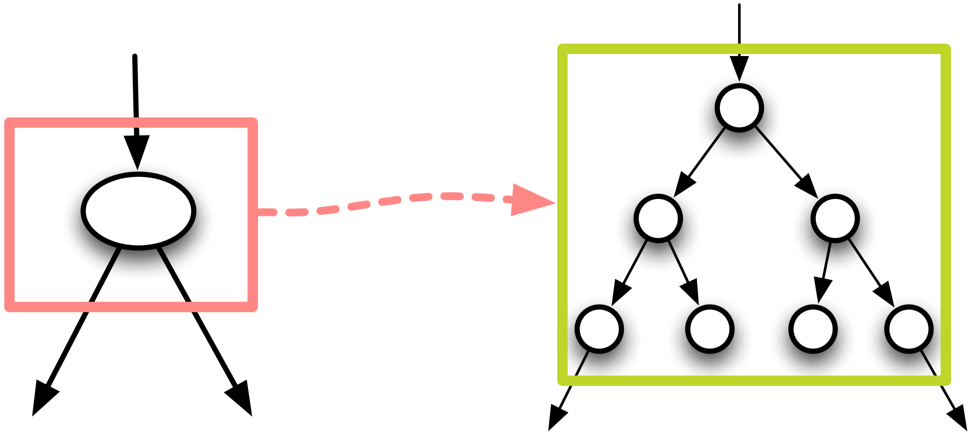


Separating Semantics from Rendering: A Scene Graph based Architecture for Graphics Applications

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

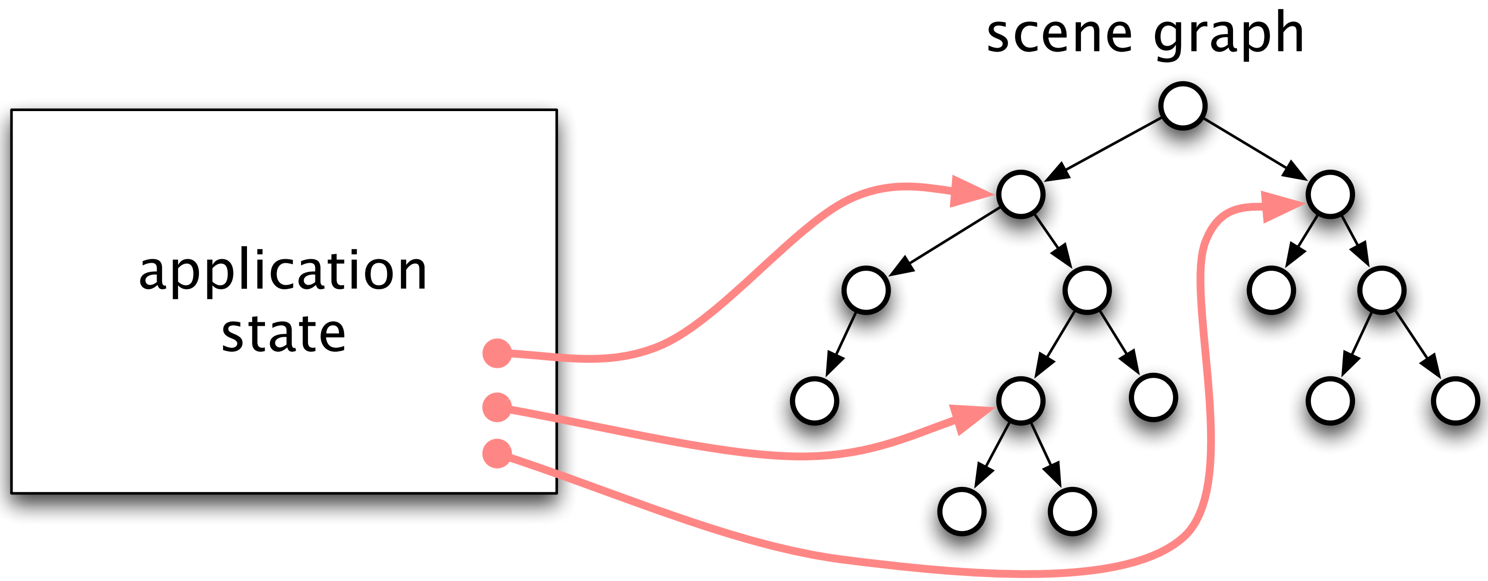
- no direct support in any scene graph system
- semantics has been shoehorned into scene graphs using tags
- ***but:*** no clean design has been proposed

