

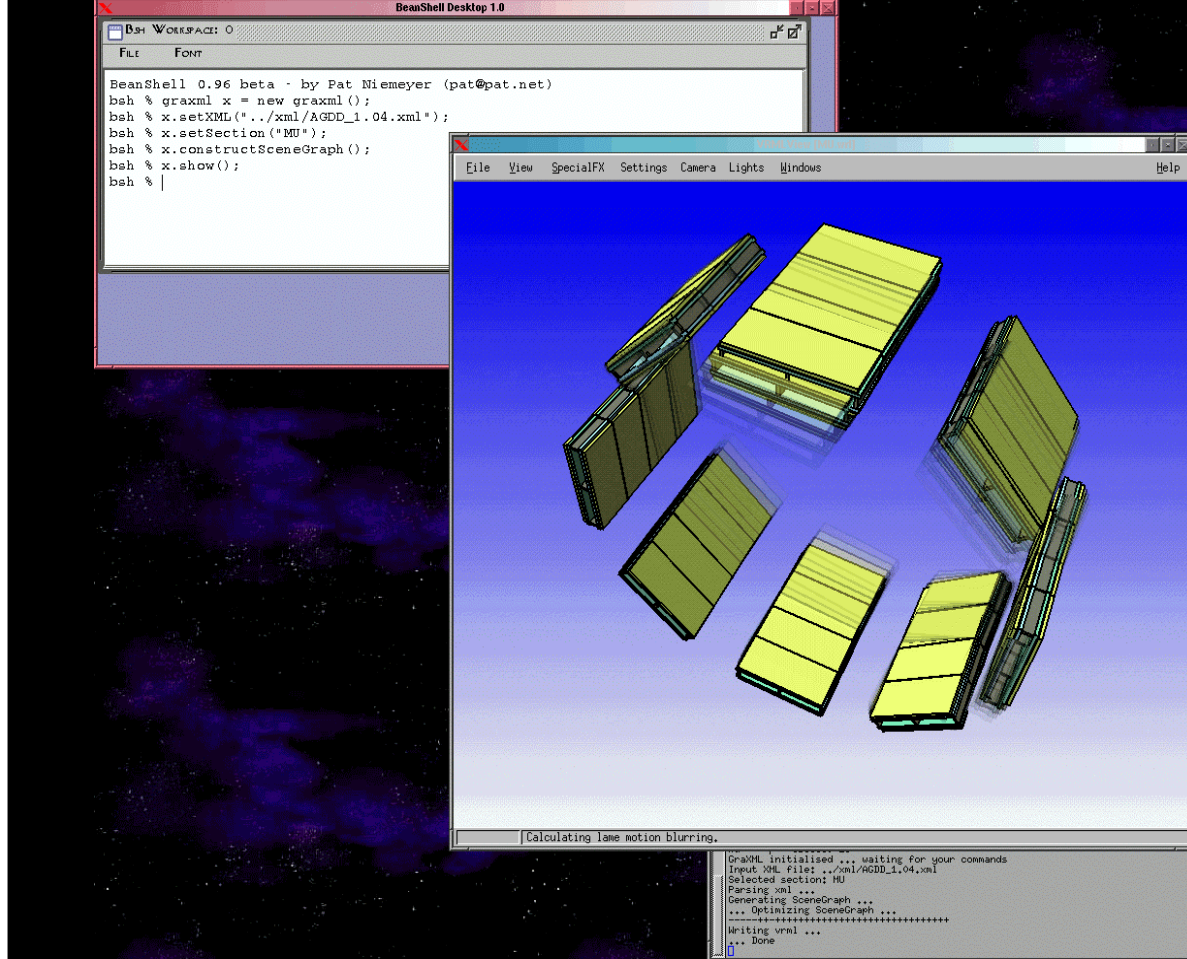
GraXML

1

work has started cca 3 weeks ago

aim:

