves computing the geometry, position and orientation of each component of the model at regular time intervals and rendering at these positions
- Re-rendering the modified scene a sufficient number of times a second (30 or greater) gives the illusion of motion
    - Real time rendering: images generated at the target frame rate – e.g. games, simulations, Virtual Reality
    - Off-line – images are computed individually and then played back at the required frame rate – e.g. movies
- Our initial focus is real-time rendering of groups of rigid bodies (could be related or independent)

Animation: Rigid Bodies

- Animating a model involves computing the position and orientation of each component of the scene at regular time intervals and rendering at these positions
- For each time step (typically, the frame rate or better)
    - Compute the transforms for each component of the model
    - Render the component with these transforms
- Thus, we need a method for computing the positions (absolute or relative) of each element of the scene at each point in time.
- Note: changes to position or orientation of the camera or lights will also require re-rendering of the scene. This can be combined into the animation process

Rendering Frames

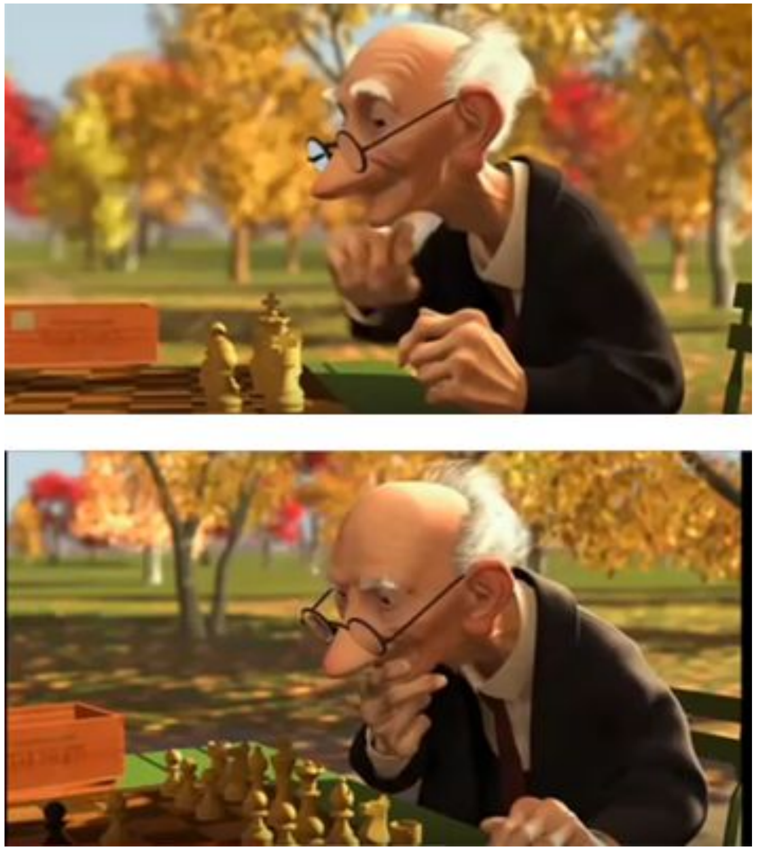
Geri’s game
Pixar

For each frame, compute the position of each object in the scene, lights and the camera

Animation: Constrained motion

Often, constraints on objects to move relative to other objects.

For example, Geri's glasses should be fixed with respect to the head, and hence should move consistently with the motion of the head. Similarly for clothes, hair, etc.



Realistic Animation*

Apart from the visual quality of scene rendering, realism requires animation to be physics-based. Need to take into account:

- physics - principles of dynamics
- collision detection and response
- “naturalness” of gait, posture etc for humans/animals

