Separating Semantics from Rendering: 
A Scene Graph based Architecture for 
Graphics Applications

Robert F. Tobler
VRVis Research Center
Vienna, Austria
Motivation

Dynamic Scene Graphs
- varying support in various scene graph systems
- can be added to any scene graph
- *but:* no clean design has been proposed

Storing Semantics in Scene Graphs
- no direct support in any scene graph system
- semantics has been shoehorned into scene graphs using tags
- *but:* no clean design has been proposed
Scene Graphs have a long History...

- Inventor [Strauss & Carey, 1992]
- Open Inventor [Werneckke, 1993]
- OpenSG [Voss, Behr, Reiners & Roth, 2002]
- Open SceneGraph [Burns & Osfield, 2004]
- SceniX [Kunz & Miller, 2009]

...and many others
Dynamic Scene Graphs: Storing State in the Application

- references from application to scene graph nodes
- application code directly modifies scene graph
Dynamic Scene Graphs: Storing State in the Scene Graph

- Typically implemented with "needs update" flags
- State responsibilities not cleanly delineated
Observation

Typical Graphics Applications...
- take input data and user input
- build a scene graph
- render the scene graph

... work somewhat like a compiler:
Semantic & Rendering Scene Graph

- **Semantic Scene Graph**
- **Rendering Scene Graph**

- Material
- Holz

- Translation

- Shadows
- Bump Maps
Dynamic Scene Graphs?

**Classical Compiler:**
- program is translated into a static binary
- similar to a generating static scene graph

**Modern Just-in-Time Compiler:**
- program is compiled as needed
- translated binary can change over time (unused portions can be deleted)
- similar to a dynamic scene graph
On-the-fly Creation of Rendering Scene Graph

Traversing a semantic scene graph to construct a rendering scene graph.

Compiled rendering scene graph caching intermediate results.
Implementation: Rule Objects

Between Semantic and Rendering Scene Graph
- generated from a rule table indexed by node type
- contain state, communicate with application
- modify rendering scene graph as needed before traversing it

RuleObject

state
virtual Action
{
  // modify state and
  // rendering scene graph
  // before traversing it
}
MVC for Scene Graphs

Model

semantic scene graph

Controller

state
state
state
state
state
state
state

View

rendering scene graph

traversal cache
Example: Rotator Node

```c
RuleObject
Action {
  t = GetWorldTime();
  TrafoLeaf.Matrix = Rotation(t);
  return TrafoLeaf;
}
```

```
Group
- GetTrafo()

Rotator
- GetTrafo()
```

```
TrafoLeaf
- GetTrafo()
```

Example: Semantic Level-of-Detail Node

```
Action {
  return level[LodFunction(cameraState)]; }
```

Level 0

Level 1

Level 2
Example: Procedural Geometry Generation

- tree
- branch
- branch
- twig
- leaf
Multiple Views on the same Semantic Scene Graph

- Traversal cache B
- Shared cache
- Traversal cache A
Editing the Semantic Scene Graph

- Semantic node
- Editing state
- Back reference for modifications
- Rendering view
- Editing view with handles
Application Example: Web-Browser

A web-browser in this Architecture:

- HTML-Graph == semantic scene graph
- Memory Data Structures for display == rendering scene graph
The architecture is in use at the VRVis

The AARDVARK framework
- 150 libraries
- around 20 application projects
Conclusion & Future Work

An Architecture for Graphics Applications

- widely applicable in different scenarios
- handles dynamic scene graphs
- makes semantics explicit as a separate graph

So what’s left to do?

- apply compiler technology to the translation step, e.g.
  - extraction of constant sub-expressions ==
    == static scene graph extraction
- automatic dependency analysis
- all sorts of optimizations
Thanks to...

...my colleagues at the VRVis Research Center, especially:

- Stefan Maierhofer
- Matthias Buchetics
- Harald Steinlechner
- Michael Schwärzler
- Christian Luksch
Thank you for your attention!

Please visit us at

http://www.VRVis.at/