Scene Graphs have a long History…

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

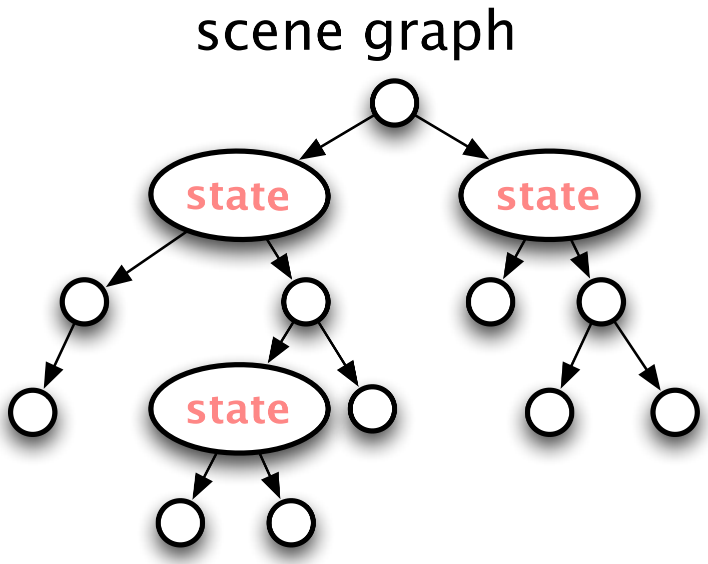
…and many others

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



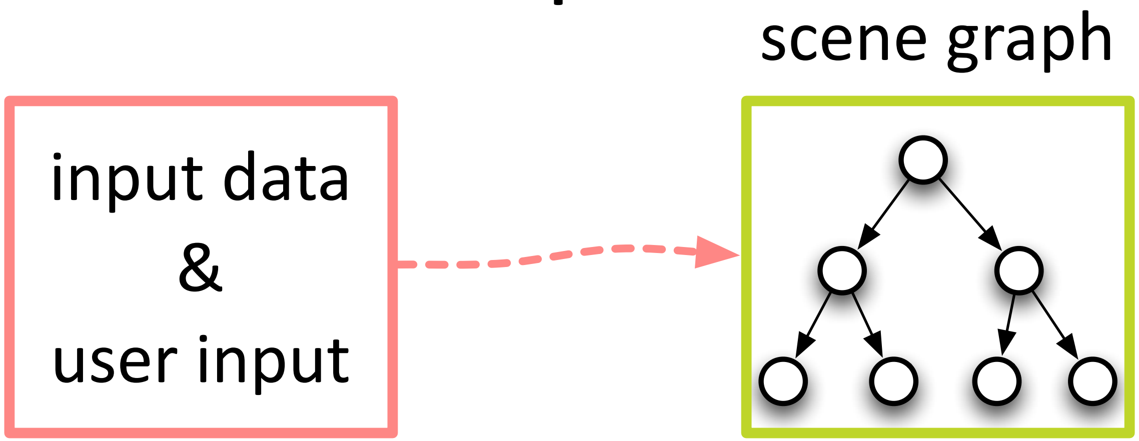
- typically implemented with "needs update" flags
- state responsibilities not cleanly delineated

Observation

Typical Graphics Applications…

- take input data and user input
- build a scene graph
- render the scene graph

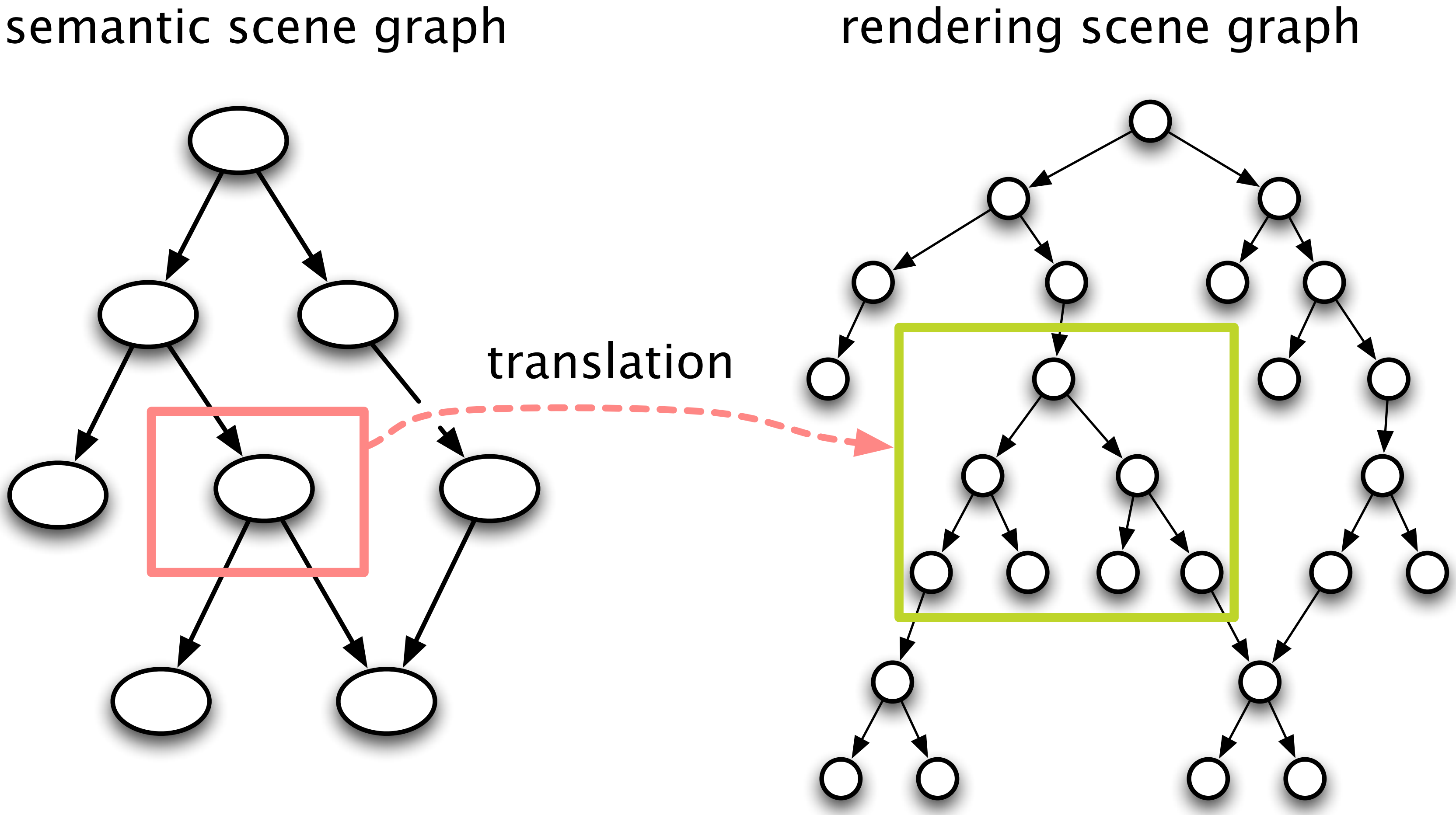
… work somewhat like a compiler:



scene graph

input data
&
user input

Semantic & Rendering Scene Graph



Dynamic Scene Graphs?