Dynamics, collision can be built into the computations to a large extent

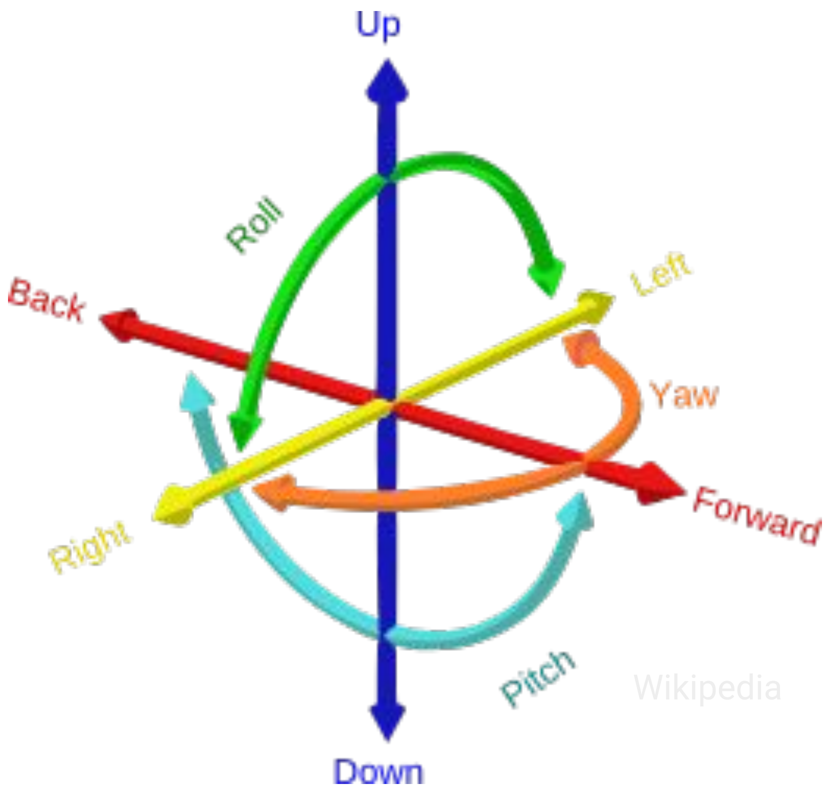
Gait, posture also require inputs from the real world

- motion capture
- AI and learning-based

* - in a subsequent course

Kinematics

- Computation of world coordinates of the parts of a moving object, as a function of time, by deriving values for the degrees of freedom (DOF) of the object – the state space of the object.
- DOF typically comprises translation and rotations.
- For an unconstrained simple rigid object in 3D, we have 6 DOF’s: 3 translations and 3 rotations
- Animation: computing consistent values in the state space of the object for successive time steps, computing corresponding coordinates of the points on the object



Constrained Motion

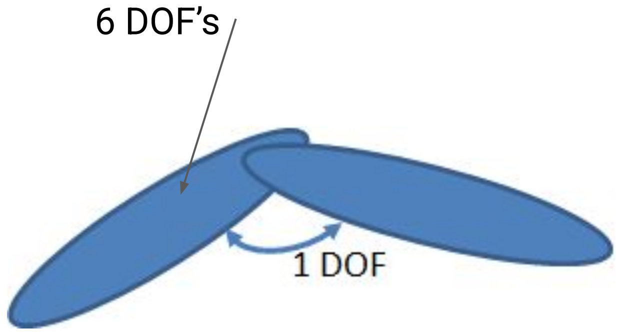
When there are multiple bodies (including obstacles) we have constraints on the positions of certain objects

- E.g. two spheres can touch each other or be apart

Constraints can be modelled as equations of position and time, and added to the kinematic equations.

Constraints help reduce the number of degrees of freedom of a set of objects

- E.g. two independent objects have a total of 12 DOF's, but if they are hinged, we have 7 DOF's



Keyframe Animation

In traditional animation, animator positions the object at a set of time values

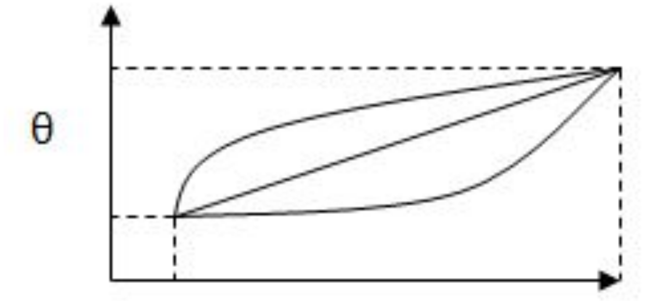
- called the **key-frames**
- Represent what the scene would look like at certain snapshots
- Can include not only position, but also colours, lighting states, etc

