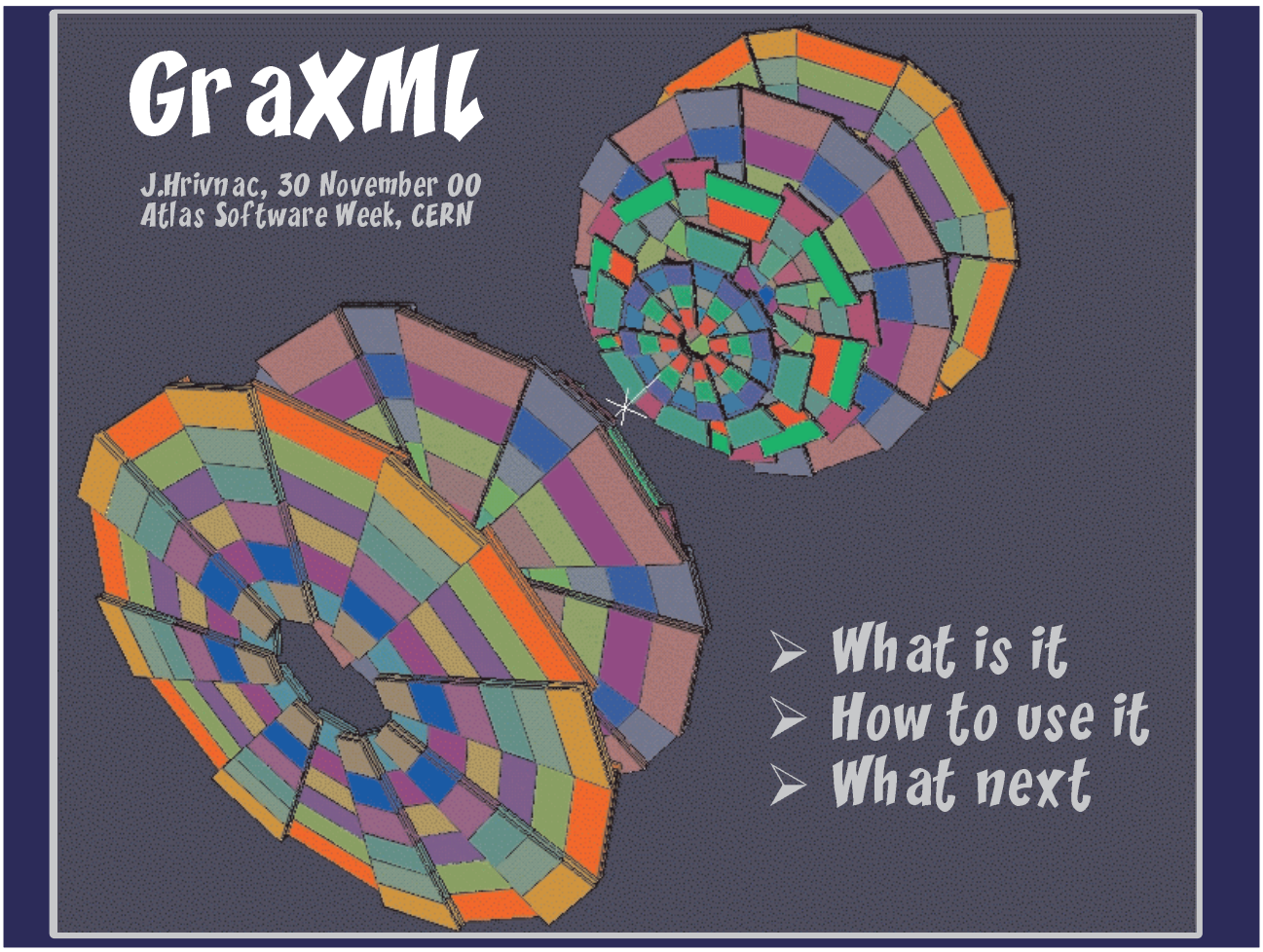