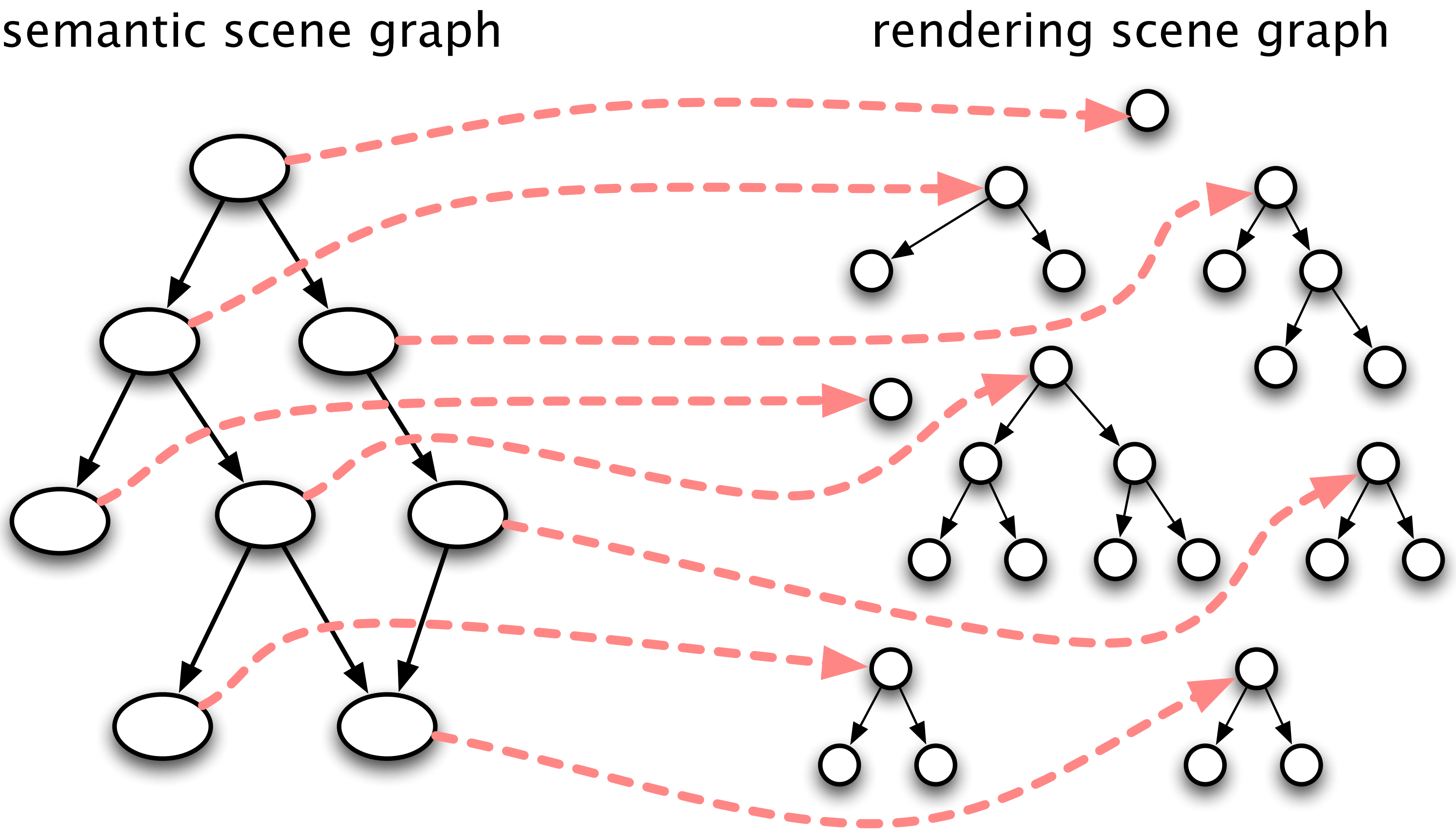
Classical Compiler:

- program is translated into a static binary
- similar to a generating static scene graph

