

# *Simulation in Computer Graphics*

# *Exercises*

Matthias Teschner

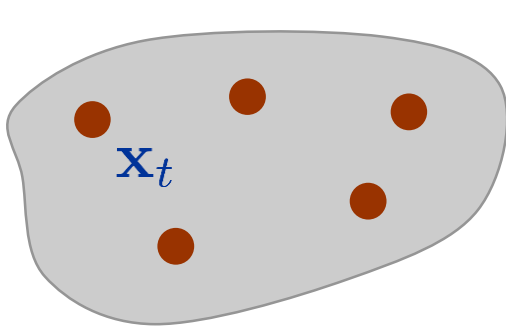
Computer Science Department  
University of Freiburg

Albert-Ludwigs-Universität Freiburg

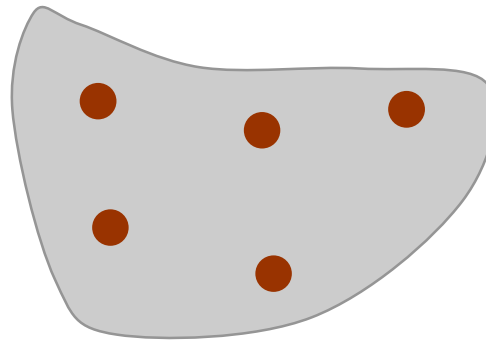
**UNI  
FREIBURG**

# General Concept

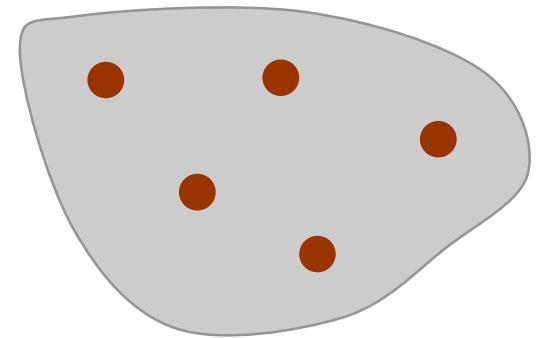
- simulation of a set of particles
  - update particle positions  $\mathbf{x}_t$  per time step  $t$



deformable object



fluid



rigid object

- visualization
  - of dynamically changing particle positions
  - of additional properties, e.g. velocities or force

