

DWARF 3D Viewer enhancements

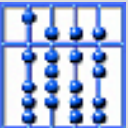
Fabian Sturm

Lehrstuhl für Angewandte Softwaretechnik

Institut für Informatik

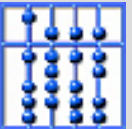
Technische Universität München

sturm@in.tum.de



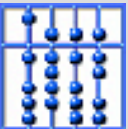
Overview

- Requirements
- Viewer API before the CAR project
- New API
- Use Case
- To do



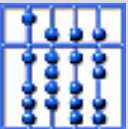
Requirements

- Non representative survey
 - Not enough responses to draw conclusion
 - Survey audience \neq Viewer audience



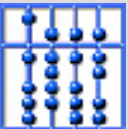
Requirements

- Simple way to insert movable objects
- Dynamic connection between objects and incoming PoseData
- Event based interface to address several Viewer at once vs. method based interface for reliability.
- Modifiable object properties
- SceneGraph agnostic interface
- No preference between Stability, Speed, Feature completeness

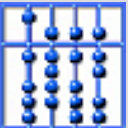
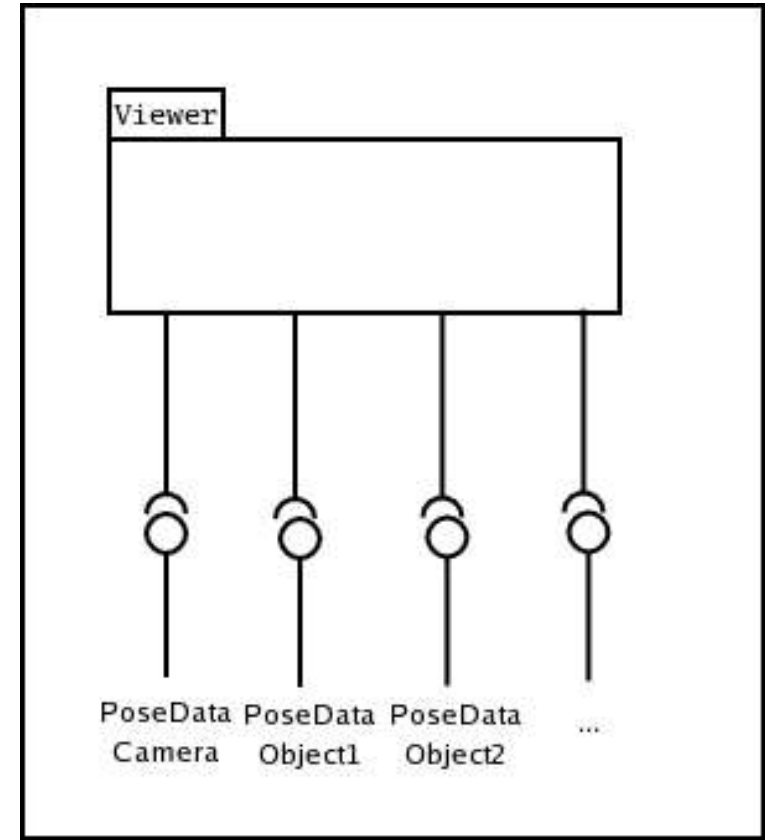
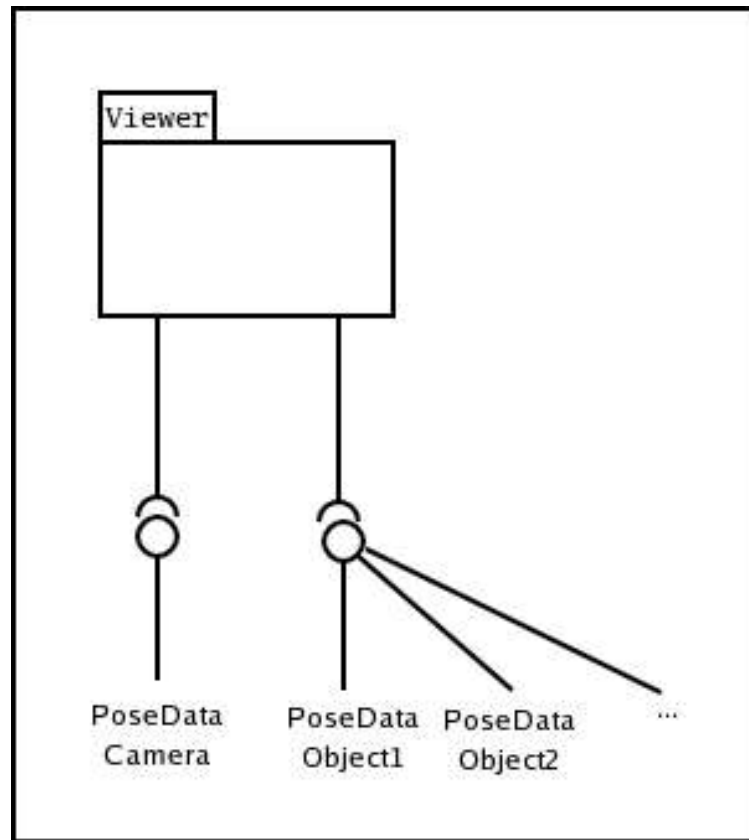