Intermediate frames are filled in with intermediate positions for the objects

- Called **in-betweening**
- Use of appropriate smoothing or interpolation techniques

Keyframe Animation

- Compute/specify positions for specific time values (key-frames)
- Interpolate positions in between using appropriate rules
- For articulated objects:
  - Compute/specify joint values at key-frames
  - Compute joint angles in between by smooth interpolation of joint values
- Velocity during interpolation can be varied to produce different motion effects
- Splines often used both for interpolation



3 different interpolations of a joint angle between key-frame points

Procedural Animation

- Numerical computation of stages of motion
- Incorporate laws of physics as needed
    - Helps achieve perception of realism. But can get very complex and computationally expensive
- Intentionally add “non real” effects if needed, especially in cartoons/games

- Typically, identify (solve for) specific positions, and interpolate position and angle parameters in between
    - Similar to key-framing and in-betweening

Relative Motion

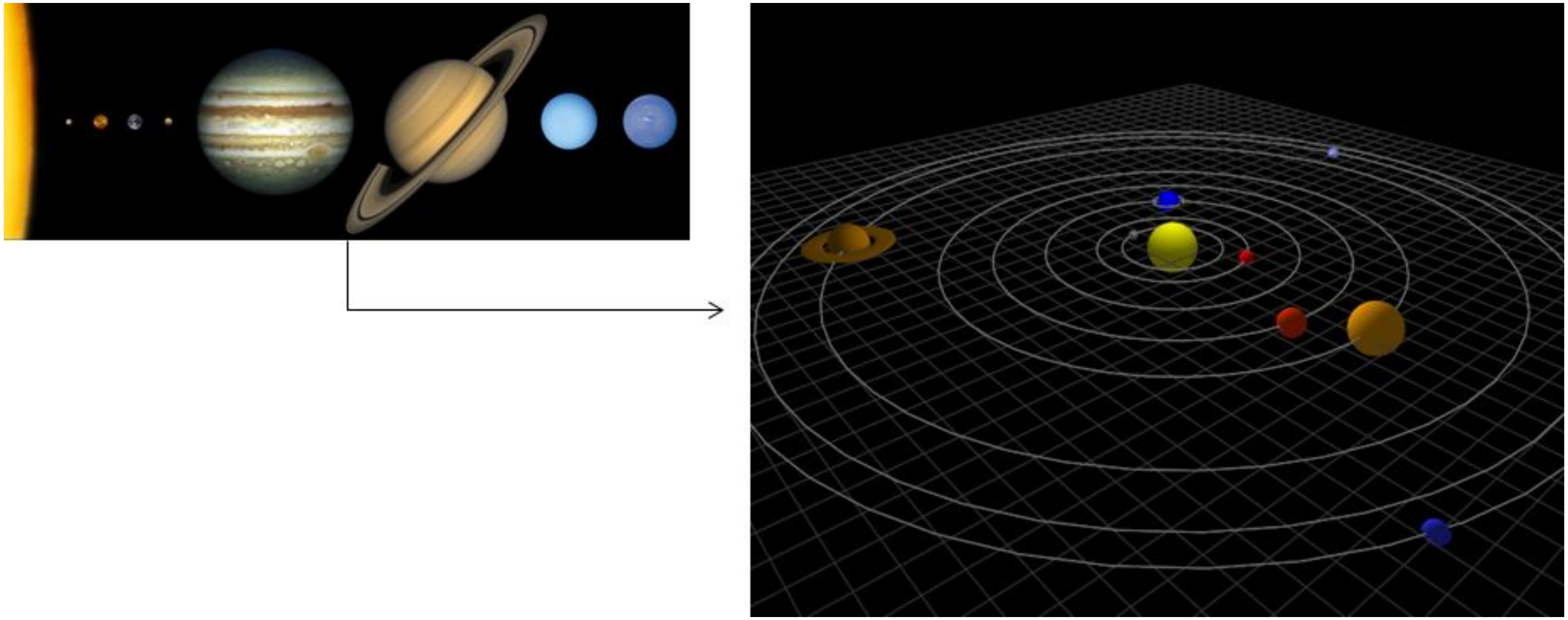


Image: Tamar Shinar, cs.ucr.edu

Object-centric Hierarchies

Often, need to structure objects in a scene based on relationships between them

- Position
- Material properties

Consider a scene comprising standard primitives (cube, cylinder, sphere etc.), scaled and positioned to create more complex objects

Thus, each object can be seen as a primitive (defined in its model frame) that is transformed to the world or scene frame.