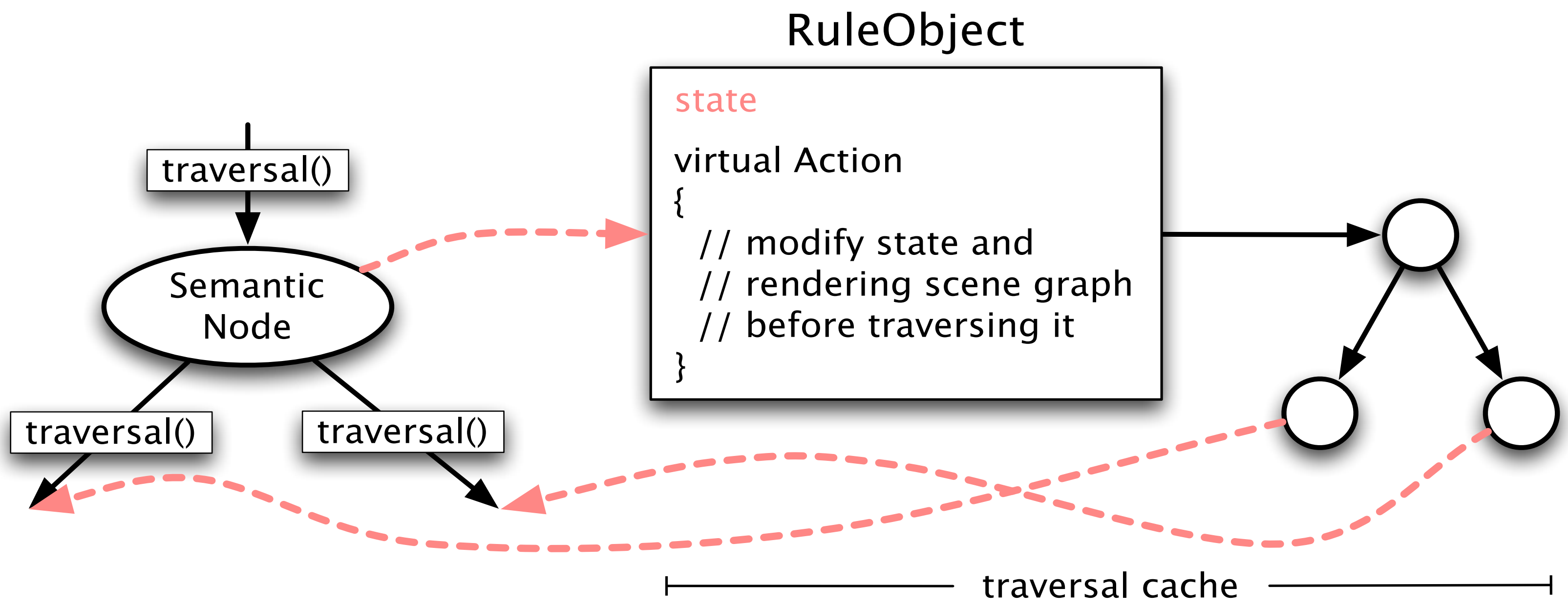
Modern Just-in-Time Compiler:

- 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



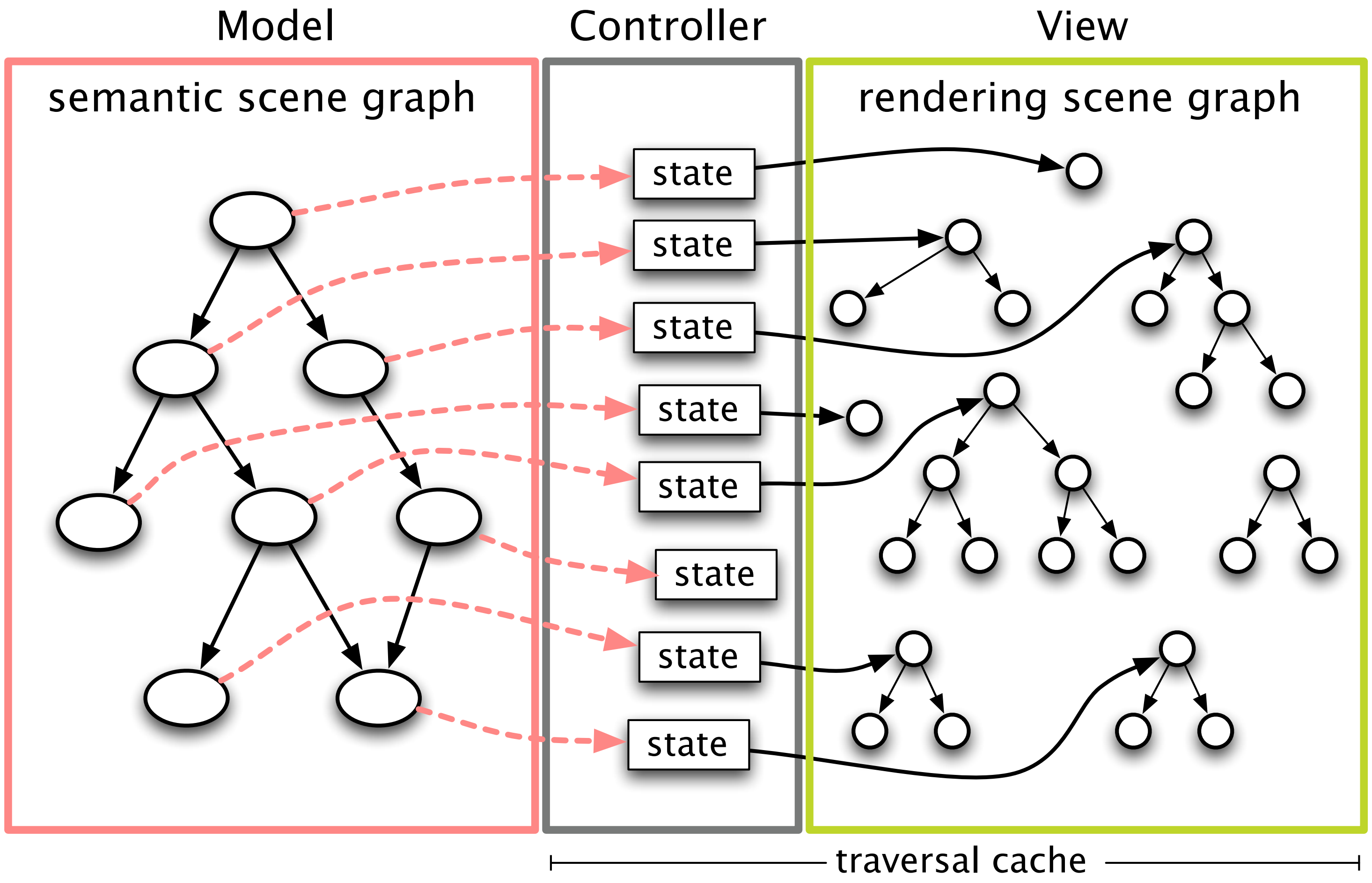
Implementation: Rule Objects



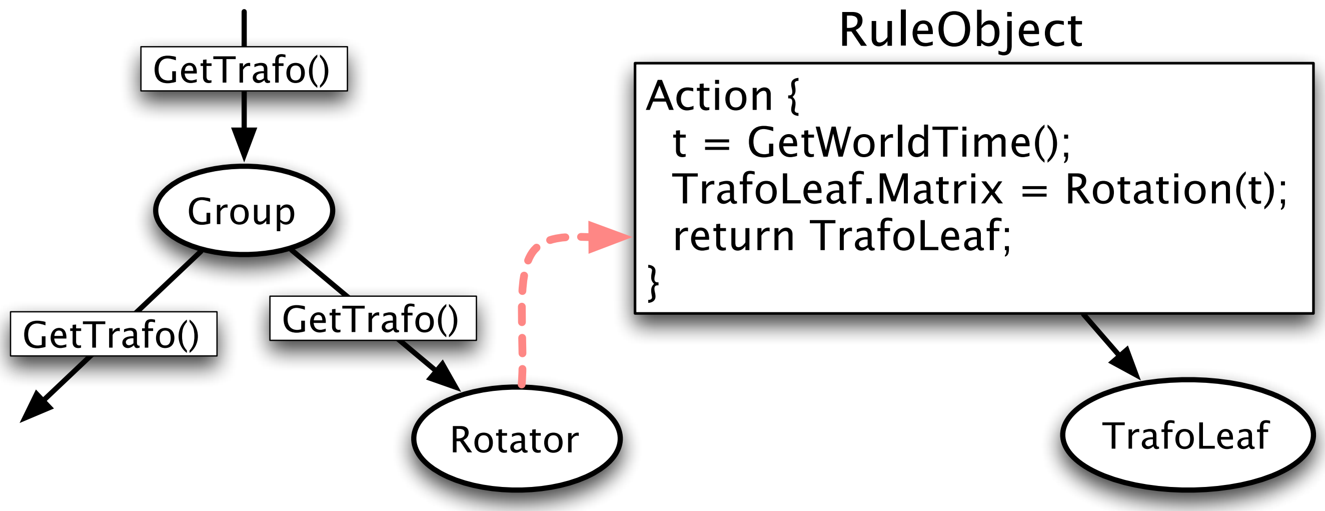
Between Semantic and Rendering Scene Graph

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

MVC for Scene Graphs



Example: Rotator Node



GetTrafo()

Group

GetTrafo()

GetTrafo()