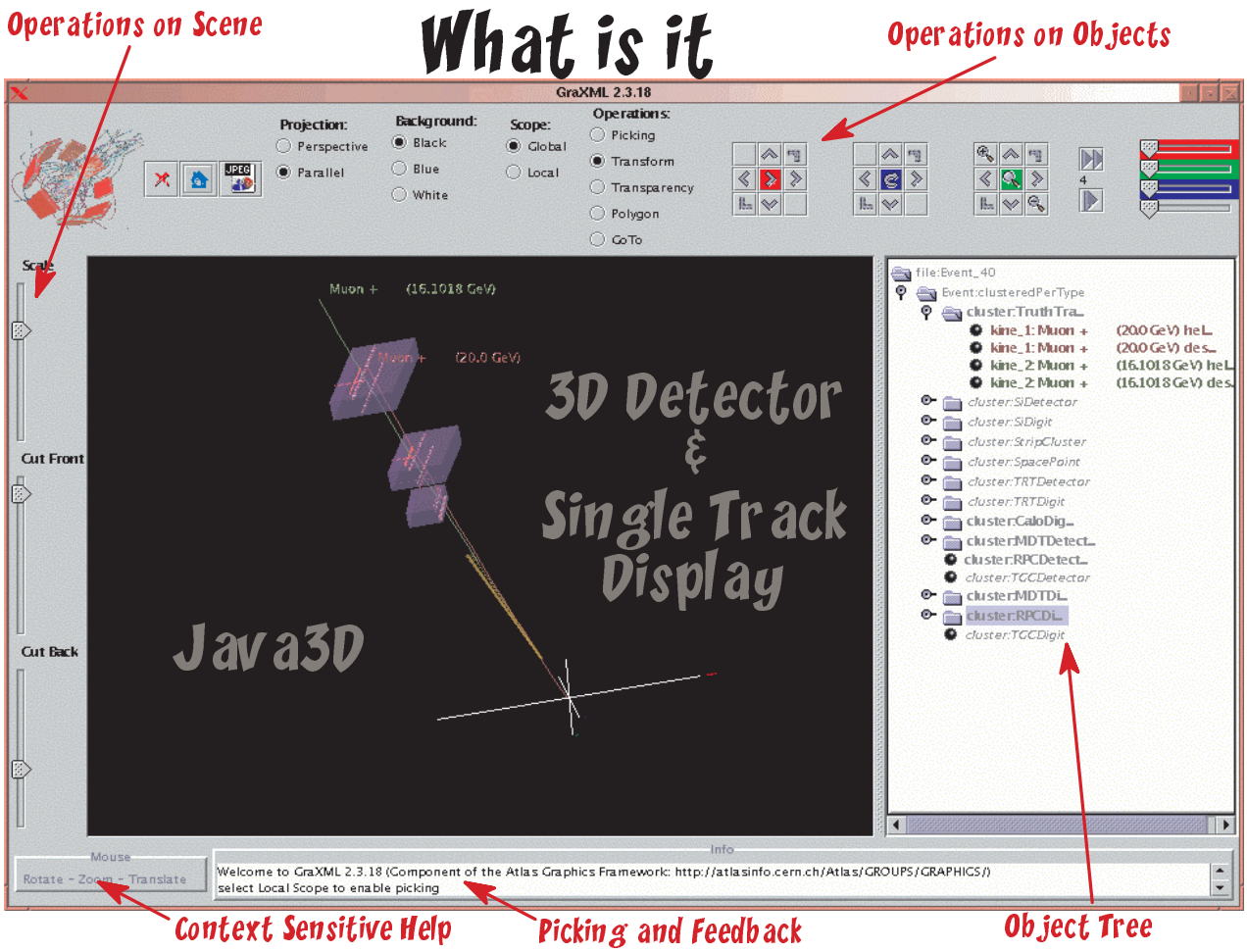