Primitive ➔ [scale → rotate → translate] ➔ Object in scene

In a scene,

- objects could share parts of their transforms
- Or, the position of an object may be defined relative to the position of another object

Constrained Motion

A (simplified) car, for example, can be modeled as a chassis and four wheels.

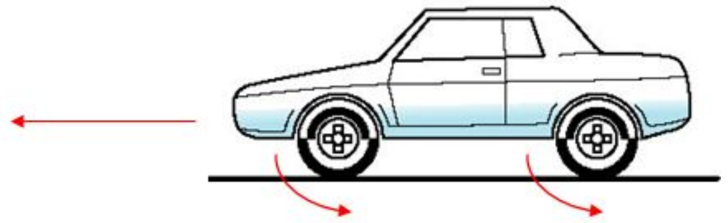
- Each wheel can be seen to have a (fixed) relative position in the reference frame of the chassis

When the car moves forward:

- The chassis moves forward
- Each wheel moves forward by the same amount, but also rotates about its axis
- The amount of rotation and the forward movement are related

Hence, to render the car after it has moved a distance x,

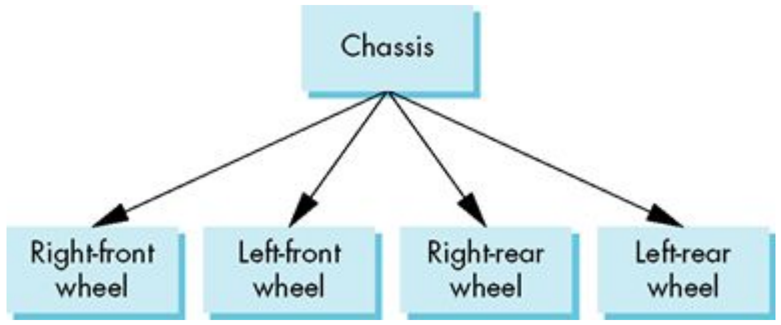
- Render the chassis with an additional translation of x
- Render the wheels
    - with the new translation of the chassis,
    - and composed with an additional rotation



E. Angel and D. Shreiner: Interactive Computer Graphics 6E

Hierarchical Models

- Model the car object as a tree, where each node inherits information from its parent, and stores its relative transform (with respect to the parent)
    - In this case, the translation of the car is stored in the Chassis node
- Each wheel stores its translation relative to the chassis as well as its rotation information
- To render, traverse the tree, composing the transforms from the root to the current node



E. Angel and D. Shreiner: Interactive Computer Graphics 6E

Kinematics of Compound Objects

Specific to a model, this involves deriving the motion of parts of the model based on a few configuration parameters, which in turn are a function of time.

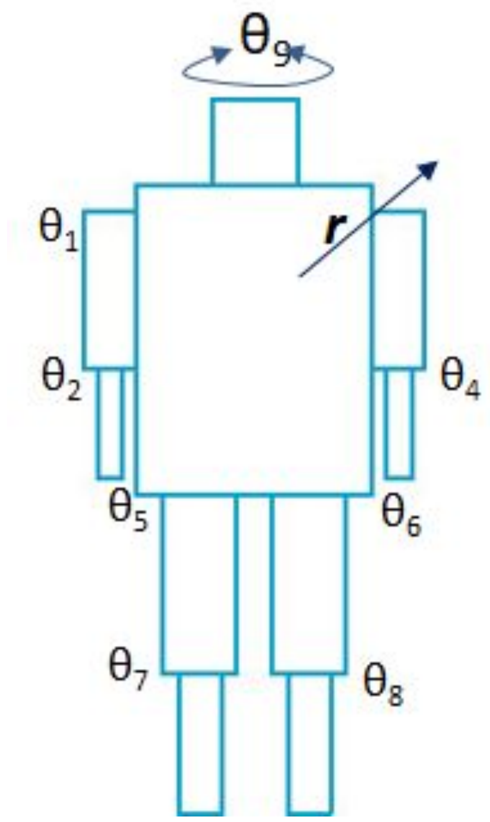
As an example, for a humanoid model, we need a function f, such that, for any time t of interest,