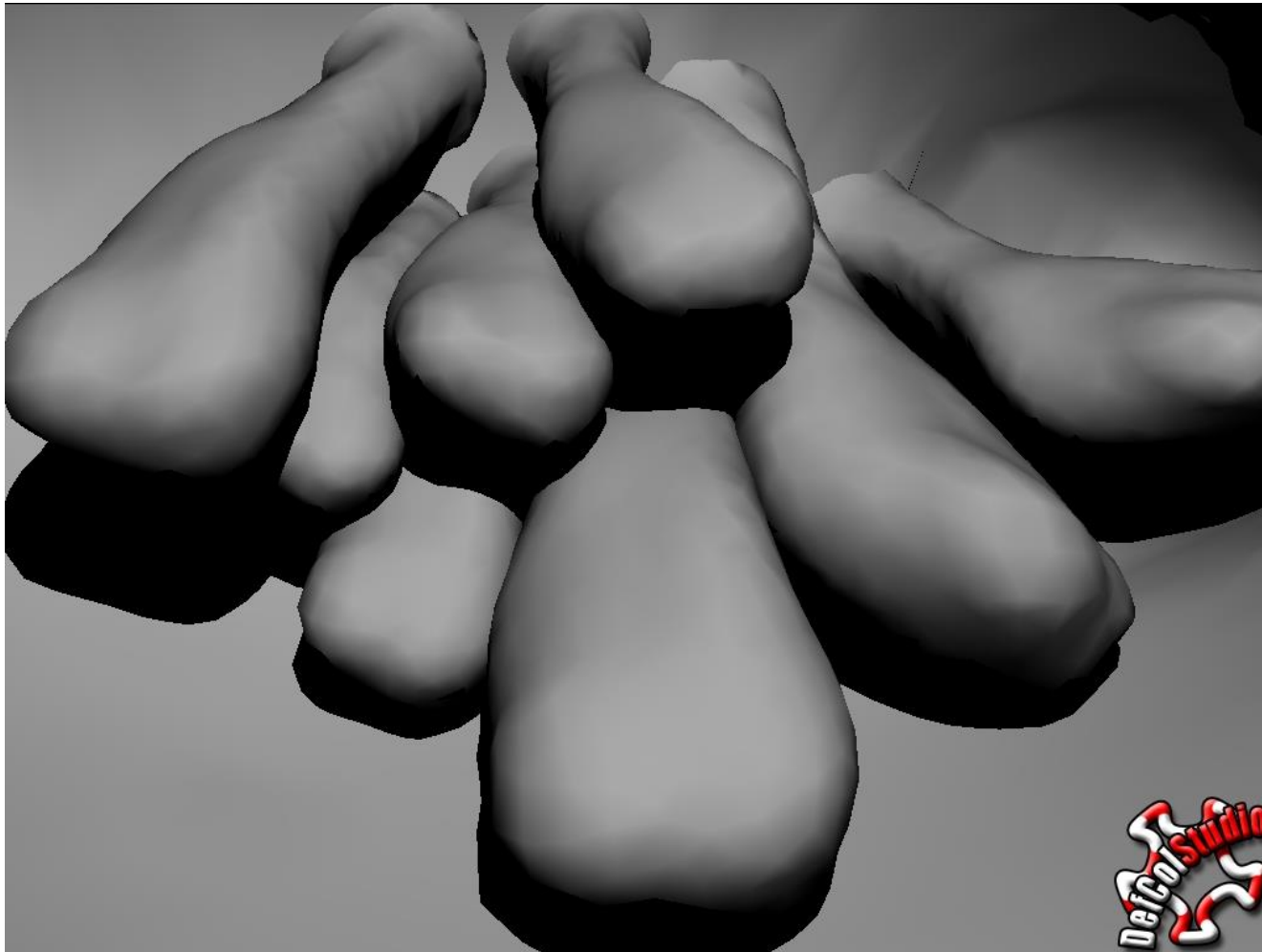
# Visualization Primitives

---

- points
  - to illustrate particle positions
  - simple representation, e.g. cube, tetrahedron, or sphere
- line segments
  - to illustrate particle connections, e.g. springs
- triangles
  - to illustrate the simulation domain (triangle mesh)
- tetrahedra
  - to illustrate volumetric elements in deformable objects

# Visualization Example

---



# Visualized Simulation

**Visualization**  
(main loop)

Callback

**Simulation**

*Object 1*  
(particles, lines, triangles, tetras)

*Object 2*  
(particles, lines, triangles, tetras)

*Object 3*  
(particles, lines, triangles, tetras)

