Simulation in Computer Graphics

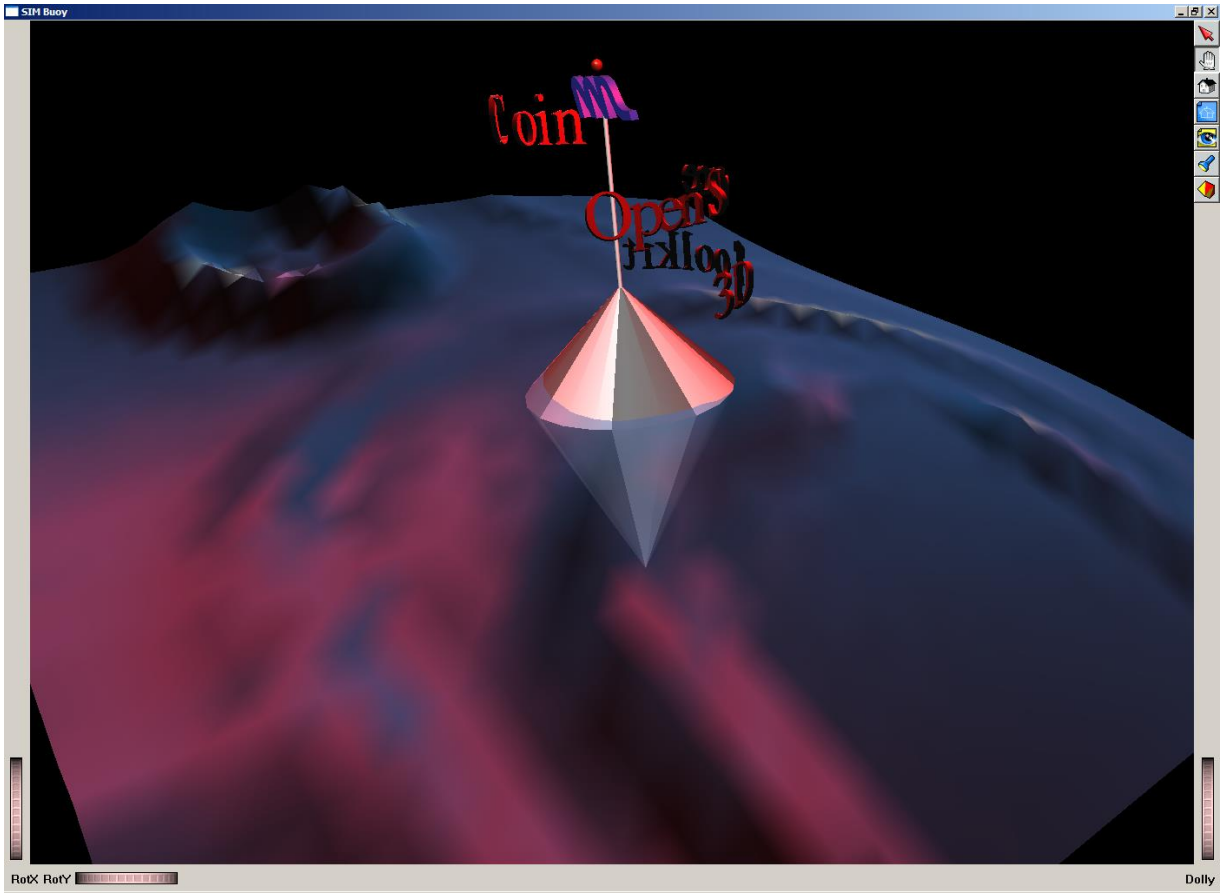
SGI's Open Inventor

Matthias Teschner

Computer Science Department
University of Freiburg

UNI FREIBURG

Motivation



Introduction

- Open Inventor is a file format for 3D objects and scenes

```
#Inventor V2.0 ascii
Separator {
    Transform {
        translation 0 0 1 }
    Sphere {
        radius 1 }
}
```

- Open Inventor is an object-oriented C/C++ library for graphics programming

```
SoTransform *myTransform = new SoTransform;
myTransform->translation.setValue(0,0,1);
SoSphere *mySphere = new SoSphere;
mySphere->radius.setValue(1);
```