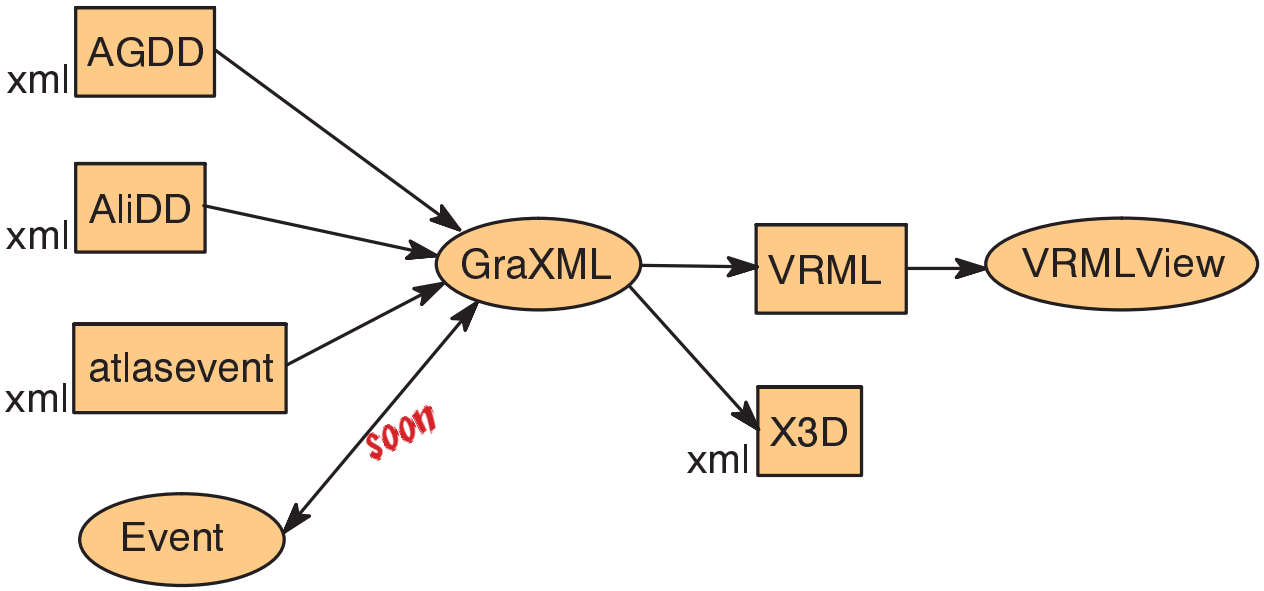
1

- one year ago started as evaluation of Java & 3D technology for Detector Description
- evolved into full featured product
- used as testbed for "playing" with 3D (Java3D, VRML, X3D) application => one can expect changes (mainly to GUI)
- executable + source + documentation < half MB
- profits from Java enormously



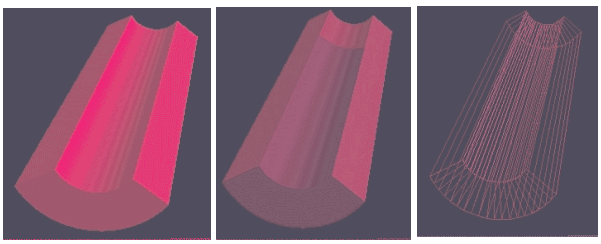
- don't try to give it the whole Event (no realistic 3D Display could do that today)
- some parts of menu are dynamic