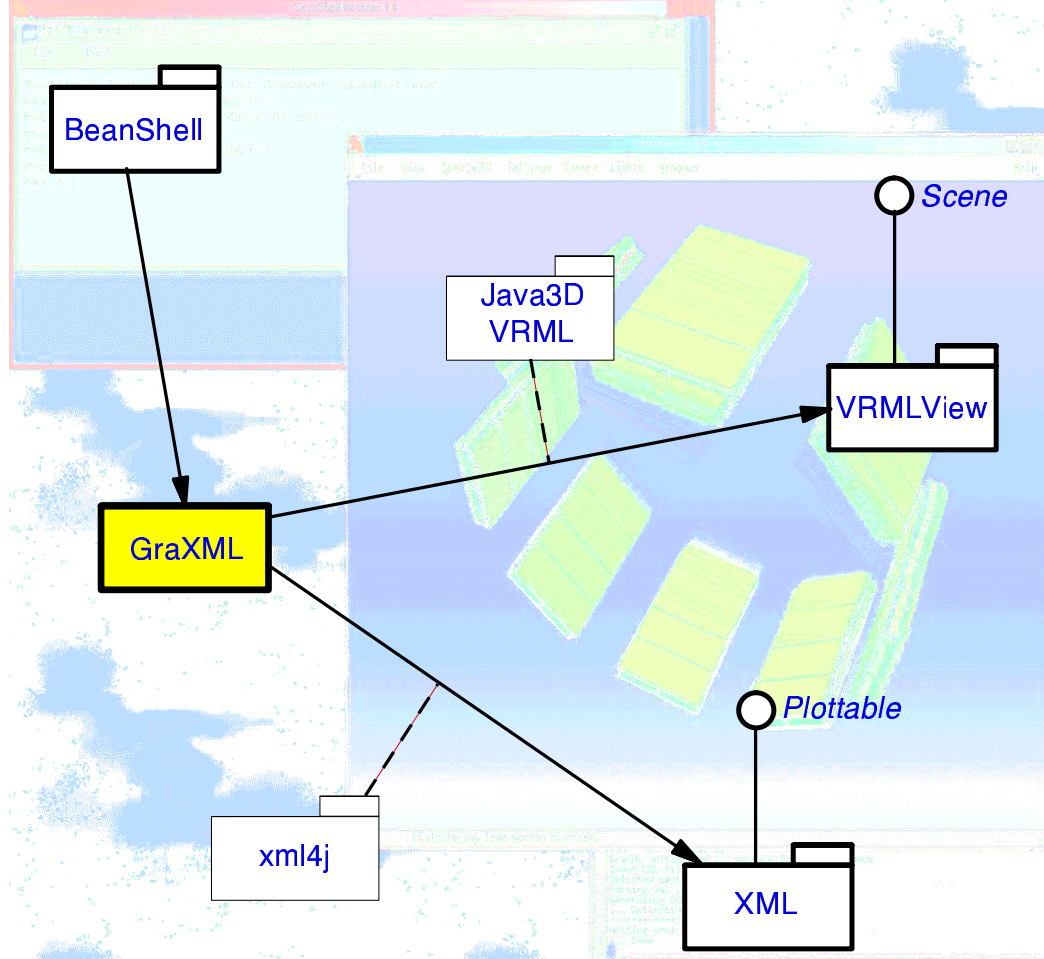
- * standalone independent Detector Display
- * evaluation (+ learning) of Java



2

how it looks:

- * 3D view with visual operations and rendering options
- * script interface (or batch) in 100% Java