Characteristics

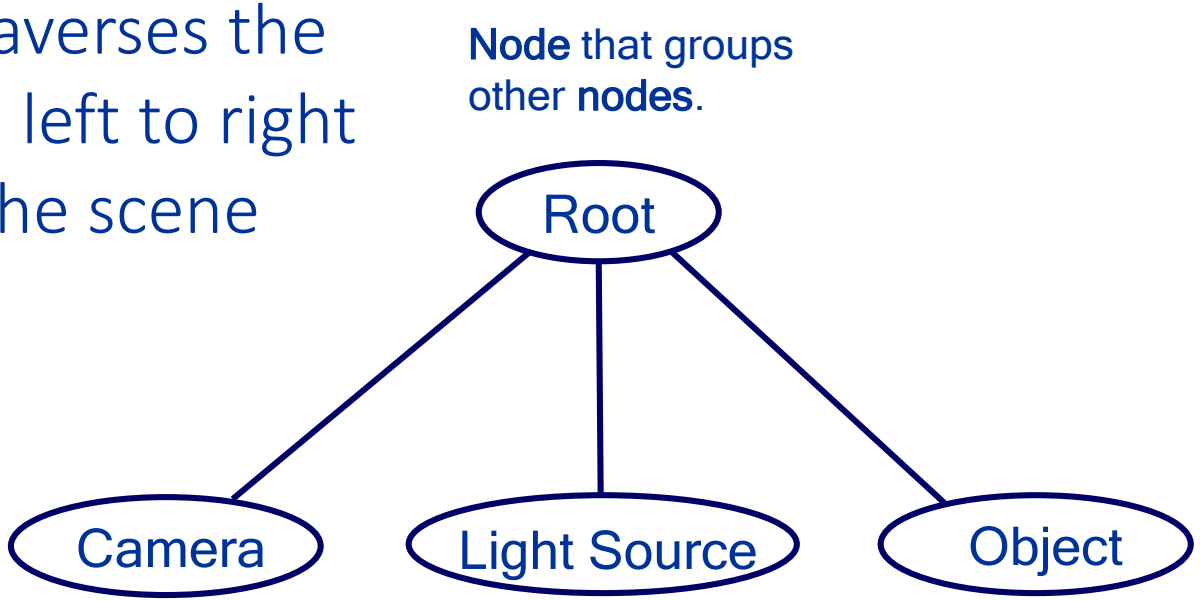
- main data structure is a graph which describes a scene
  - graph consists of nodes and is traversed to display the scene
- object-oriented instead of drawing-oriented
  - camera, light, object shape, material, texture
  - viewers, sensors, manipulators, animation
- built on top of OpenGL
  - OpenGL is used when Inventor's render action is invoked
- open source
  - platform and window system independent
  - SGI Irix, Linux, Microsoft Windows, Mac OS

Outline

- example and viewer
- nodes and scene graphs
- cameras and lights
- illumination models and shading
- geometries
- events and sensors
- combination with simulations

The First Scene Graph

- Inventor traverses the graph from left to right to render the scene

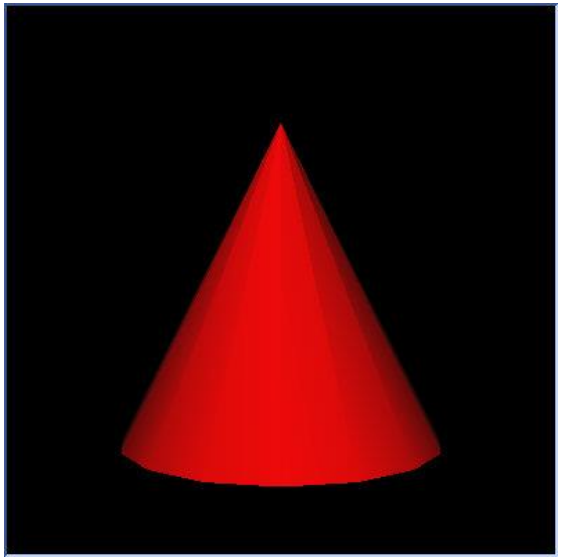


First Nodes

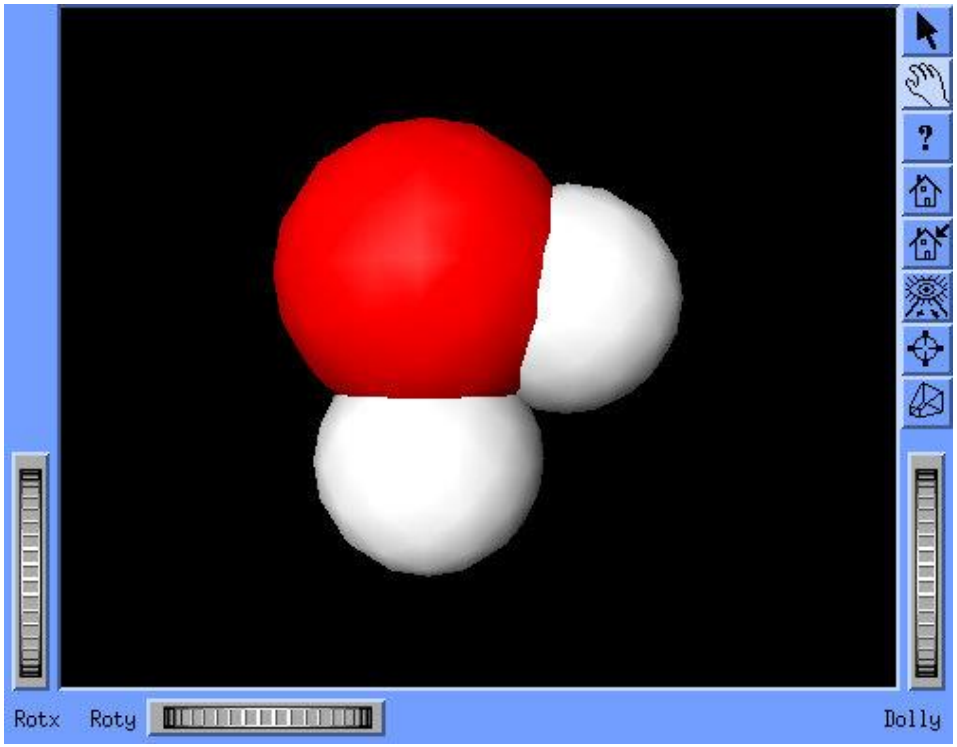
- SoPerspectiveCamera
  - parameters: position, orientation, aspectRatio, nearDistance, farDistance, focalDistance
  - methods: point at, view all
- SoDirectionalLight
  - parameter: on/off, intensity, color, direction
- SoSeparator
  - group node
  - saves traversal state before traversing its children
  - restores traversal state after traversing its children