Rigid body  
simulation

Deformable solid  
simulation

Fluid simulation

Update of particle positions

Collision Handling

visualize particle positions

compute particle positions

# Visualization Tools

---

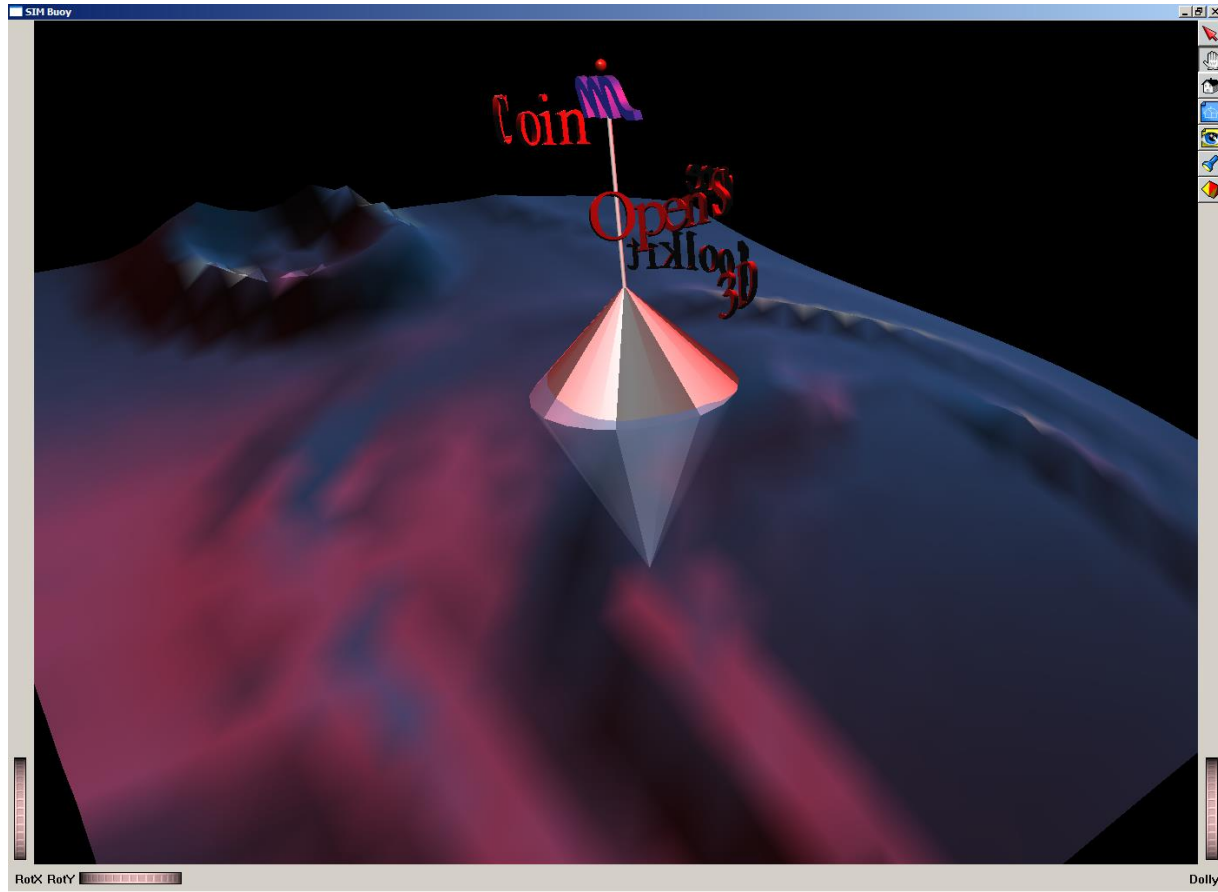
- Coin3D [github.com/coin3d](https://github.com/coin3d)
- OpenGL
- VTK [www.vtk.org](http://www.vtk.org)
- OSG [www.openscenegraph.org](http://www.openscenegraph.org)
- Ogre3D [www.ogre3d.org](http://www.ogre3d.org)
- ...

# Visualization Tools

---

- Coin3D
  - exercises on web page use Coin3D
  - can be difficult to install
- VTK
  - sample setting on web page
  - easy to install and to use
  - supported
  - less optimal documentation
  - better performance compared to Coin3D