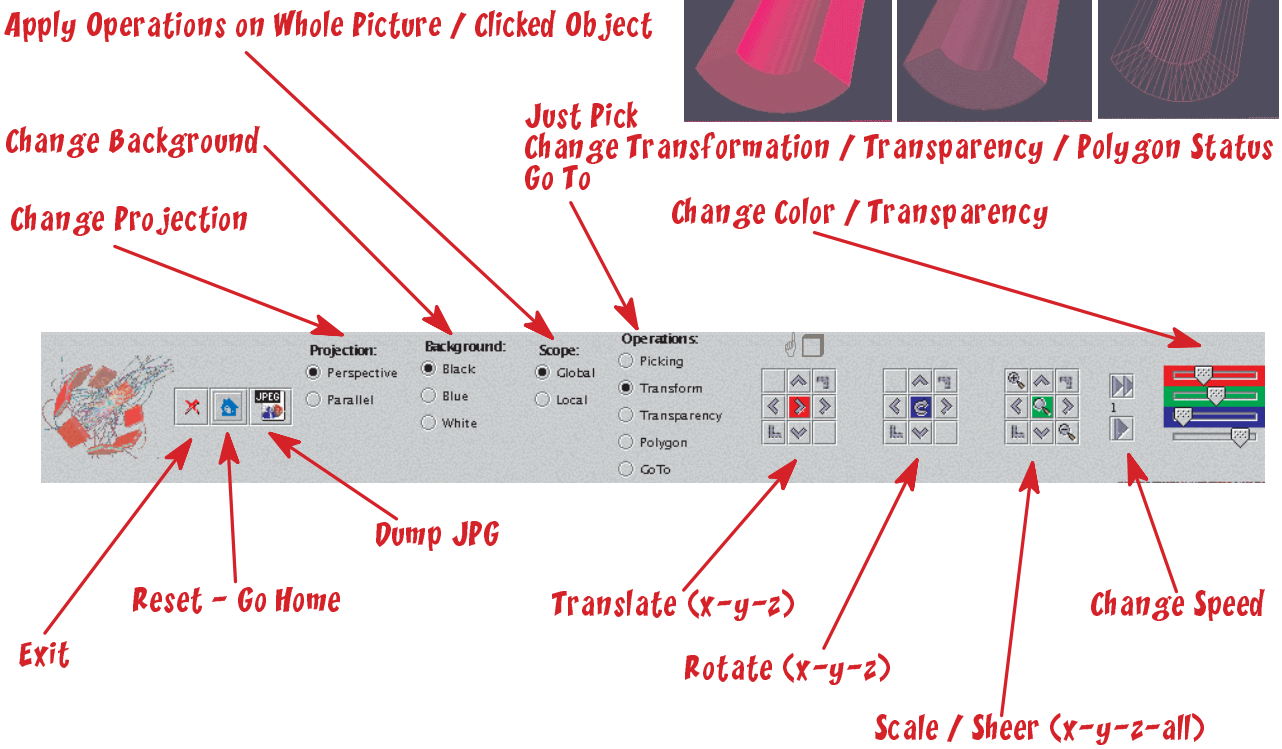
Flow



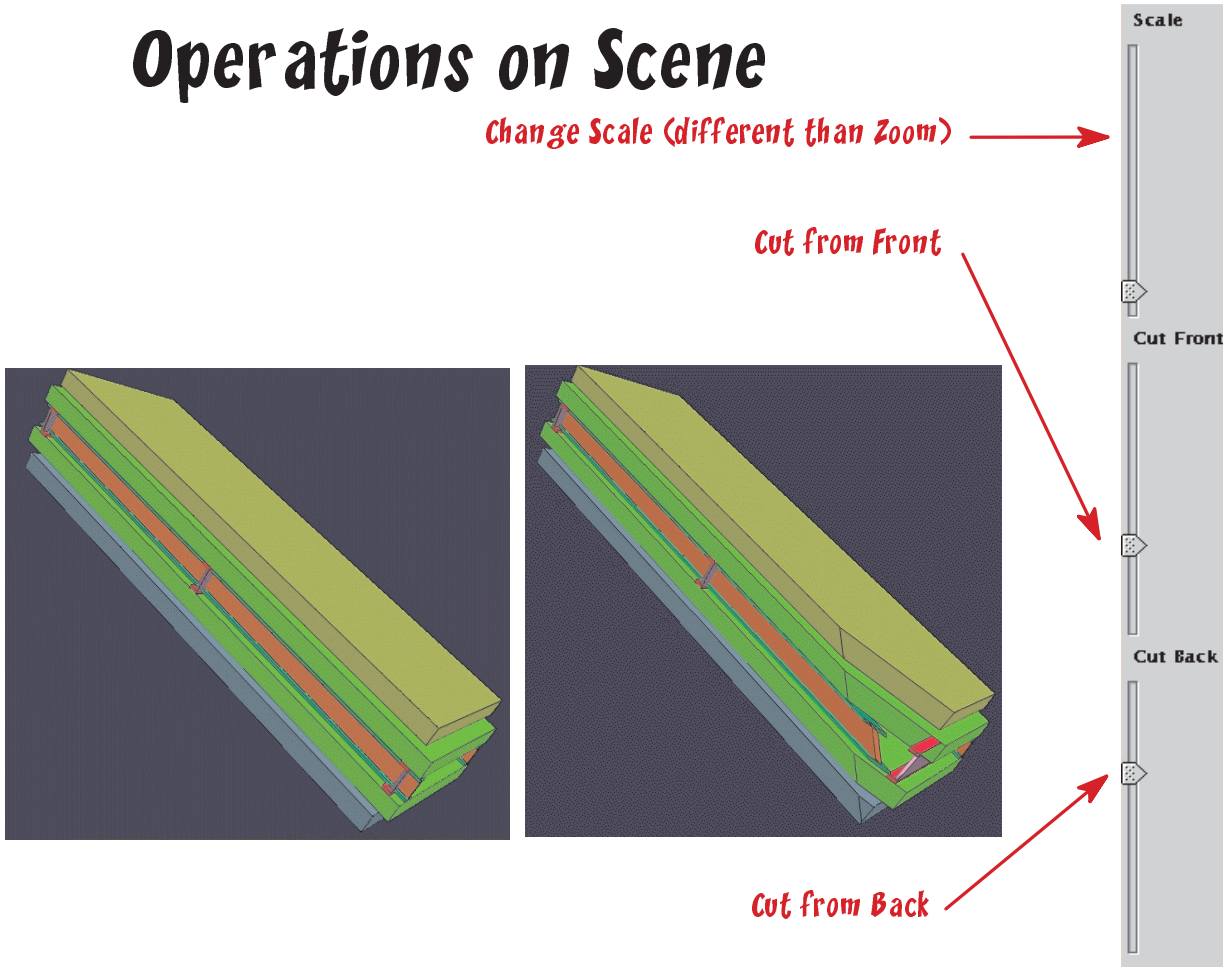
- can read several kinds (dtds) of XML files for both Detector Description and Event
- AliDD is extention of AGDD used in Alice (general Transformation and additional Geant4 elements are added)
- VRML can be read by any standard VRML 2.0 browser (e.g. vrmlview or vrwave on Linux)
- X3D is the future standard of VRML using XML

Operations on Objects





Operations on Scene



- Scaling much faster than Zooming
- Scaling to be used in Parallel Projection (it doesn't do anything usefull in Perspective Projection, while Zooming doesn't do anything usefull in Parallel projection)