C/C++ Example

```
SoSeparator          *root     = new SoSeparator;
SoPerspectiveCamera  *myCamera = new SoPerspectiveCamera;
SoDirectionalLight   *myLight  = new SoDirectionalLight;
SoCone               *myCone   = new SoCone;

root->addChild (myCamera);
root->addChild (myLight);
root->addChild (myCone);

myCamera->viewAll (root);

SoXtRenderArea *myRenderArea =
new SoXtRenderArea;

myRenderArea->setSceneGraph (root);
myRenderArea->show ();
```



Scene Viewer



selection mode
viewing mode
help
reset camera to home
define current camera as home
set camera to view all
define point to zoom in
orthographic/perspective camera

zoom

rotation

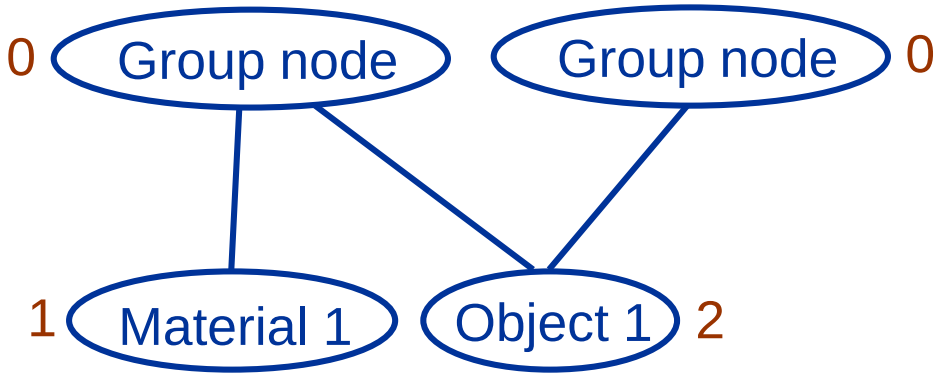
- left mouse: rotation; middle mouse: translation; left and middle: zoom; right mouse: rendering mode

Node Types

- shape nodes (geometry) — SoCone, SoCube, SoCylinder, SoNurbsSurface, SoSphere, SoText3
- appearance nodes (shading) — SoBaseColor, SoMaterial, SoFont, SoDrawStyle
- transform nodes — SoTranslation, SoRotation, SoScale, SoRotationXYZ, SoMatrixTransform, SoResetTransformation
- group nodes — SoSeparator, SoSwitch

Node Reference Counter

- number of references to a node (parent-child links)



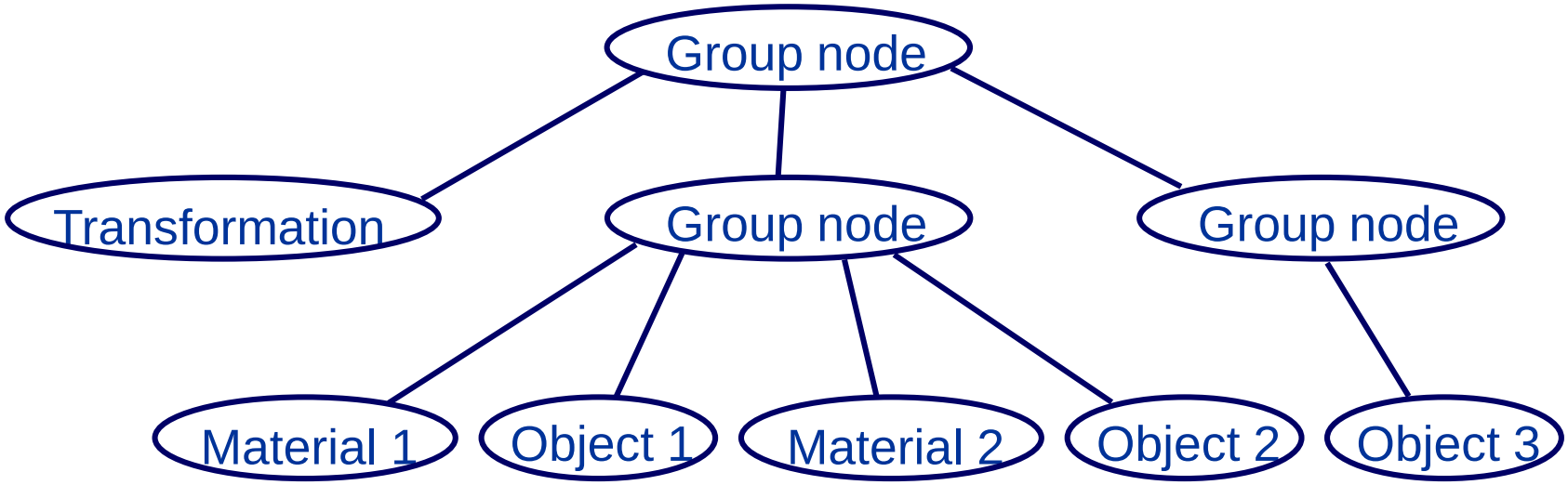
- adding a node as a child to a parent node increments the reference counter of the child node
- removing a child node from a parent node decrements the reference counter of the child node
- the reference counter can be manually changed with ref() and unref()