$$\boldsymbol{f}: t \rightarrow (\boldsymbol{r}, \boldsymbol{\theta})$$

where $\boldsymbol{r}$ is the position vector of the torso, and $\boldsymbol{\theta}$ is the vector of the values of the joint angles:

8 joints + head rotation + DOF's of the torso = total DOF of 15

We use these values to compute the transforms at each node, and render the scene.

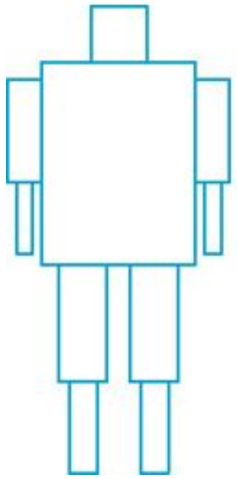


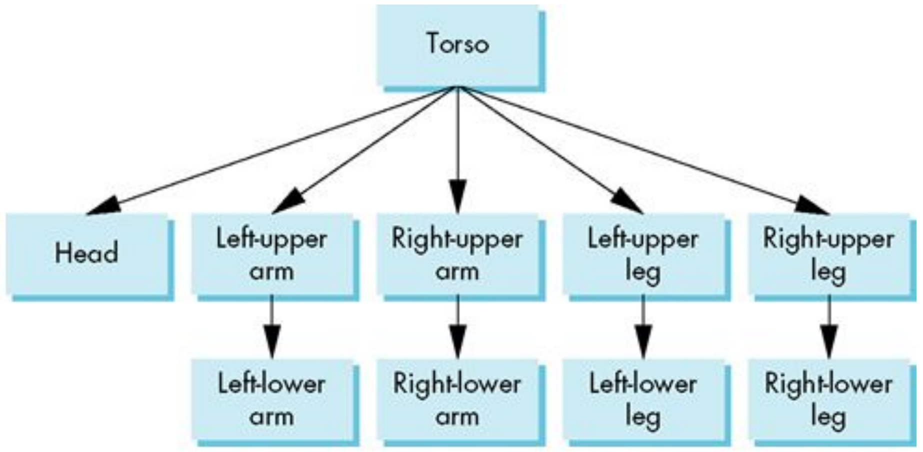
Hierarchical Model: Humanoid

Group objects into sub-hierarchies based on their positional dependencies

- E.g. the left lower arm moves when the left upper arm is moved, and it can have a further rotation about the elbow joint

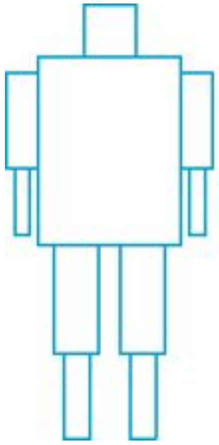
The configuration of the assembly can be defined by the position of the torso and rotations at each of the joints (1 position vector and 8 joint angles + head rotation)