Viewer API before CAR

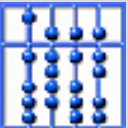
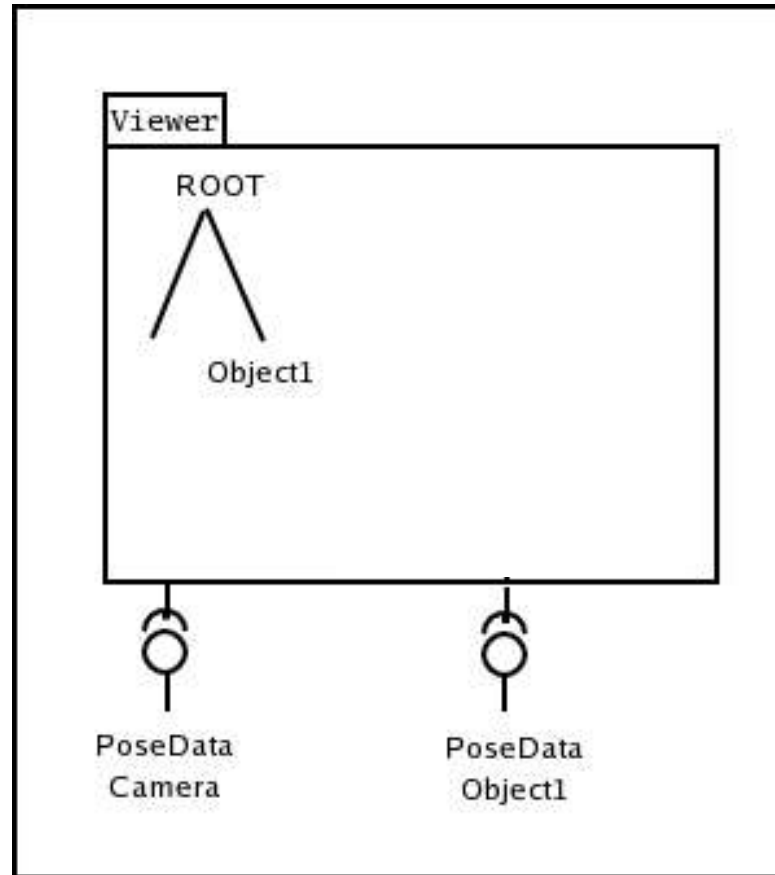
- PoseData (camera, sun, objects)
- SceneData
 - CreateObject
 - DeleteObject
 - ReplaceScene
 - SuperImpose
- UserAction
 - SelectVirtualObject
 - DeselectVirtualObject
- CameraMatrix