Node Deletion

- when a node's reference counter decreases from 1 to 0, the node is deleted by Inventor
- adding a node to a graph: 0 -> 1
- removing it from the graph: 1 -> 0 -> deletion
- simple, but:
  - removing a node from a graph that you want to keep
  - deleting a node with reference counter 0
  - actions applied to a node increase the reference counter and decrease it afterwards
- to solve or avoid these problems the reference counter can be adjusted with ref() and unref()

Groups and Ordering

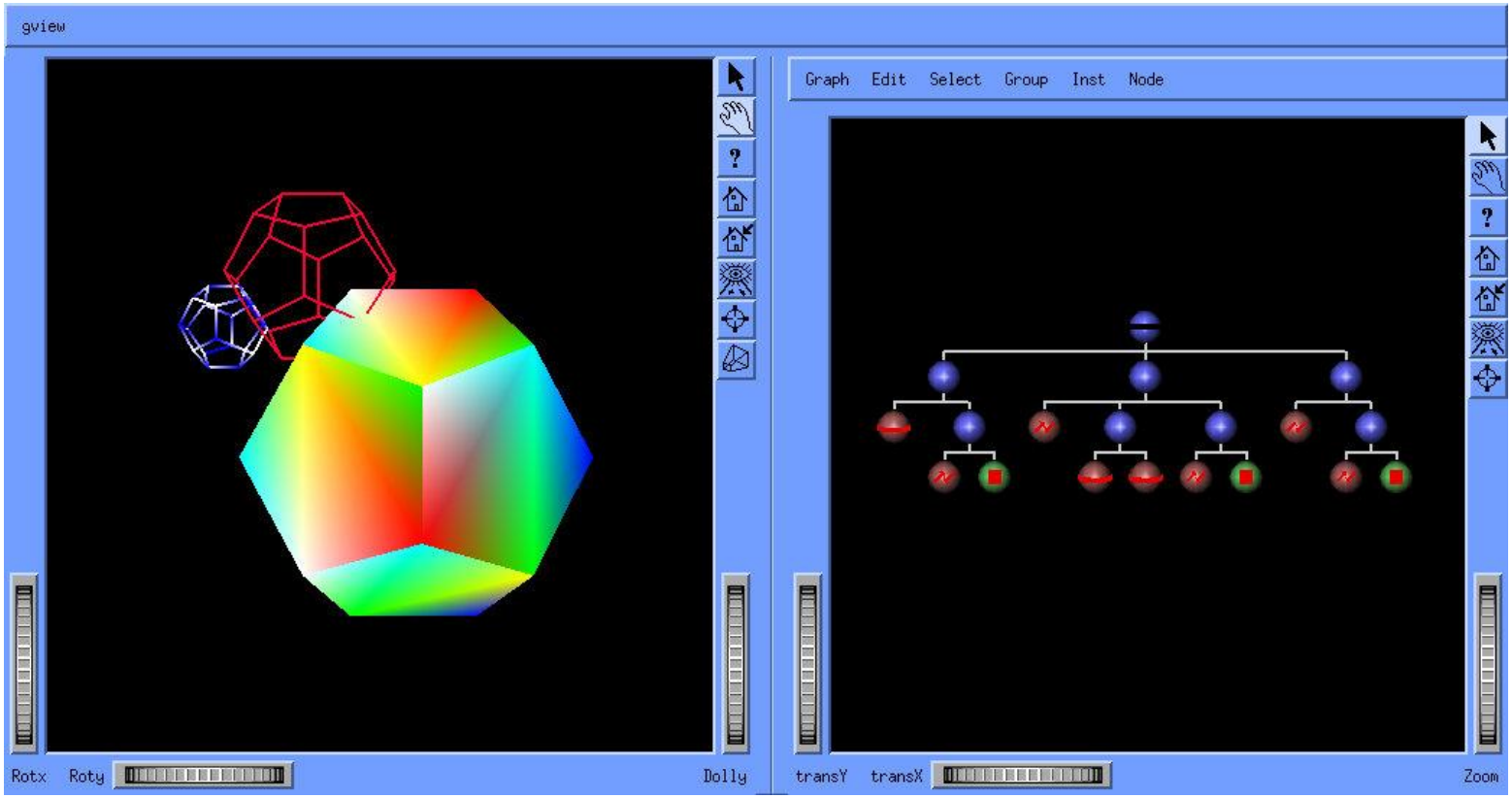
- group nodes save and restore the traversal state



- transformation is applied to object 1, 2, 3
- material 1 is applied to o. 1, material 2 is applied to o. 2
- neither material 1 nor material 2 is applied to object 3