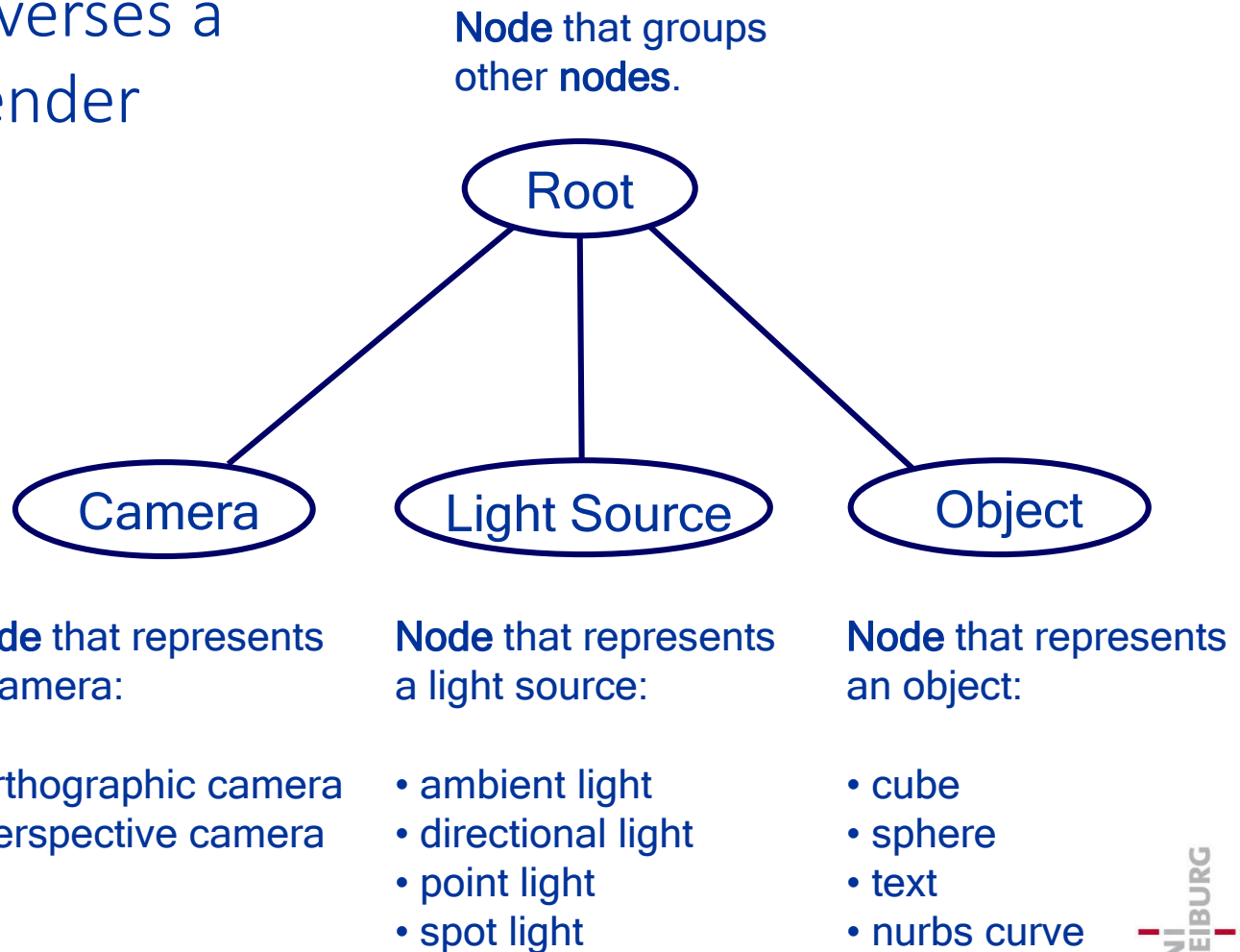
# Coin3D - Example





# A First Scene Graph

- Coin3D traverses a graph to render the scene



# C/C++ Example

---

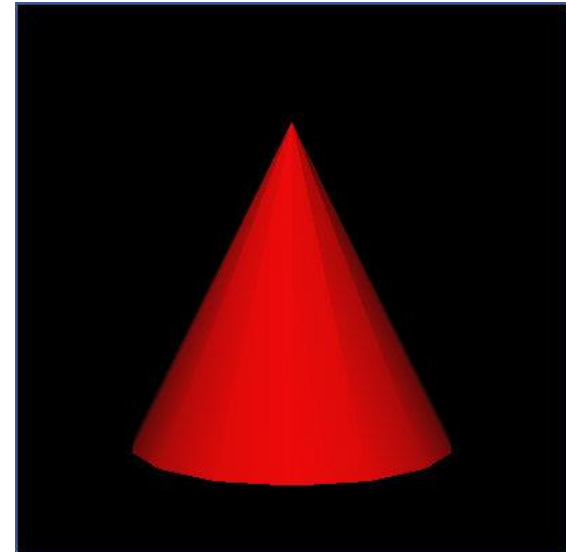
```
SoSeparator          *root          = new SoSeparator;  
SoPerspectiveCamera *myCamera = new SoPerspectiveCamera;  
SoDirectionalLight  *myLight      = new SoDirectionalLight;  
SoCone               *myCone       = new SoCone;
```

