

Handouts: Smooth Shading

Retained Mode Graphics



Two approaches to graphics



- Say "how"
 - Immediate mode: explicit commands

- Say "what"
 - Retained mode: define and change model

Handouts: Smooth Shading

What's in a Geometric Model



- Geometry
 - Spatial layout
 - Attributes
- Topology
 - Connectivity
 - Structure
- Application-specific data

Model represents graphical part of application data

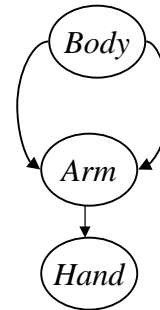


- App + graphical data structures
 - App ds changes => graphical ds changes
- Modern libs let you integrate them
 - Inventor, Java-3D

Handouts: Smooth Shading

Hierarchy in Geometric Models

- Build models bottom-up
 - Define Objects in own coordinates
 - Relate them to logical parents
 - Modularity
- Define a DAG
 - Include objects more than once



Hierarchy Inheritance

- Transforms
 - Nested Coordinate systems
- Attributes
 - "red car"

Handouts: Smooth Shading

Recall Blobby Man



- You had a (simple) data structure

- Multiple uses
 - Rendering
 - Picking

Retained Mode Packages



- Maintain graphical data structure

- Traverse graphical data structure
 - Automatic screen regeneration
 - Picking
 - Intelligent handling of special nodes
 - Lights, transparency
 - Optimizations

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

Object oriented model



- Inventor, Java-3D, Repo-3D

- Objects for each "object"
 - Internal DAG nodes for structure
 - Leaf nodes are "things"
 - Attributes attached to node