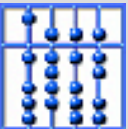
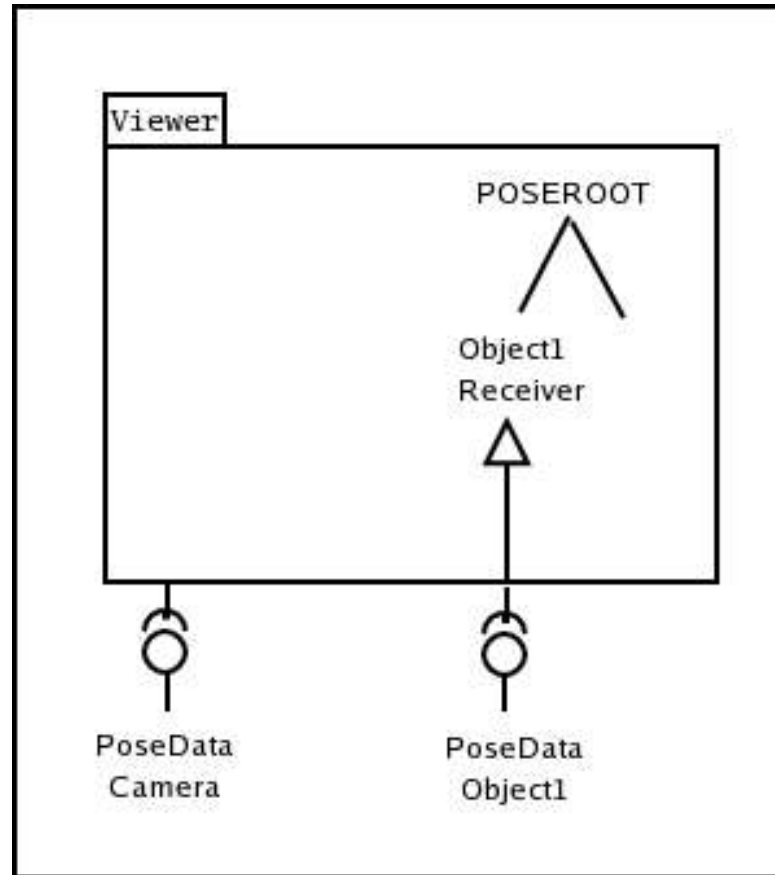
Old vs. New PoseData Interface



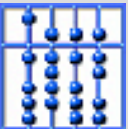
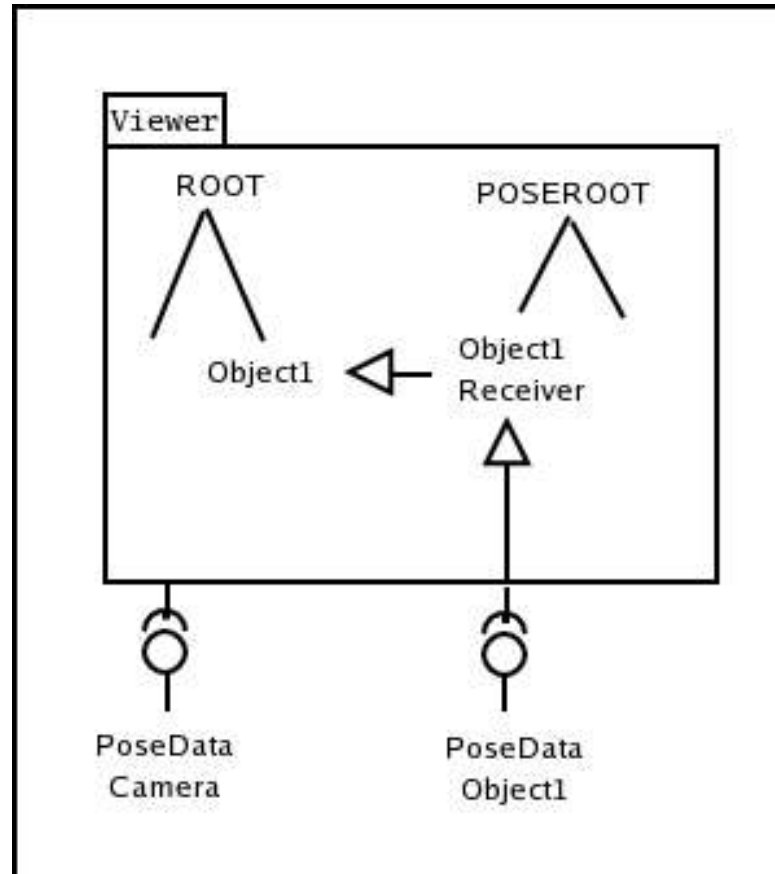
Old vs. New PoseData Interface



