Open Inventor
Scene Graph

CSE 681 Autumn 2011

Han-Wei Shen
Open Inventor

- Developed by SGI, originally IRIS Inventor
- a C++ object oriented **retained mode** 3D graphics API
- Built on top of OpenGL to allow for a higher level of programming
- Retained Mode: a persistent representation of graphical objects, their spatial relationships, appearance and the position of the viewer is held in memory (Open Inventor)
- Immediate Mode: data the above are transmitted one at a time for rendering. At each frame, the objects are retransmitted. (OpenGL)
Scene Graph

- A general data structure that arranges the logical and spatial representation of a graphical scene
- A collection of nodes in a graph (typically acyclic) or tree structure
- A node may have many children but often a single parent
- An operation (transformation, for example) applied to a parent is propagated to all of its children
- Rendering is done while traversing the tree in certain
Scene Graph Example

```
robot
  / \ 
body /   
  \  
  head

xf1  bronze  myCylinder  xf2  silver  mySphere
```
Scene Graph Example

```
robot
  --body
    bronze /bodyCylinder
    leftLeg /rightLeg
      --leftTransform
      --rightTransform
      leg
        thigh /calf
        calfTransform /footTransform
        call /foot
    head
      silver /headSphere
```

Scene Graph Nodes

- The fundamental element of a scene graph
- it contains data and methods that define some specific 3D shape, property, or grouping
- A node is connected to other nodes in the database to construct a hierarchy

Types of Nodes
- Shape nodes
- Property nodes
- Group nodes
- Light/Camera nodes
Node classes

* Shape nodes:
  - cone, cube, cylinder, faceset, sphere, quadmesh, lineset, ...

* Property nodes:
  - transform, basecolor, coordinates, drawstyle, normal, material...

* Light/camera nodes:
  - PointLight, SpotLight, DirectionLight, PerspectiveCamera, OrthographicCamera

* Group nodes: Group, Separator, Manipulator, ...
Group Node

- A group node is a container for collecting child objects such as shape and property (‘head’ is an SoGroup, for example)
- A node typically inherits property from its parent; children of a group node usually inherit from prior siblings
Separator Group Node

* A special type of group node, which save state before and restore state after traversing children
Camera

- SoCamera
- SoPerspectiveCamera
- foreshortening
- SoOrthographicCamera
- parallel projection
Light

- SoLight
- SoPointLight
- SoDirectionLight
- SoSpotLight
Name Convention

- **Sb**: Open Inventor Basic Type (for Scene basic)
  - SbVector3f, SbRotation, SbMatrix, etc
- **So**: All other classes (for Scene Object)
  - SoTransform, SoCube, SoSphere, SoMaterial, SoLight, etc

Each node is composed of a set of data elements, known as fields.

- e.g., SoPointLight contains four fields: intensity, color, location, and on/off
- Many fields’ values can be set and obtained with
// create body parts
SoTransform *xf1 = new SoTransform;
xf1->translation.setValue(0.0, 3.0, 0.0);

SoMaterial *bronze = new SoMaterial;
bronze->ambientColor.setValue(.33, .22, .27);
bronze->diffuseColor.setValue(.78, .57, .11);
bronze->specularColor.setValue(.99, .94, .81);
bronze->shininess = .28;

SoCylinder *myCylinder = new SoCylinder;
myCylinder->radius = 2.5;
myCylinder->height = 6;

// construct body out of parts
SoSeparator *body = new SoSeparator;
body->addChild(xf1);
body->addChild(bronze);
body->addChild(myCylinder);

// create head parts
SoTransform *xf2 = new SoTransform;
xf2->translation.setValue(0, 7.5, 0);
xf2->scaleFactor.setValue(1.5, 1.5, 1.5);

SoMaterial *silver = new SoMaterial;
silver->ambientColor.setValue(.2, .2, .2);
silver->diffuseColor.setValue(.6, .6, .6);
silver->specularColor.setValue(.5, .5, .5);
silver->shininess = .5;

SoSphere *mySphere = new SoSphere;

// construct head out of parts
SoSeparator *head = new SoSeparator;
head->addChild(xf2);
head->addChild(silver);
head->addChild(mySphere);

// add head and body
SoSeparator *robot = new SoSeparator;
robot->addChild(body);
robot->addChild(head);
Open Inventor File Format

* Comments begin with #
  - #Inventor V2.0 ascii

* Writing a Node
  - Name of the node
  - Open brace {
  - Fields within the node, followed by children of the node if any
  - Close brace }

# sphere
Separator {
  Material {
    diffuseColor 1 0 0
  }
  Sphere {}
}

Values within a Field

Fields within a node are written as the name of the field, followed by the values contained in the field.

```plaintext
Transform {
    translation 0 -4 0.2
}

LightModel {
    model BASE_COLOR
}

Material {
    ambientColor .3 .1 .1
    diffuseColor [.8 .7 .2,
                  1 .2 .2,
                  .2 1 .2,
                  .2 .2 1]
    specularColor .4 .3 .1
    emissiveColor .1 0 .1
```
Sphere Examples

```inventor
#Inventor V2.1 ascii

PerspectiveCamera {
  position 0 0 5
  orientation 0 0 1 0
  nearDistance 1
  farDistance 20
  focalDistance 15
  heightAngle 0.72
}

# set lighting model to BASE_COLOR for ivview
LightModel {
  model BASE_COLOR
}

Separator {
  # sphere 1
  Separator {
    Material {
      diffuseColor 1 0 0
    }
    Sphere {}
  }
  # sphere 2
  Separator {
    Transform {
      translation 1.5 0 0
    }
    Material {
      diffuseColor 0 1 0
    }
    Sphere {}
  }
}
```

Coin3D

- In this class, the version of Open Inventor that we will use is called Coin3D (http://www.coin3d.org/)
- Coin3D includes a complete Open Inventor as well as other advanced 3D rendering features and GUI
- You need to install Coin3D on your local Windows or Mac machine if you want to use your own machine instead of CSE Unix server stdsun
Open Inventor 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);

SoXtRenderArea  *mycanvas = new SoXtRenderArea;

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