

Handouts: Smooth Shading

Retained Mode Graphics



Retained Mode Packages



- Maintain graphical data structure
- Traverse graphical data structure
 - Auto redisplay, picking, intelligent handling of special nodes, optimizations

- Goal: Make your life easier!
 - You should be thinking: this is what I would just be doing anyway

Handouts: Smooth Shading

Consider OpenGL Display Lists



- Provide some of this
 - Contain geometry, attributes
 - Hierarchy (DAG)
 - Seem similar to SPHIGS structures
- Problems?
 - "Output only", expensive to create/change
 - Not real hierarchy
 - C macros vs. procedures

SPHIGS structures



- Similar to OpenGL DLs
 - "recorded actions"
- Better hierarchy support
 - Editing
 - Nesting
 - Traversal: picking, special handling

Handouts: Smooth Shading

Problems with DL model



- Attributes, geometry mixed together
 - Programmer must know where everything is
 - Less opportunity for optimization
 - Cannot integrate application data

Object oriented model



- Inventor, Java-3D, Repo-3D
- Objects for each "object"
 - Internal DAG nodes for structure
 - Leaf nodes are "things"
 - Properties attached to node

Handouts: Smooth Shading

Major Objects



- Geometry
 - Spheres, boxes, polygons, cameras, lights, ...
- Properties
 - Transformations
 - Attributes (rendering style, color, etc.)
 - Geometric Properties
- Groups
 - Structure

Special Group: "Root"



- Attach a hierarchy to a window
- Need
 - Camera
 - Window
 - Viewport

Handouts: Smooth Shading

Properties

- Every property has a default value
- Attached to nodes
 - Define property value for graph rooted at current node

Advantage to Objects: Indirection of method calls

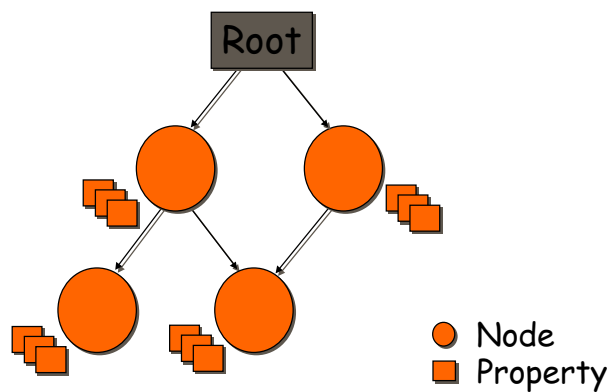
- Property value?
 - *type* get()
- Incorporate time!
 - *type* get(int time)

Handouts: Smooth Shading

Variations of properties (Obliq-3D)

- For *every* type of property (color, transformation, float, int, ...)
 - Constant
 - Synchronous time-based animation
 - Arbitrary function

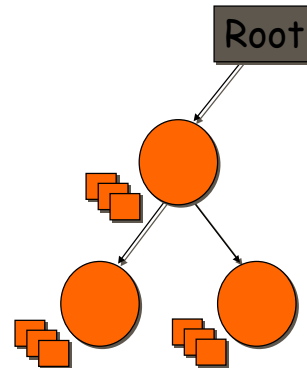
Example Graph



Handouts: Smooth Shading

Graph Traversal: Rendering

- Save/restore state
- Traverse multiple times
- Decide when to render



Graph Traversal: Picking

- How to incorporate OpenGL Picking?

Handouts: Smooth Shading

Optimizations: Elision

- Pruning
- Culling
- LOD

Optimization: Rendering

- State changes are expensive
- Display lists are expensive to create by faster to draw