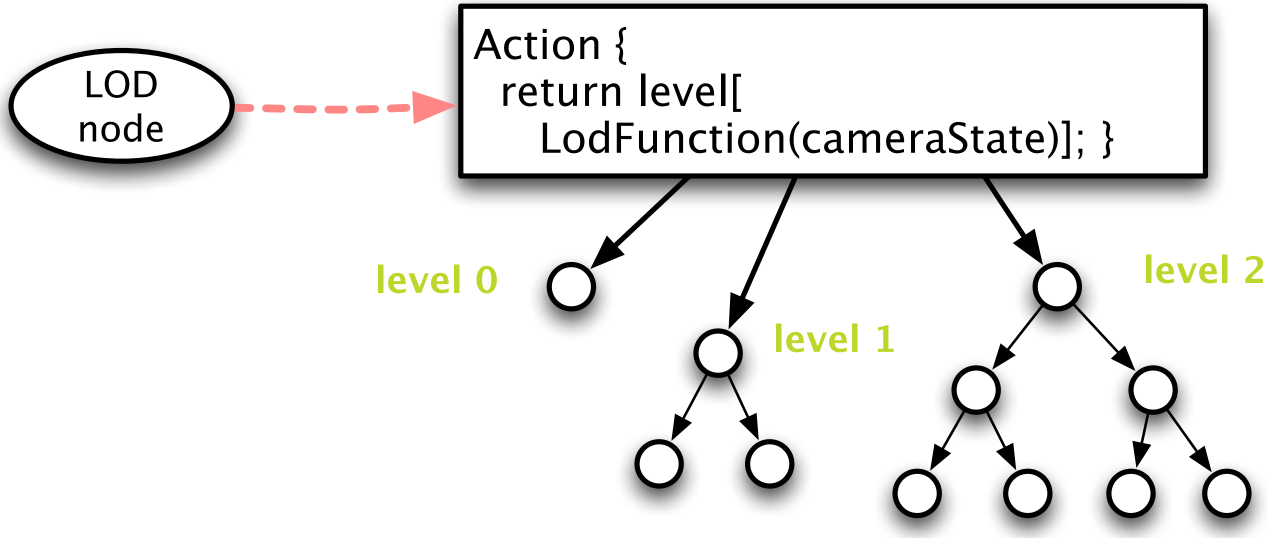
Rotator

RuleObject

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

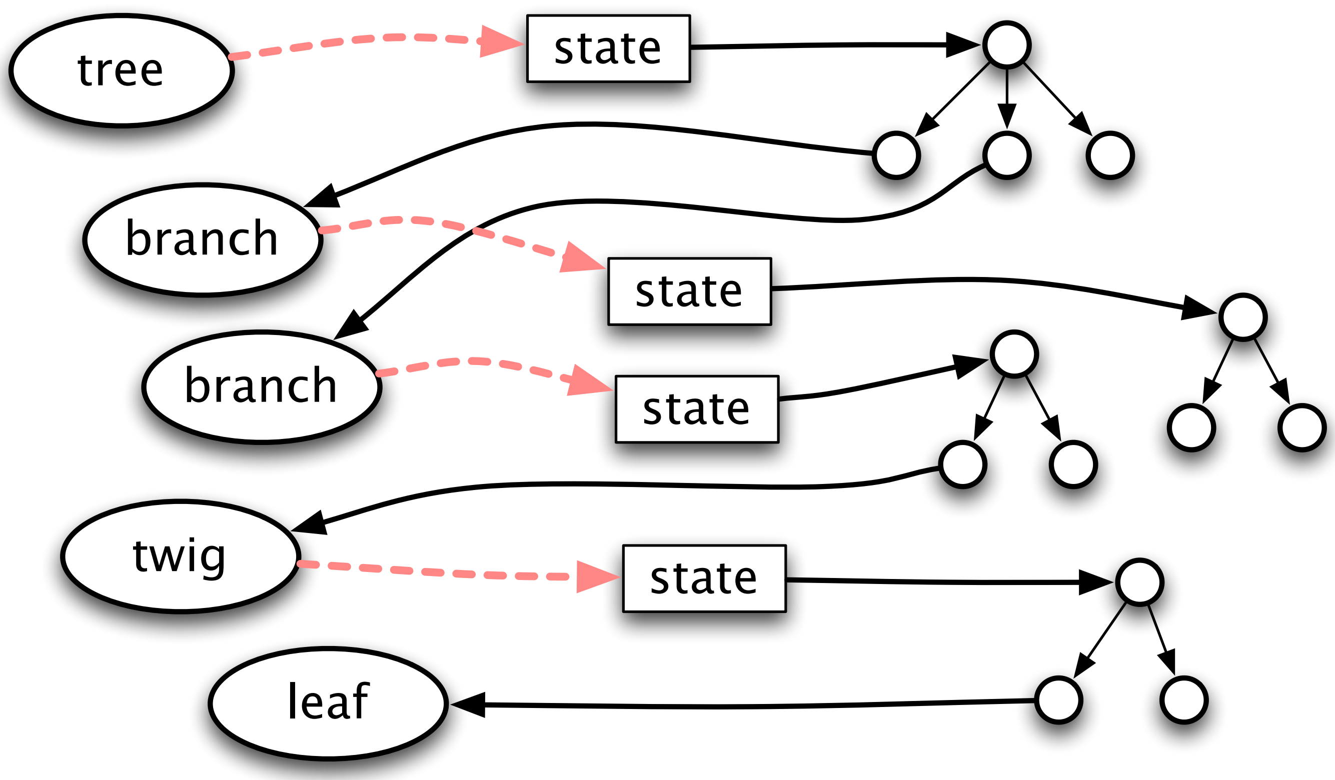
TrafoLeaf

Example: Semantic Level-of-Detail Node

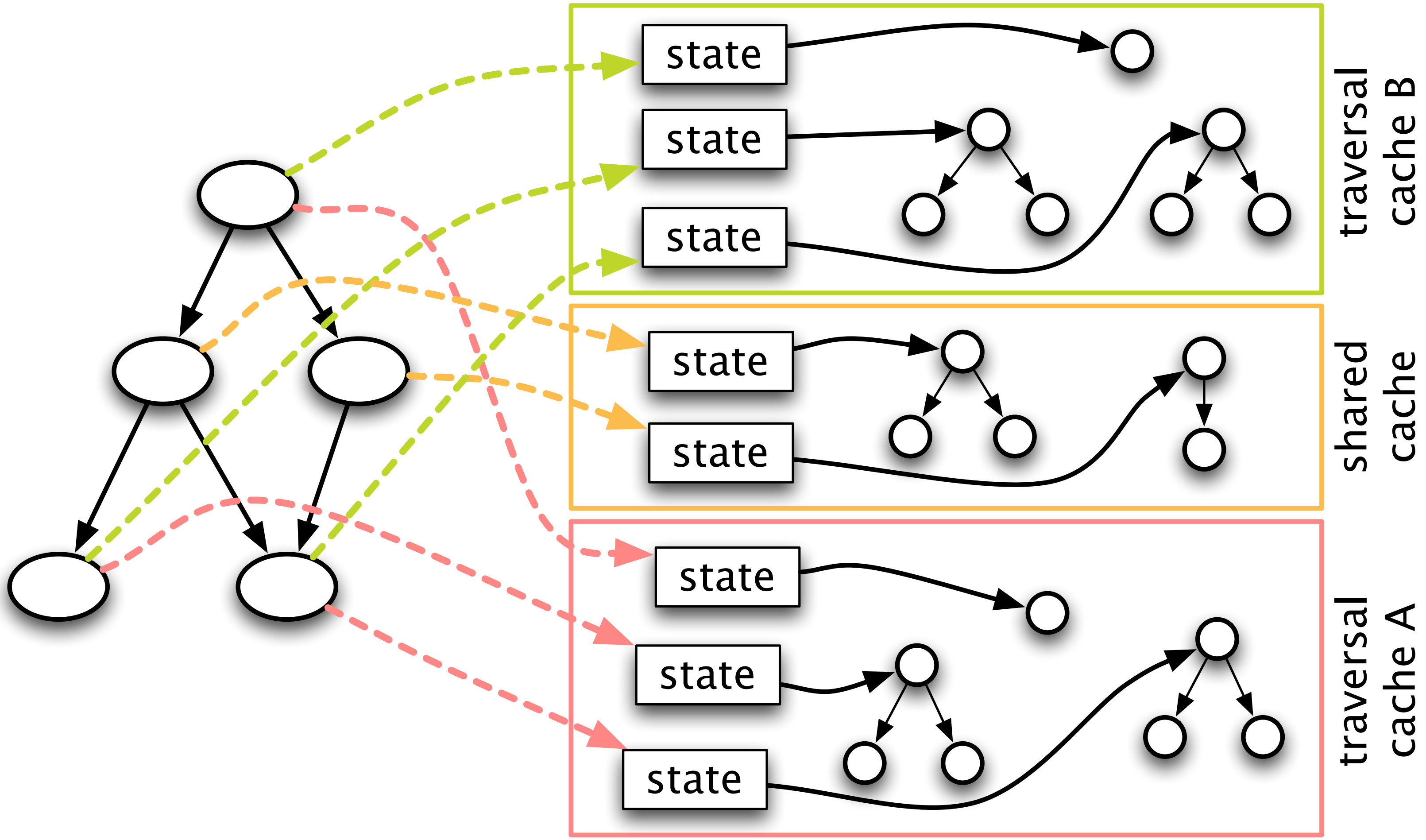


```
Action {
  return level[
     LodFunction(cameraState)]; }
```