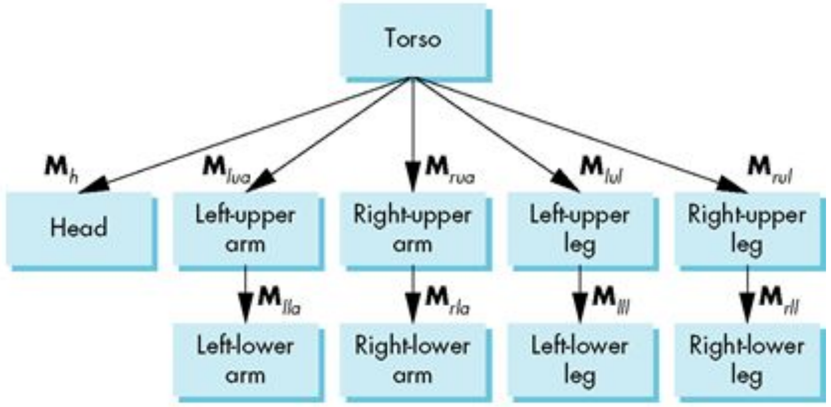




Managing Relative Transforms

- Group objects into sub-hierarchies based on their positional dependencies
  - E.g. the left lower arm moves when the left upper arm is moved, and it can have a further rotation about the elbow joint
- Matrices specify position of a node relative to its parent
- To render the object, traverse the tree, composing the transforms along the path from the torso to the current node, apply this transform to the current node, and then render the node

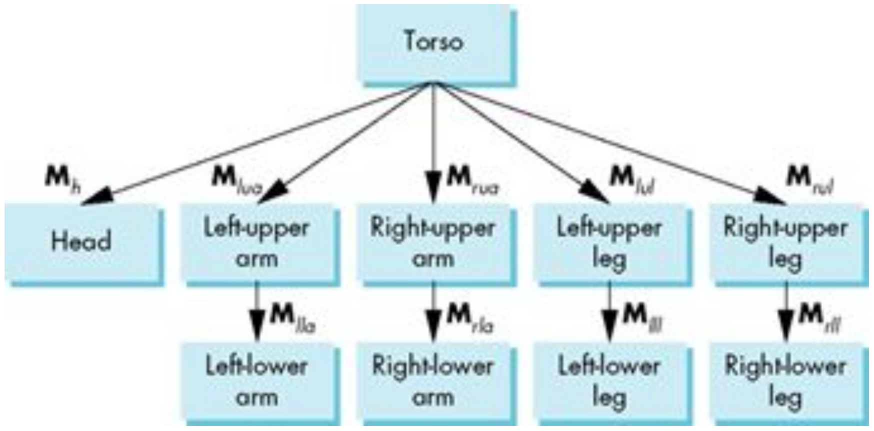




Rendering - Tree Traversal

Assuming each object has a render() method that takes a transform as argument, the traversal code:

```
{
    torso.render(M);
    head.render(M M_h);
    leftUpperArm.render(M M_lua);
    leftLowerArm.render(M M_lua M_lla);
    rightUpperArm.render(M M_rua);
    rightLowerArm.render(M M_rua M_rla;
    …
}
```



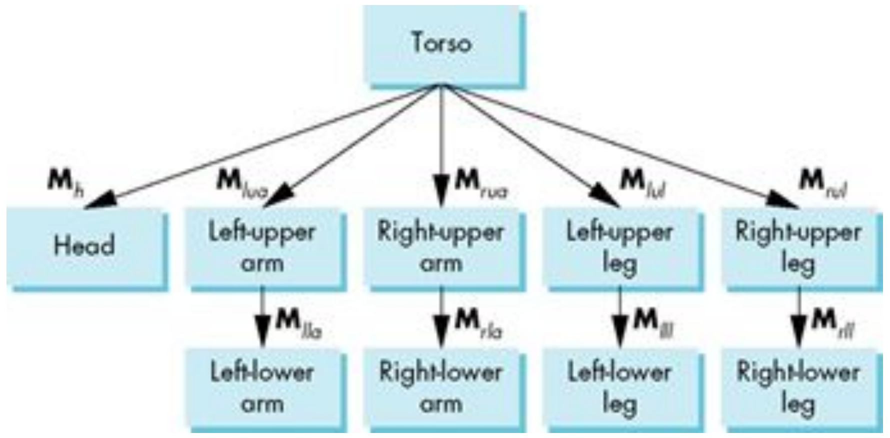
Rendering Hierarchical Models

Can be made more efficient by maintaining a matrix stack that corresponds to the current state of the traversal

To render a node, send the current matrix to the display system, followed by the data for the node

```
node::render() {
  // compose node's transform  with current matrix
SetTransform(currentTransform*nodeTransfo
rm);
  //  Display data for node
   …
 for each child {
     PushMatrix(); //  save current matrix state
     child.render();
     PopMatrix();  //  restore state of matrix
 }
}
```