Old vs. New PoseData Interface

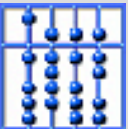


Old vs. New PoseData Interface



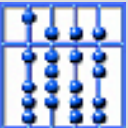
Old vs. New PoseData Interface

- PoseData Needs can be created at runtime
- Connection multiplexing back to Attribute and Predicate level
- Arbitrary connection between Object and PoseData Receiver



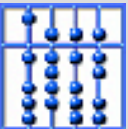
Stacked Objects

- Both Objects and PoseDataReceiver can have a parent
 - Objects will inherit their parents transform
 - PDR will multiply the inverse of their parents transform on own incoming position - > relative movement



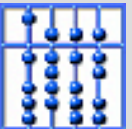
SuperImpose

- There is no more superimpose
- We have one perspective camera rendering everything under ROOT
- And one Orthographic camera rendering everything under FOREGROUNDROOT

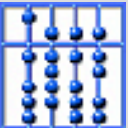
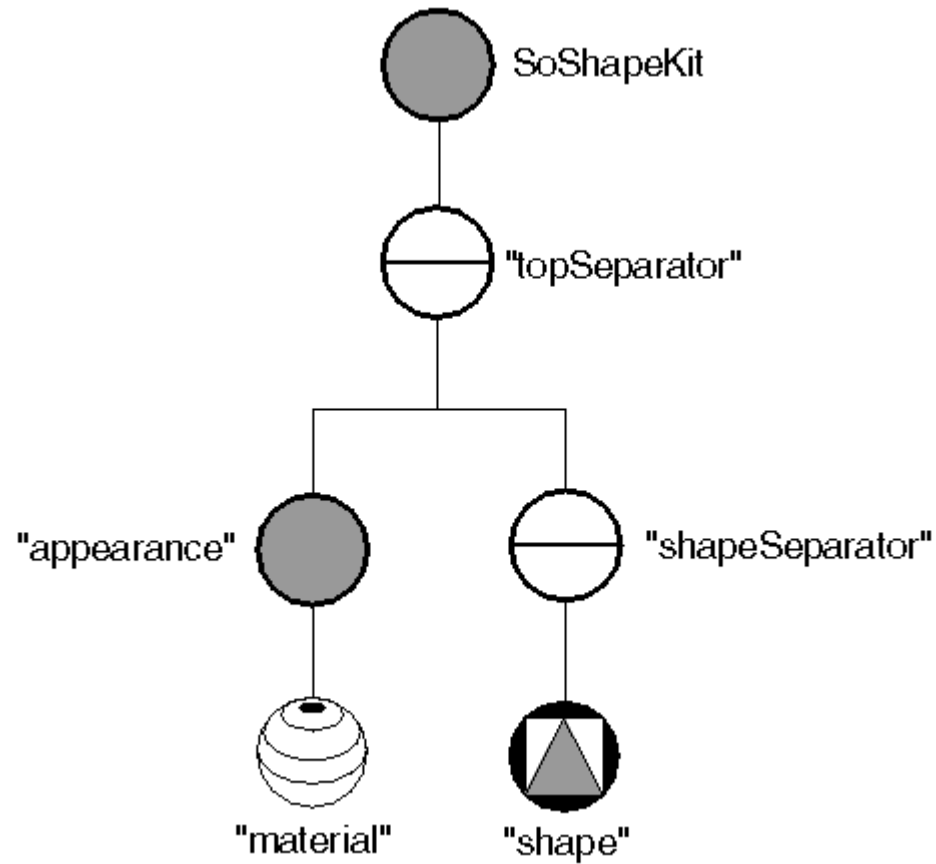