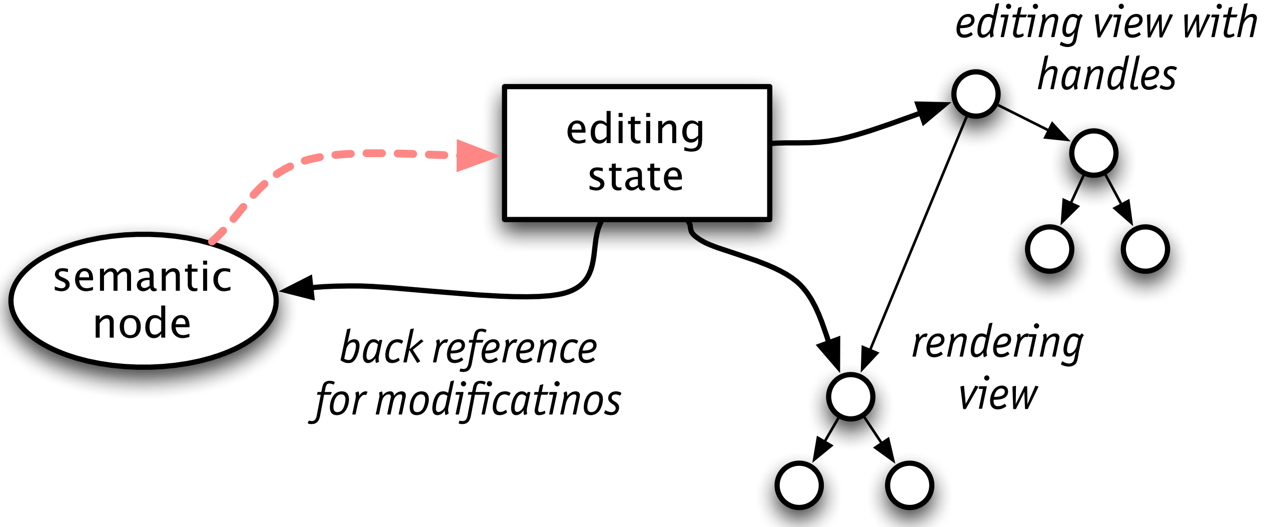
Example: Procedural Geometry Generation

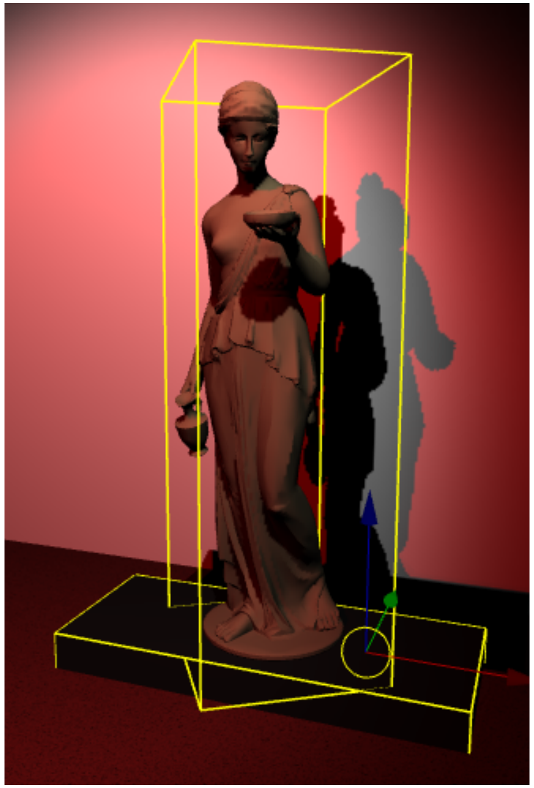


Multiple Views on the same Semantic Scene Graph

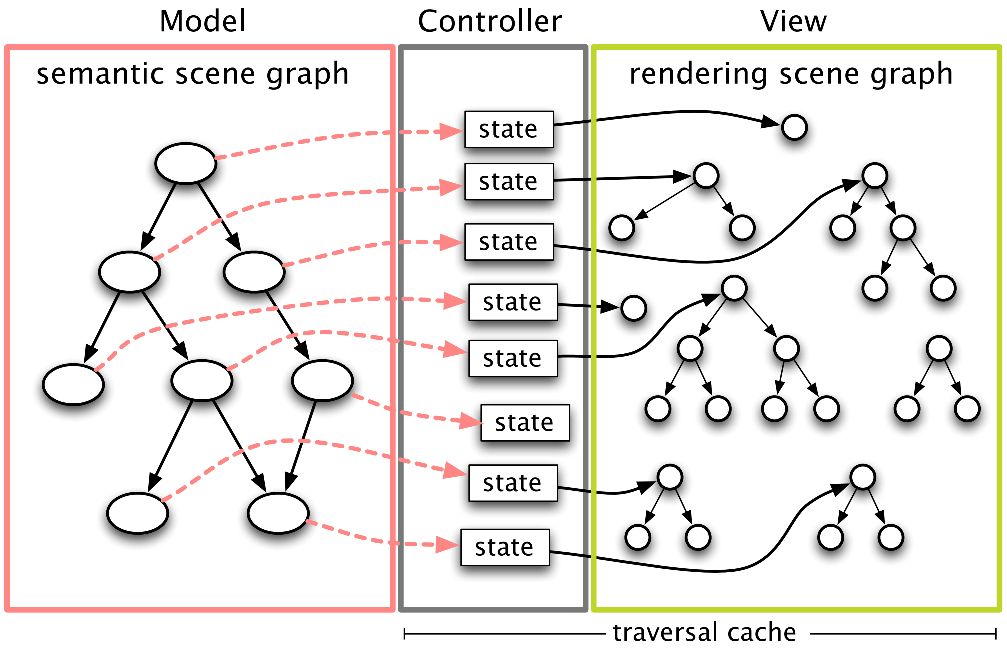


Editing the Semantic Scene Graph





Application Example: Web-Browser



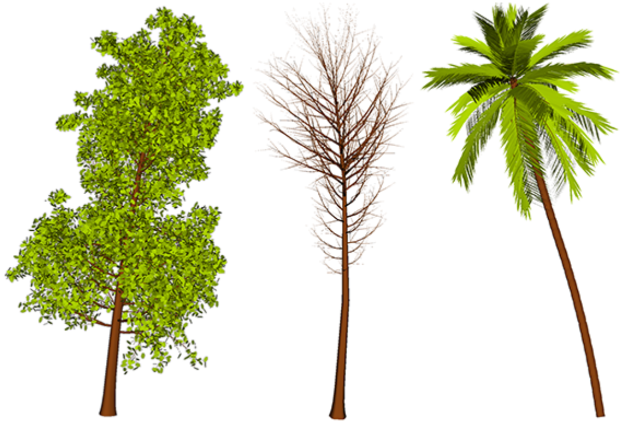