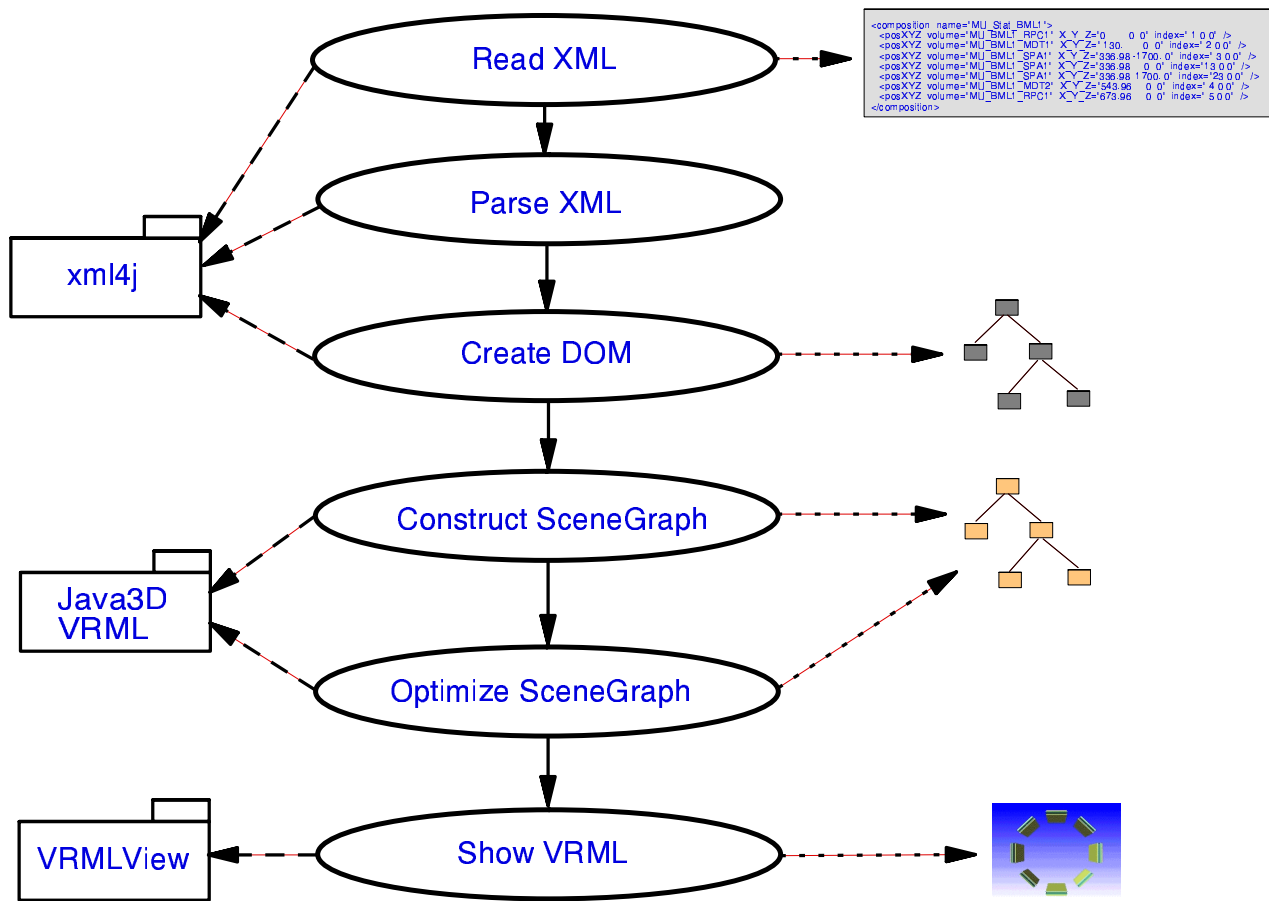


3

reuse as much as possible

- * only GraXML is written and it has about 600 loc (mostly geometrical algorithms)
- * Bean Shell is 100% Java scripting + command-line interface (a'la Cint, but almost trivial in Java), used by Wired, in Asis - 2 other candidates known
- * xml4j is Java version of xml4c - many other candidates known
- * Java3D is part of Java 2 (alpha^2 on Linux), VRML can created VRML SceneGraph - other candidates known

