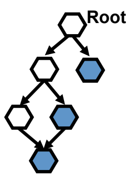
Scene Graphs

Scene Graph Overview

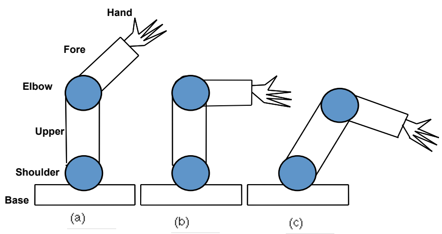
- Building Scene Structures
- Traversal
- Examples
- Instancing and Re-Use
- More Transformations

Concept of Scene Graph

- Objects placed relative to one another
- Objects made of similar components
- Directed acyclic graph
- Links are transformations
- Nodes are empty or contain geometry
- The root of the graph corresponds to "world coordinates"

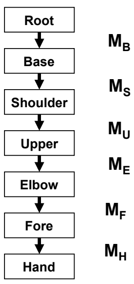


Use for Animation/Modelling



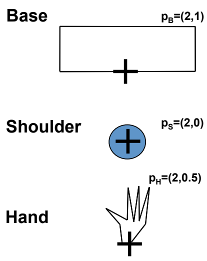
Robot as a Graph

- Each node other than root contain a piece of geometry
- Each link is a transformation matrix, $\mathbf{M}_B$, $\mathbf{M}_S$, etc.
- Main concept is that robot can be posed by changing rotation in Shoulder and Elbow



Local Coordinates

- Each part of the robot is modelled in its own local coordinate (LC) system
- Local coordinates are defined by the person modelling the system
- Choice is determined by convenience
- Common choices:
  - The centre of the object
  - The centre of the object
  - A corner of the object



World Coordinates

- Everything is eventually positioned relative to the world coordinates (WC) or room coordinates
- We know how to convert WC to viewing coordinates (VC) – it's the general camera model
- Eventually we need to convert points in an object's LC into WC

Local Transform

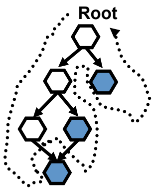
- An object's local transformation maps LC to the parent's LC
  - shoulder is translation (0 1 0) from base (MS)
  - upper arm is translation (0 3 0) from shoulder (MU)
  - elbow is translation (0 3 0) from upper arm (ME)
  - fore arm is rotation Z by 90 then translation (0 2 0) (MF)
  - Etc.
- Note that directions such as "up" depend on what transformations have been defined by ancestors in the tree

Rendering Traverse

- Must get object definitions in WC before passing to camera
- For a vertex in the base object
  - p.$\mathbf{M}_B$ is in WC
- Matrices are inherited down stack
- So for object under shoulder
  - p.$\mathbf{M}_S\mathbf{M}_B$ is in WC
  - (Note that p.$\mathbf{M}_S$ is in the local coordinates of the base!)

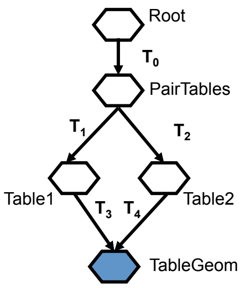
Implementation

- Generally implemented by a straightforward recursive descent
  - "push" on graph descend
  - "pop" on graph ascend
- The concatenation of all LT matrices above a node is called the current transformation matrix (CTM)



Sharing Nodes

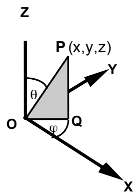
- A common "pattern" found in a scene graph is a multiply instanced geometry
- One table, many places
- Node Table1 has CTM $\mathbf{T}_1\mathbf{T}_0$
- Node Table2 has CTM $\mathbf{T}_2\mathbf{T}_0$
- $\mathbf{T}_3 = \mathbf{T}_4 = \mathbf{I}$
- So TableGeom appears in two different positions



Rotations

Spherical Coordinates

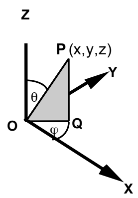
- Represent a point using two angles φ and θ, and with r = length(x,y,z)



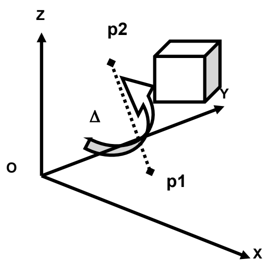
Q is projection of P onto XY plane
φ is angle between X axis and OQ
θ is angle between OP and Z axis

Spherical Coordinates

- Length OQ = r sin(θ)
- So
  - $x = r \sin(\theta)\cos(\varphi)$
  - $y = r \sin(\theta)\sin(\varphi)$
  - $z = r \cos(\theta)$
- The other way:
  - $r = \mathrm{sqrt}(x^2+y^2+z^2)$
  - $\tan(\varphi) = y/x$
  - $\cos(\theta) = z/r$



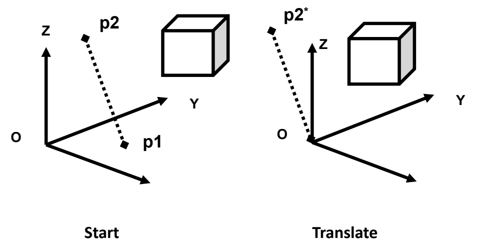
Rotation About an Arbitrary Axis



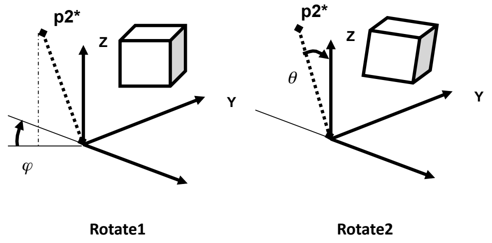
…

1. Translate p1 so it is at the origin:
   O = p1* = p1.**T**, where **T** is translation by –p1)
2. Let p2* = p2.**T** (new position of p2)
   find spherical co-ordinate of p2* (r,θ,φ)
3. Rotate about Z by - φ to bring p2* into ZX plane
4. Rotate about Y by - θ to bring p2* onto Z axis
5. Now rotate about Z by Δ
6. Invert steps 4-1

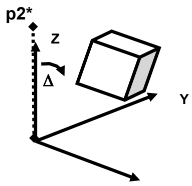
…



…



…



After Steps 1-4

- Now we apply the transformation (rotation by Δ degrees) we are after
- Invert steps 4-1

Rotation Matrices

- Reminder: rotation around X, Y, Z axis

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Nice exercise:
derive a rotation matrix!

Scene Graph Recap

- Use of **scene graph** to model environments
- Notion of render traversal and the **current transformation matrix**
- Instancing and sharing of nodes
- Rotation of objects around an arbitrary axis