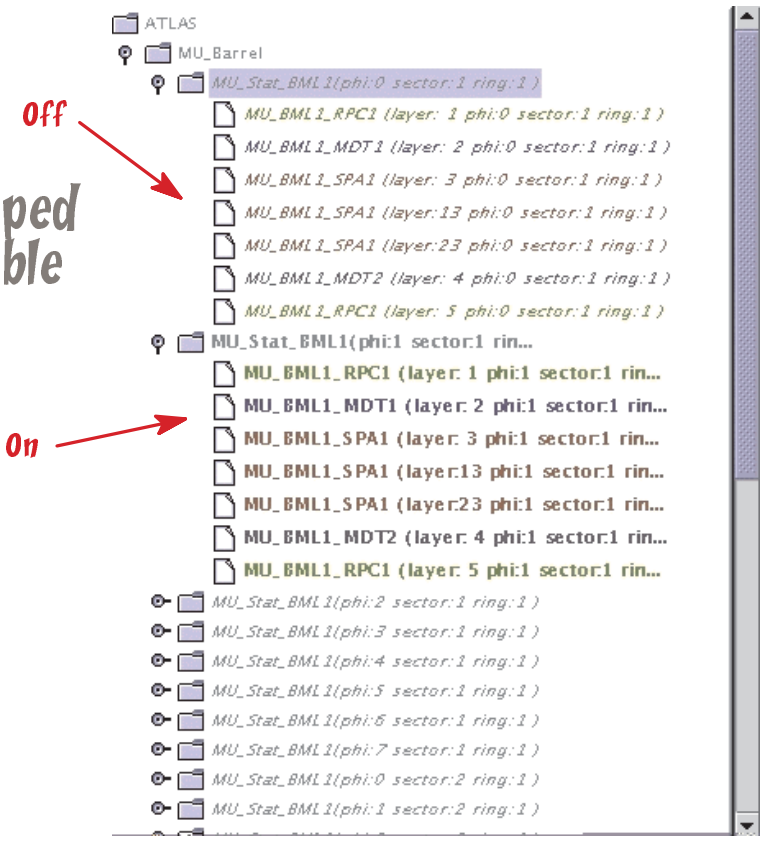
Other Operations

- Transformations (Translate-Rotate-Zoom):
  - Mouse
  - Keybord
  - Menu

- all three way behave in different ways
- Menu based Transformations usefull for big Scenes

Tree View

- Tree of element maped into Tree of breakable connections
- Allows switching On/Off
- Uses identifiers
- Uses object' colors
- Can allow other operations



- identifiers not yet fully debugged

Modes

- Interactive [default]: `-i | -n`
  - Access to per-Object features
  - Slower & bigger
- Optimised (compressed): `-o`
  - Equivalent Objects as references
  - Faster & Smaller
  - Objects can't be always switched off (as they may have hidden references)
  - some Local Interactivity is lost (as Objects may be just references to other Objects)
- Browser: `-b [vrmlview|xml|vrml|...]`
  - External output or browser
- Quality (1-9, default = 5) `-q <quality>`
  - More Quality means more memory and CPU, but nicer picture
- Event Rearangement `[-atlasevent,perType|-alasevent,perKind]`
  - is reflected in Tree View

Selectable at start-time.

- the way how the SceneGraph is build and shown
- Optimised mode can bring several orders of magnitude in CPU and memory needs, but some Interactivity is lost
- Quality changes granularity of curved shapes, antialiasing,...

Options

- What to display:
  - Objects contributing into the SceneGraph
- How to display:
  - Representation to use (asLine, asBox, ...)
  - Cuts, colors, dimensions,... to use
- How to render:
  - Wireframe, Outline, Antialiasing,...
- How to optimise SceneGraph

Currently selectable at compile-time,
will use GUI / XML Options Files / Command-line.

- large amount of Options
- some Options handled by Quality parameter

Context Sensitive Actions soon

- 3D Rep Objects are alive and carry references to their Plottable origins
- Context sensitive Popup Menu
- Will use **JACO** access to C++
- Authors of original objects should define context sensitive actions (via Model)

- not yet fully implemented as meaningless without access to real Objects

What next

- GUI
- Performance
- Missing pieces
- Connection with Framework
- Modularisation & Moving to FreeHEP/HEPStudio:
  - 3D Solids done
  - Interacters
  - Tree Navigator/Selector
  - VRML Writer

Currently GraXML is used mainly as a playground
for developing code based on Java3D.
It is usable at the same time.

- 3D Solids: Java3D implementation of Geant4 Solids (all primitive solids and some others are implemented)
- Interacter: adds local interactivity to any Java3D BranchGroup
- Tree Navigator/Selector: breaks Java3D SceneGraph into detachable pieces and creates JTree to operate on them
- VRML Writer: originaly from CyberVRML package, heavily debugged and extended, the only know package which is able to write Java3D into VRML