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

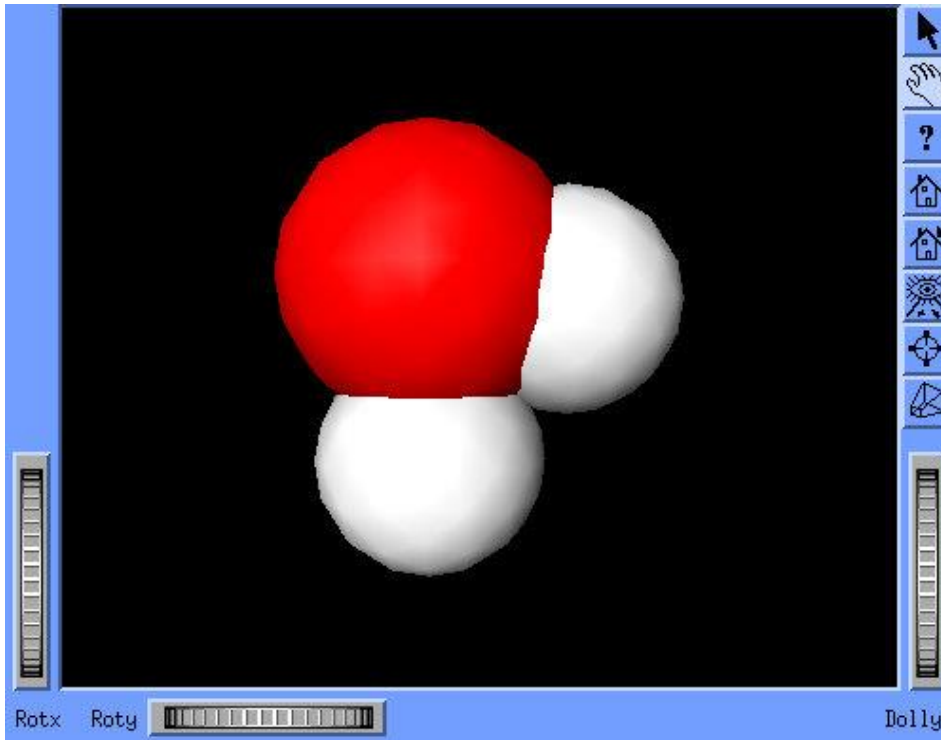
```
myCamera->viewAll (root);
```

```
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

rotation

zoom

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

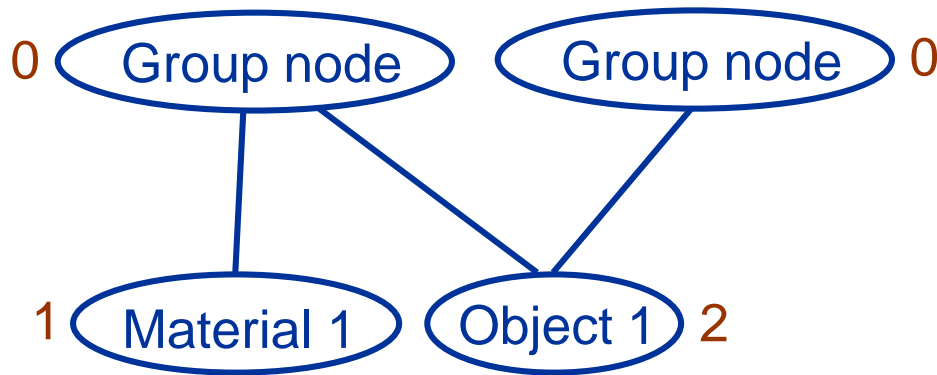
# Node Types

---

- 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()`

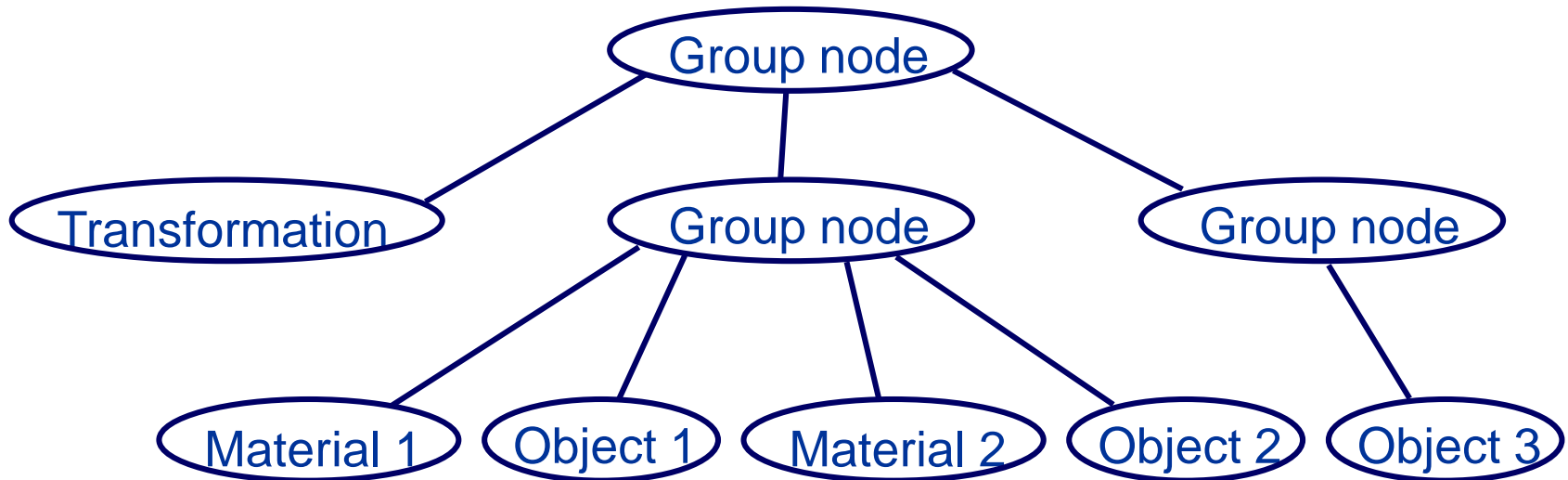
# Node Deletion

---

- when a node's reference counter decreases from 1 to 0, the node is deleted by Coin3D
- 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

# Scene Interaction

---

- events            mouse and keyboard events
- sensors           notifications for some reasons



# 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
- SoldleSensor      triggered when there is nothing to do

# Visualized Simulation

**Visualization**  
(main loop)

Callback

**Simulation**

*Object 1*  
(particles, lines, triangles, tetras)

*Object 2*  
(particles, lines, triangles, tetras)

*Object 3*  
(particles, lines, triangles, tetras)

Rigid body  
simulation

Deformable solid  
simulation

Fluid simulation

Update of particle positions

Collision Handling

visualize particle positions

compute particle positions