Example

- scene and corresponding scene graph



Scene Graph Summary

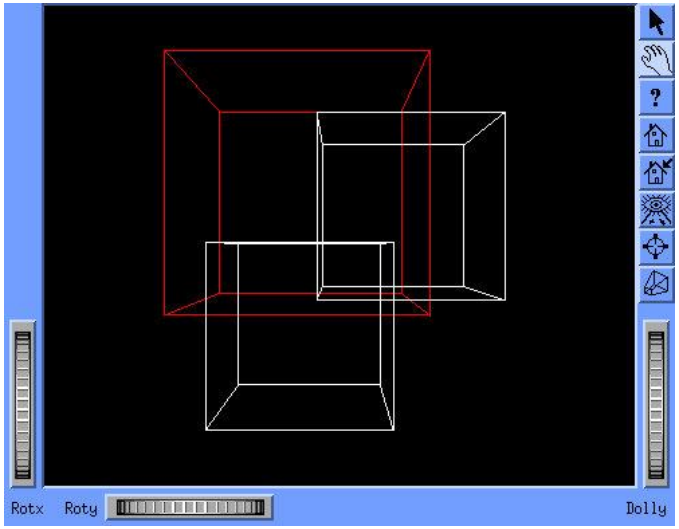
- object-oriented description of a scene
- various types of nodes (shapes, properties, groups)
- reference counter for nodes
- graph traversal for rendering
- groups encapsulate the rendering environment
- scene graph can be manipulated by
  - editing an Open Inventor file (*.iv)
  - using the C/C++ library
  - using the scene graph viewer (gview)

Rendering

- cameras
- lights
- illumination models
- shading

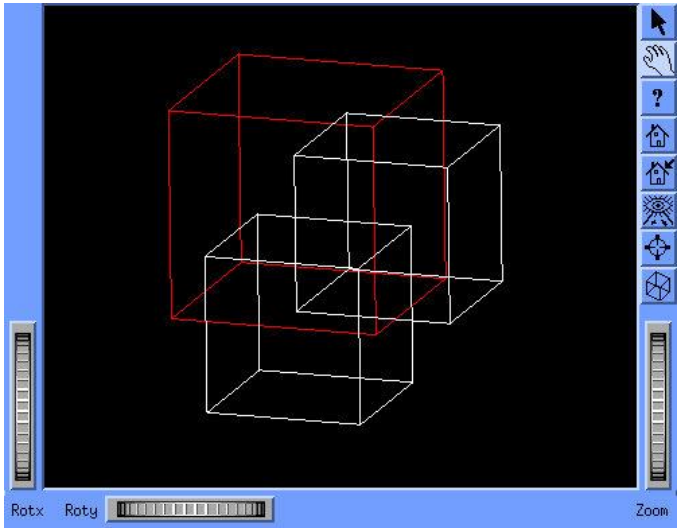
Cameras

- SoPerspectiveCamera



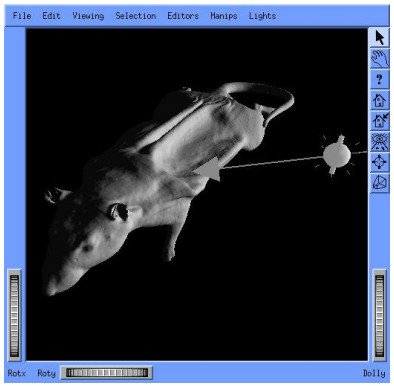
- parameters:
  position, orientation,
  focalDistance, aspectRatio,
  nearDistance, farDistance