Object Properties

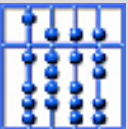
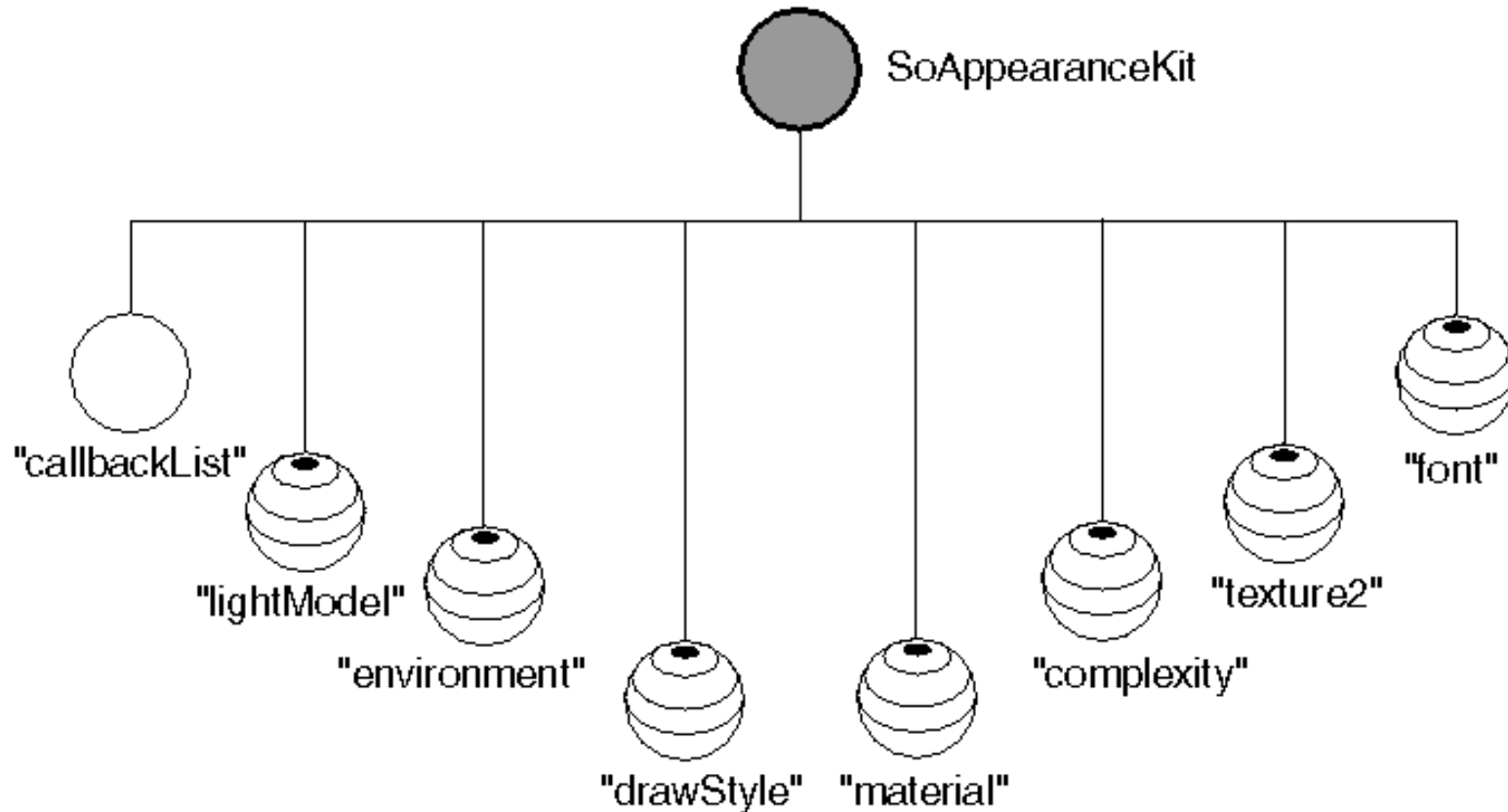
- Properties can now be modified at runtime
 - Appearance
 - LightModel
 - Drawstyle
 - Material
 - ...
 - Transform
 - ScaleFactor