Assume pushMatrix() saves the current matrix to a stack and popMatrix() pops the last element of the stack



Graph of Scene Objects

Similar to position, other properties could also be hierarchically defined

- Material properties (color, material) : sometimes a node inherits properties of parent, and sometimes overrides them
- Can tag these to the nodes of the tree/graph and process them appropriately during traversal

In addition to the model being rendered, objects such as lights and camera also have geometric properties as well as specific properties

If we can add all these objects into a single graph, then traversing this graph allows the scene to be rendered with the correct geometry, material, lighting, camera view.

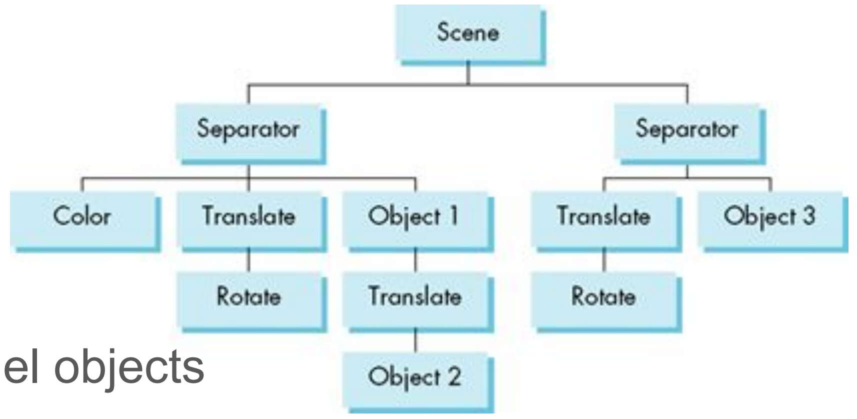
A structure such as this is needed in any case, since OpenGL does not store display information once rendered (immediate mode). Data of the scene needs to be resent to the pipeline after any change in scene – geometry, material, lighting, camera view.

Scene Graph

A graph that represents the objects of a scene and their relationships: camera, lighting and the hierarchical model that is to be rendered

Comprises the following types of nodes:

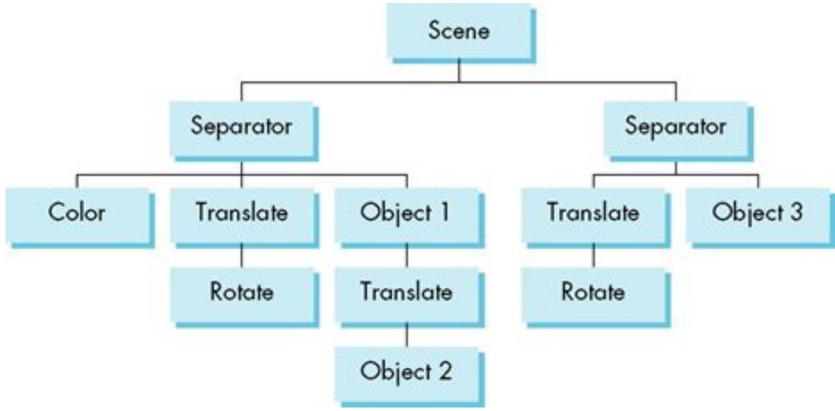
- Geometry: defining the shapes – primitives, mesh surfaces, parametric surfaces, curves etc.
- Lighting: type, position and orientation
- Camera: type, position and orientation
- Transform
- Appearance: material, color etc.
- Group/Separator: logical grouping of model objects
    - Isolate state changes
    - Equivalent to push/pop



Scene Graph

Traversal of the graph is used for multiple purposes:

- Display
- Interactive pick
- Computing bounding boxes
- Searching
- Writing to stream



Specific actions in an SG traversal are application specific and need to be designed (for a particular kind of application)