A web-browser in this Architecture:

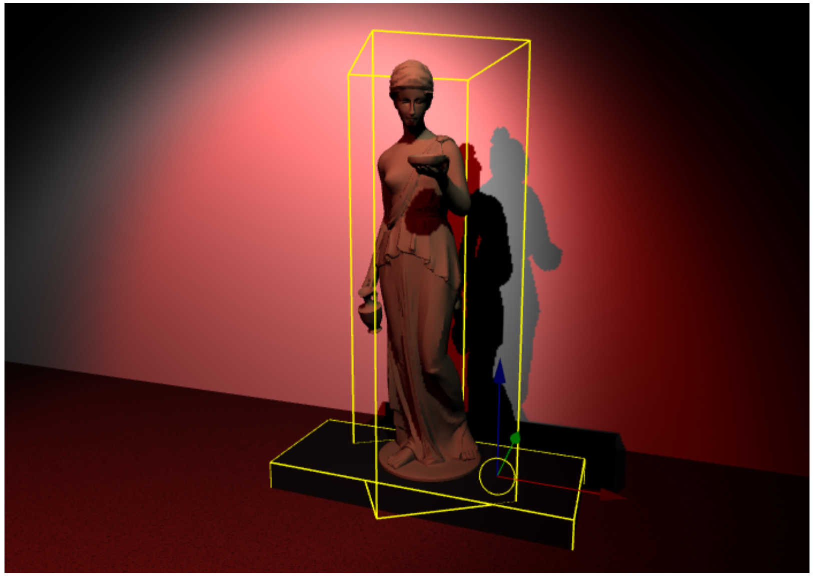
- HTML-Graph == semantic scene graph
- Memory Data Structures for display == rendering scene graph

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

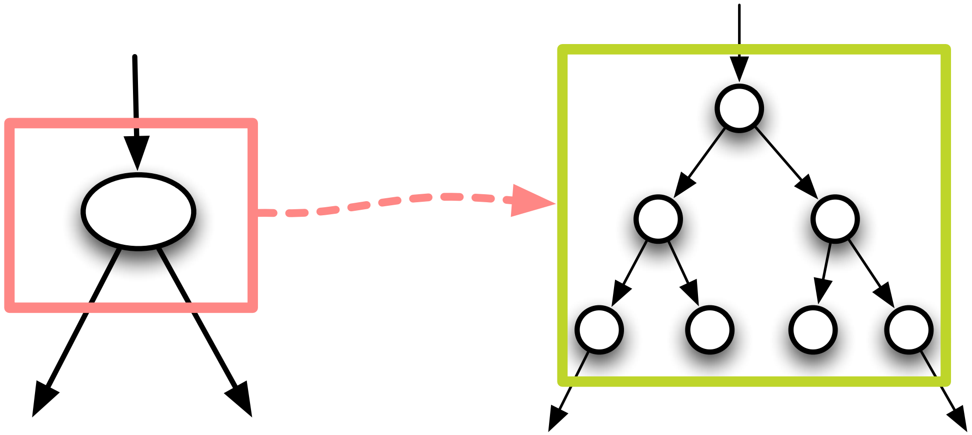
So what's left to do?

- 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/