SoOrthographicCamera

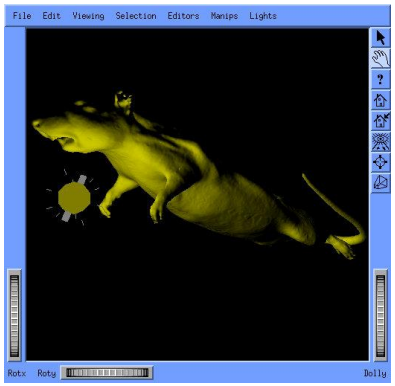


methods:
point at, view all

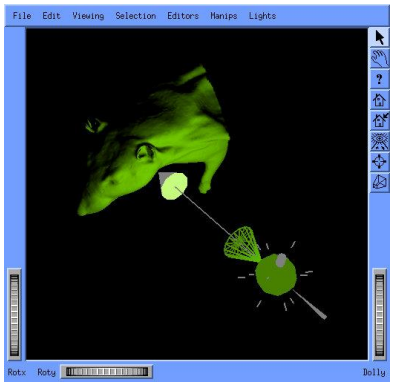
Lights



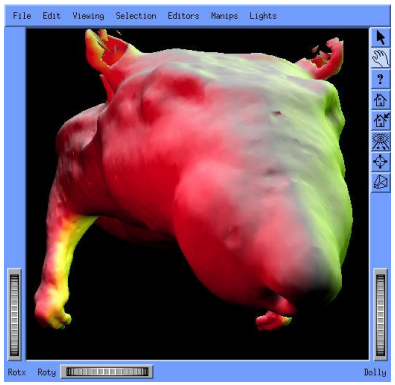
SoDirectionalLight



SoPointLight



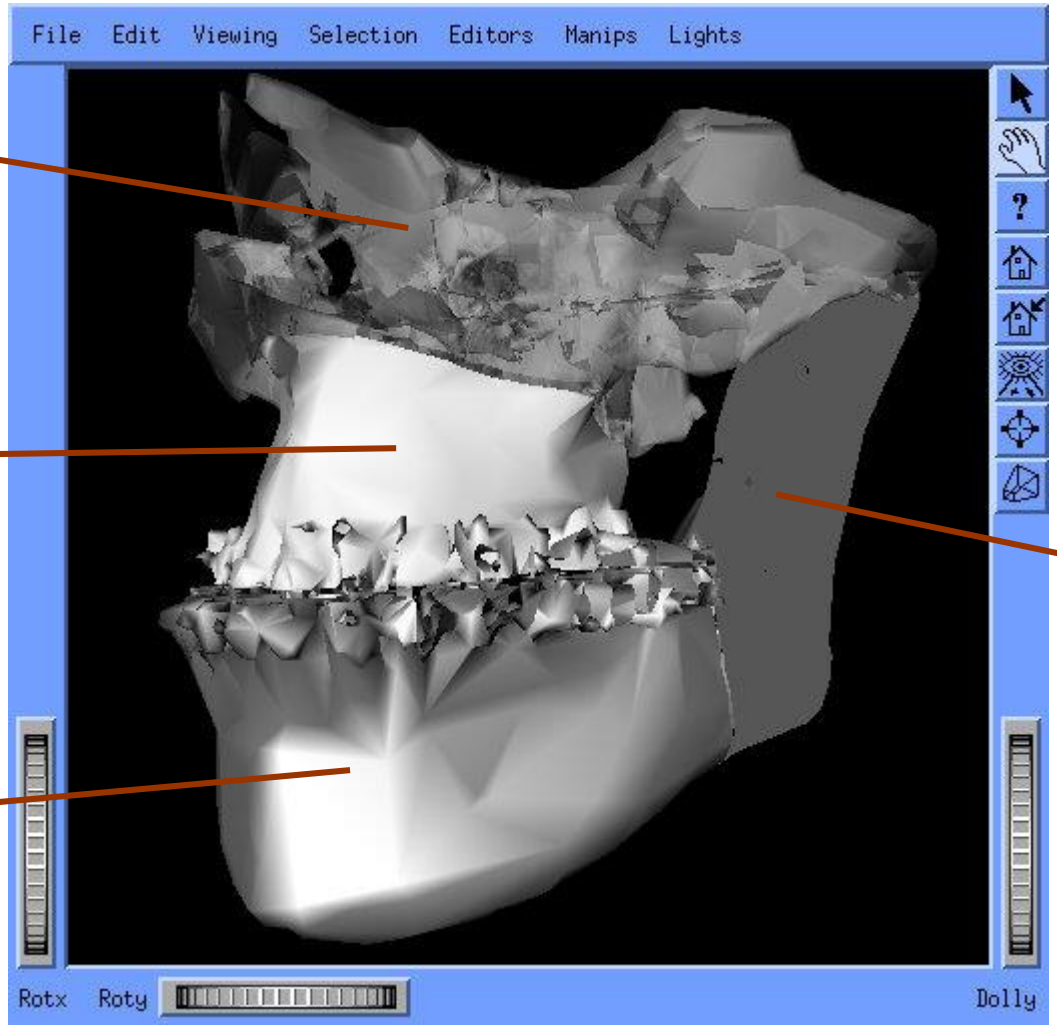
SoSpotLight



Mixed lighting

- courtesy of Stanford/NASA National Biocomputation Center, Palo Alto

Illumination Models



transparent

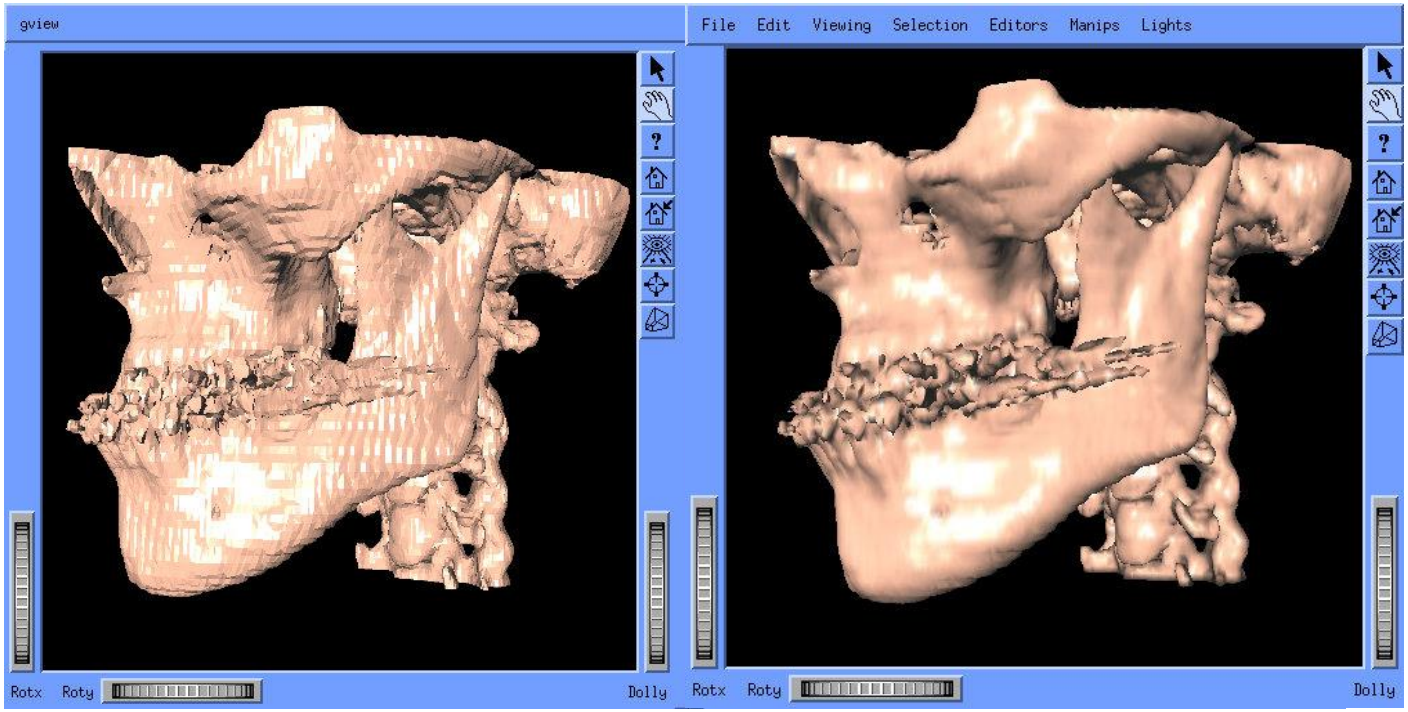
matte/diffuse

specular

ambient

Shading

- flat, Gouraud



flat Gouraud

Cameras and Light - Summary

- camera
  - orthographic
  - perspective
- light source
  - directional light source
  - point light source
  - spot light source
- material
  - ambient
  - diffuse
  - specular
- shading
  - flat
  - Gouraud

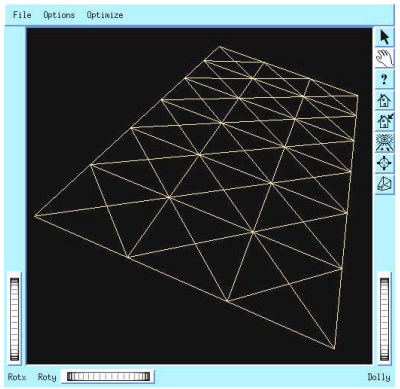
Geometries

- simple shapes
  - cube
  - cone
  - sphere
  - cylinder
- complex shapes
  - point set
  - line set
  - face set

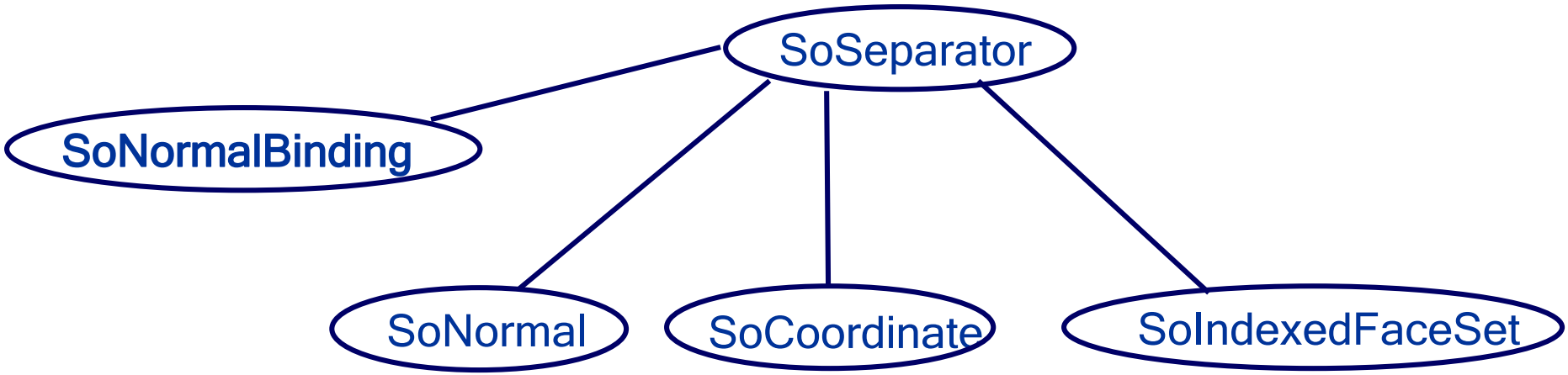
Face Set

- group (**separator**) consisting of two nodes (**coordinate3**, **indexedFaceSet**)
- **coordinate3** has a field *point* with 3D vertices
- **indexedFaceSet** has a field *coordIndex* that contains indices of vertices which form faces, separated by -1

```
#Inventor V2.1 ascii
Separator {
   Coordinate3 {
      point   [ 0 0 0,
                1 0 0,
                2 1 -1,
              3 5 3 ]
   }
   IndexedFaceSet {
      coordIndex [ 0, 1, 2, -1,
                                0, 1, 3, -1,
                   1, 2, 3, -1]
   }
}
```



Face Set
Normals and Bindings



- specification how normals are bound to vertices or faces
- also: SoMaterialBinding, SoTextureCoordinateBinding

Geometry - Summary

- point set
- line set
- face set

- cube
- cone
- sphere
- cylinder

Scene Interaction

- events mouse and keyboard events
- picking pick objects with the mouse
- manipulators interact with objects
- draggers add callback functions for everything
- sensors notifications for some reasons
- engines connect input/output of engines with fields