Plottable + Scene is standard Architecture used in Atlas Graphics

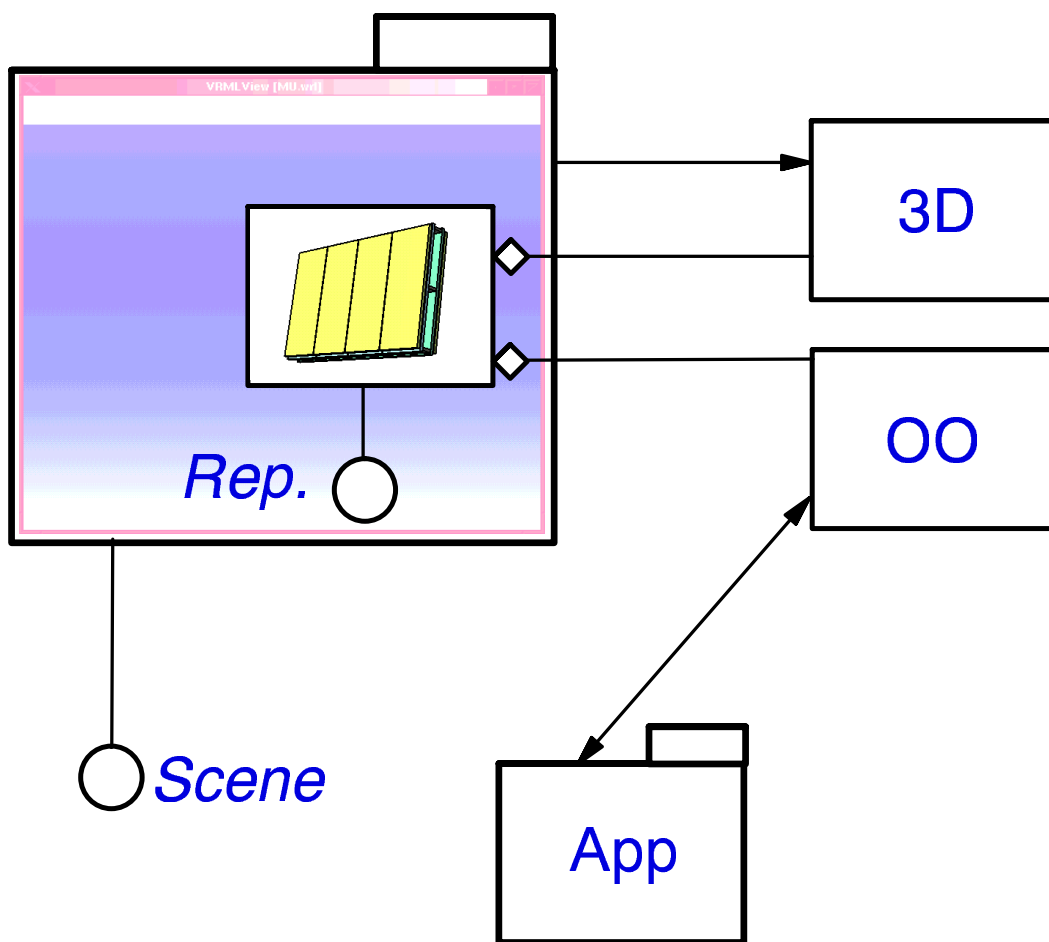


4

data flow diagram is not OO, but usefull

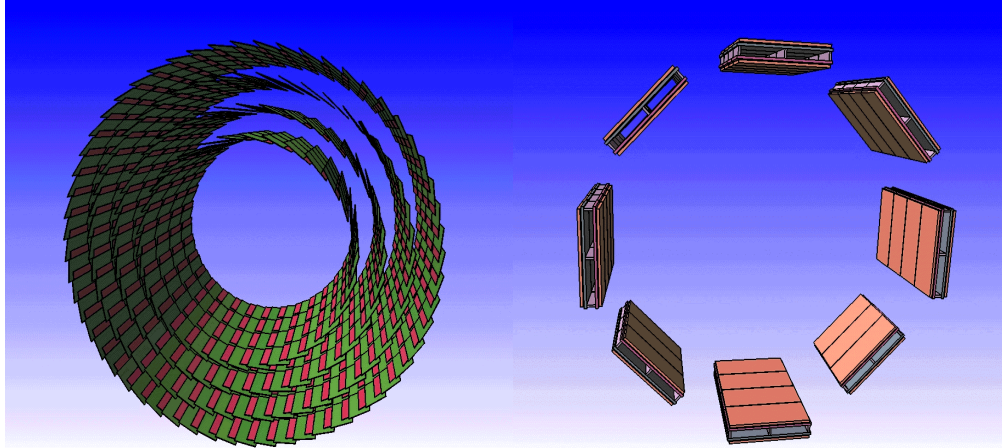
basic philosophy is to create one Tree (XML-DOM) and convert is to another Tree (VRML-SceneGraph)

DOM Tree is unexpanded, so it fits well into memory; by making expansion one looses information about structure, which is needed for SceneGraph optimization

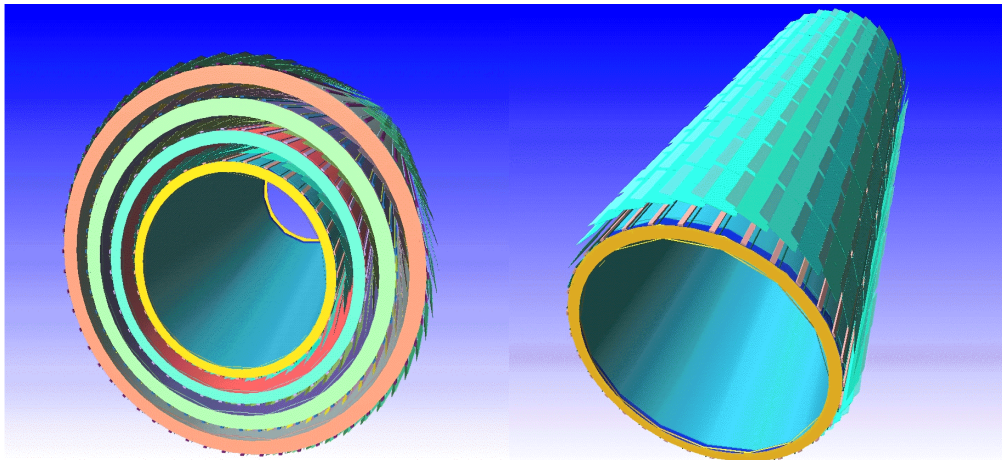


5

- # VRML Representation within VRML Scene has two components:
- * VRML itself, which talks to the Scene
 - * Java Object, which has access to VRML attributes and to any other Application - it can give behaviour and interactions



- * Java
- * Active 3D Objects
- * Scripting interface
- * Standalone
- * Small
- * Functional
- * Reuse of Modules
- * Platform neutral
- * Extensible
- * Open Source



6

- # same pictures as G4 and Persint -> proof of consistency (but some hidden assumptions about geometry in DTD)
- # different rendering options (hidden lines, wireframe, ...) for different pictures
- # after one afternoon first pictures, after one equivalent-week functional prototype
- # implements only what is in XML (implementation on demand)
- # quite ugly implementation (needs re-design)
- # very easy task due to the proper choice of technology
- # it would be possible to do that in C++ (Open Inventor), but cca 10x more complicated
- # many features are automatic, while in C++ they would make big task (scripting, interaction with outside,...)
- # source is freely available
- # executable is 20kB small
- # future:
 - * integrate into Atlas Graphics, add interactivity
 - * what about Wired, JAS