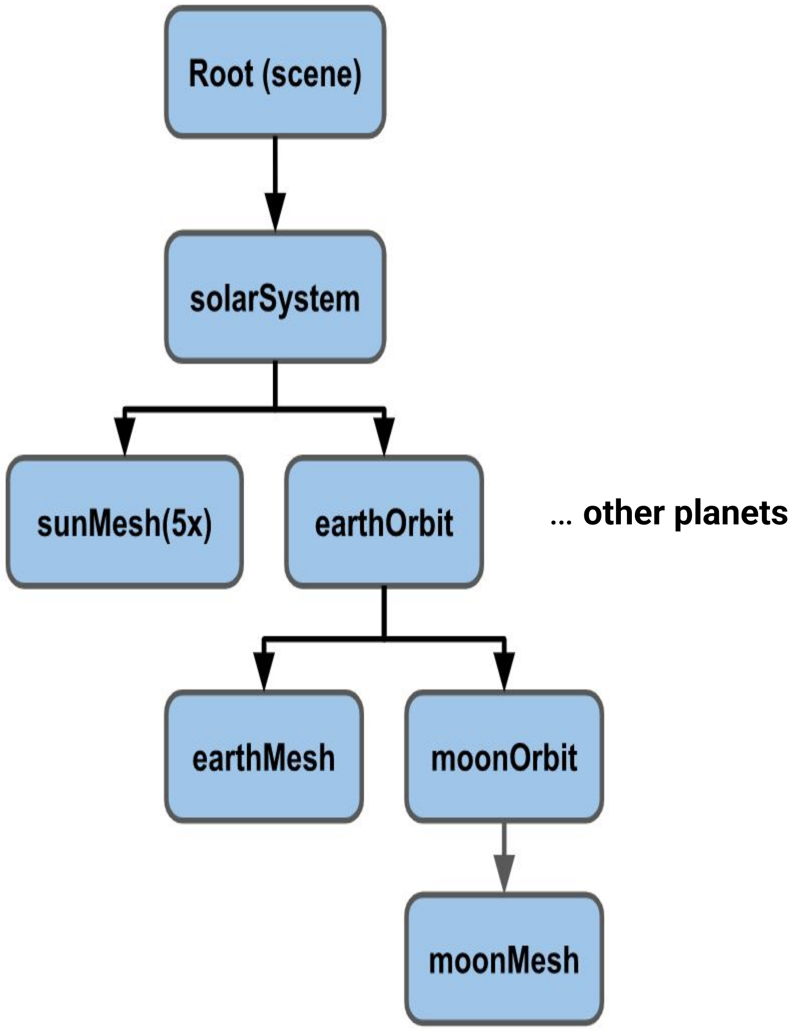
Concept of scene graph is used in most modeling, graphical editing and gaming systems. Encouraged to use this where possible

Solar System: Example

How would you create a dynamic model of the solar system

- The Earth
  - completes an elliptical orbit around the sun in 365 earth-days
  - Has its axis of rotation tilted at 23.5 degrees to the plane of orbit
  - Rotates on its axis in 1 day
  - Has a satellite – the moon
    - Revolves around the earth in ~28 days
    - Rotates once for each revolution around the earth
- Similar descriptions for each of the planets (including images of the surface appearance)

Scene Graph: Solar System



https://threejsfundamentals.org/threejs/lessons/threejs-scenegraph.html

Scene Graph: Animation

- Scene Graphs provide a convenient way to manage scenes and their animation
  - Most commonly used approach – gaming engines, simulation displays etc.
- Represent the scene as a hierarchical structure, with (relative) transforms at each node
- For each time step of the animation
  - Update the positions (and other properties) of the nodes of the graph
  - Render the nodes
- Both above involve pre-order traversal of the graph, and using a run-time stack for maintaining transforms and other properties of each sub-tree

Scene Graph: Program Flow

Controller instantiates Model objects and adds them to the Scene Graph (as needed by the application). Lights, camera added as needed.

Sets up hanlder/callback from View or other events (e.g. timer) .

Within the callback, repeatedly invokes the following:

```
{
     // update time;
    …
    // update all nodes of the sceneGraph
     sceneGraph.update(time); // Note: this can also be set up as a recursive method
     sceneGraph.render();
}
```

Note that pick/select can be handled the same way