Events

- SoMouseButtonEvent (mouse press and release events)
- SoKeyboardEvent (keyboard press and release events)

```
// Declaration of a callback function
SoEventCallback *myEventCB = new SoEventCallback;
myEventCB->addEventCallback(myKeyPressCB, myUserData);

// Adding the function's node to the scene graph
separator->addChild(myEventCB);

// Implementation of the callback function
void myKeyPressCB(void *userData, SoEventCallback *eventCB)
{
// SoKeyboardEvent
   if (SO_KEY_PRESS_EVENT(event,Q)) exit(0);
}
```

Sensors

- SoSensor
- detect changes to time or to nodes
- incorporate callback functions in alarm cases
- SoAlarmSensor — one-time callback
- SoTimeSensor — repeat callback at regular intervals
- SoNodeSensor — detects node changes or changes to children of group nodes
- SoFieldSensor — attached to a field
- SoIdleSensor — triggered when there is nothing to do

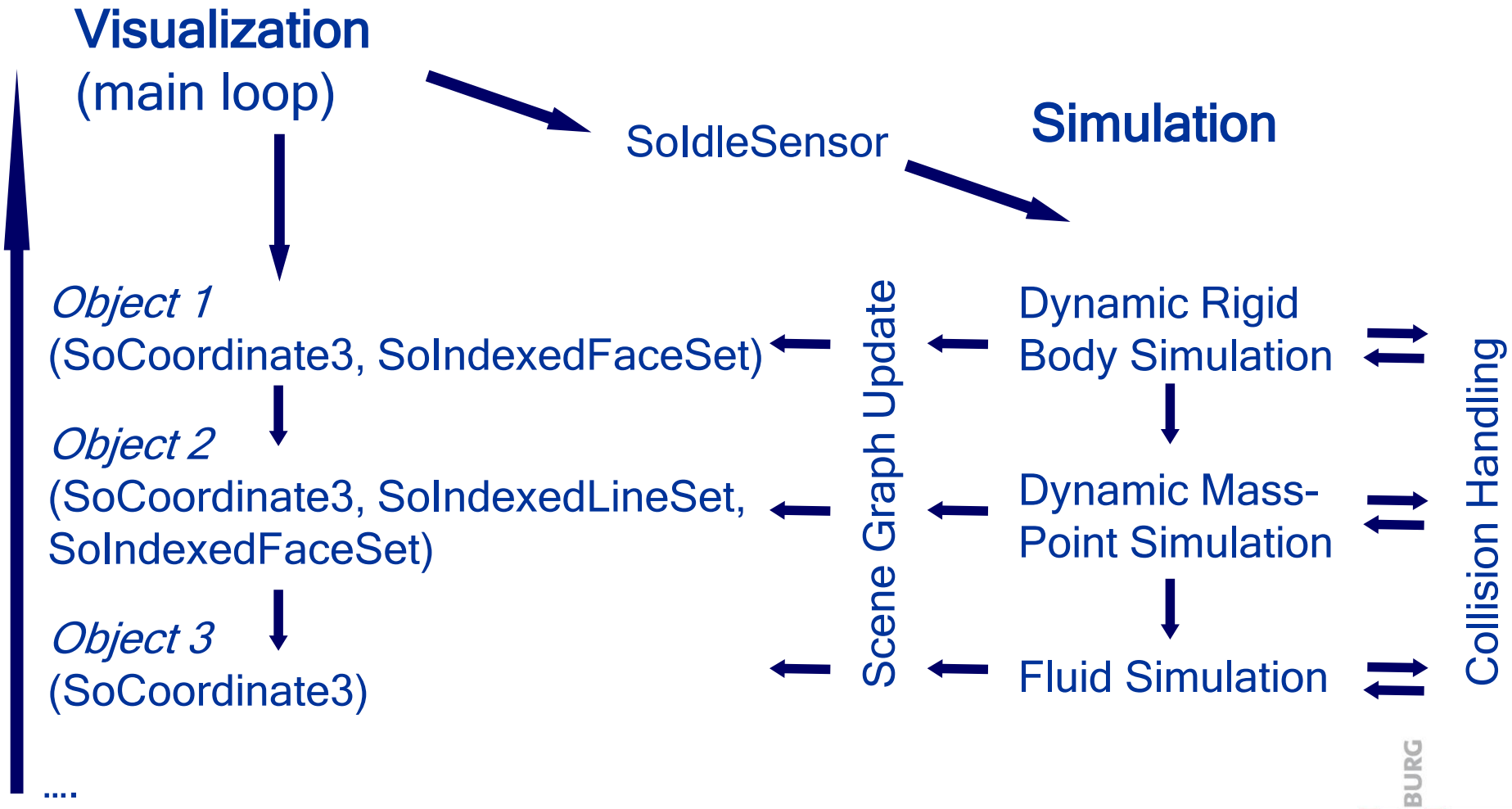
Scene Interaction - Summary

- events
- picking
- sensors

Open Inventor - Summary

- file format and C / C++ visualization library
- scene graph
- nodes
  - classes with parameters and methods
  - reference counter
- camera, light, object shape, material, shading, texture, transformations
- event handling and animation

Simulation Environment



References

- Paul Strauss, Rikk Carey, “An object-oriented 3D graphics toolkit,” *ACM Computer Graphics, Proc. of SIGGRAPH’92*, Chicago, July 26-31, 1992, pp. 341-349. *www.acm.org/dl*
- Josie Wernecke, *The Inventor Mentor*, Addison-Wesley Publishing Company, Reading, Massachusetts,
ISBN 0-201-62495-8, 1994.