Object Properties

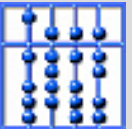


Object Properties



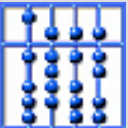
Object Properties

- "drawStyle { style LINES }
transform { scaleFactor 2.0 2.0 2.0 } "



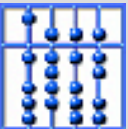
Object Properties

- "drawStyle { style LINES }
transform { scaleFactor 2.0 2.0 2.0 }"
- **! Warning this interface is dependent on Inventor Syntax !**



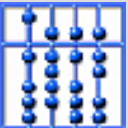
New API

- ViewerControl.idl
 - createObject
 - createPoseDataReceiver
 - moveObject
 - movePoseDataReceiver
 - connectObject
 - disconnectObject
 - replaceObject
 - deleteObject
 - deletePoseDataReceiver
 - setProperty



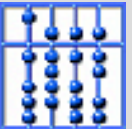
New API

- Composed Commands
 - createConnectedObject
 - deleteConnectedObject



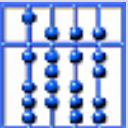
Use Case

- Sheep Demo
 - Insert new objects into scene
 - Track objects
 - Create magic Wand
 - Pickup objects
 - Drop Objects
 - Change Sheep color



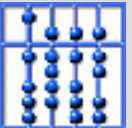
Use Case: Implementation

- UIC from Otmar Hilliges
 - Interactive Java application
- ViewerController from Martin Bauer
 - Python script



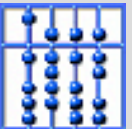
Use Case: Insert new Object

- `sheep = file("sheep.vrm1").read()`
- `viewer.createObject("sheep 1", "ROOT",
sheep)`



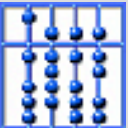
Use Case: Insert new Object

- `sheep = file("sheep.vrml").read()`
- `viewer.createObject("sheep 1", "ROOT", sheep)`
- `viewer.createConnectedObject("sheep 1", "ROOT", sheep)`



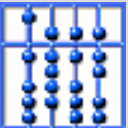
Use Case: Create Pose Receiver

- `viewer.createPoseDataReceiver("wand")`



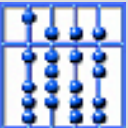
Use Case: Pickup Object

- `viewer.connectObject("sheep 1", "wand")`



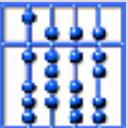
Use Case: Drop Object

- `viewer.connectObject("sheep 1", "sheep 1")`



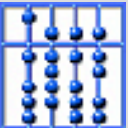
Use Case: Change Object Color

- `viewer.setProperty("sheep 1",
"material { diffuseColor 1 0 1 }")`



To be done:

- Camera is not yet a SoPoseDataReceiver
 - Interesting for putting objects easily fixed to camera
- SceneGraphManager should be the only one changing the SceneGraph
- SceneGraph changes must happen in order
- Needs might actually be abilities and vice versa



Questions?

