Updates on Visualization

Laurent Garnier. IRISA
Guy Barrand IJCLab
John Allison G4AI
Frederick JONES TRIUMPH

10.7 Release status

- **OpenGL drivers:**
  - Study code signing / notarizing issues for using OpenGL from Geant4-based applications on MacOS 10.15 Catalina
  - Improvements to toolbar in OpenGL Qt -
  - Adapt to newer OpenGL versions, exploit new functionalities and replace deprecated calls such as glBegin/glEnd - O
    - i. => Investigate a way to switch from OpenGL to other thing
    - **Guy : Apple/Metal at the Orsay forge**
      John : A Qt3D visualisation driver for Geant4

  - OGLFile to produce image files in batch jobs where no graphics card is present - O
- **Other drivers:**
  - Wt driver: Remove code
  - Implication of the Qt license changes

Fred: Progress on the Open Inventor Qt Visualization Driver
Extra slide : John : Debugging the geometry of the human phantom

Apple/Metal at the Orsay forge

Geant4 collaboration meeting Sep 2020

G.Barrand, CNRS/IN2P3/IJCLAB

2018 :-(

- WWDC June 2018 : Apple, in a // session, announced that the Apple/ OpenGL is deprecated.
- Bad news for people looking for a standard to do visualisation.
- Bad new for me and Geant4, and a lot of scientific software.
- Due to the impact of Apple concerning interactivity, we can’t ignore that…
- Apple promotes their proprietary Metal in remplacement of OpenGL on their devices. We have to look!

- (No date given about a strong removal of Apple/OpenGL on macOS and iOS)

inlib/exlib/sg scene graph logic

- In spirit, same scene graph logic as the great OpenInventor.
- A graph is rendered on screen (or offscreen!) by using an implementation of a “renderer” for a given technology, for example OpenGL.
- See softinex at http://gbarrand.github.io
- g4tools/plotting done with that by using some offscreen renderers.
- GL-ES renderer. It permits (today) with SAME CODE to visualise on Linux, macOS, Windows, iOS, Android.
- Then I have to provide a renderer for Apple/Metal…

Not so easy to do !

- API is in Objective-C or in Swift.
- Apple examples are in Swift buildable from Xcode.
- Nothing in C++ buildable from a “simple make”.
- Stucked…….up to the end of June 2020
- Some googling gave a hit on GitHub : naleksiev/mtlpp which is a C++ wrapper around Metal with an example to draw a triangle buildable with make : bingo!

Summer 2020 at the forge…

- After two months of very painful coding, I have now one C++ app (a display for ESSnu) that works on macOS by using Cocoa and Metal.
- And this by using straight the Objective-C Metal API from C++ (Apple clang permits to mix both languages).
- Painful because the logic of Metal is not similar than GL-ES (even if ideas of rendering pipline, buffers, etc… are the same). We have to rethink a new renderer (which was not the case for offscreen or WebGL ones).

Summer 2020 at the forge… (2)

- I have correct 3D rendering for basic primitives (points, lines, segments, triangles, triangle-fan and strip).
- I have lighting.
- I have texture mapping.
- With that I can have my apps working on Metal.

And be sure it had not be easy to get !

Can it help for Geant4 ?

- My R&D apps g4exa, g4view should run with Metal and I am going to release versions of these.
- But it is not based on the "G4 vis system" largely used now.
- The G4 vis system is in principle designed to handle multiple heterogenous graphics systems (= drivers).
- For example there is an OpenInventor driver and some offscreen ones (HepRep, VRML).

Can it help for Geant4 ? (2)

- Right now what is promoted by Geant4 is Qt for the GUI and OpenGL for the graphics.
- Qt comes now with Qt3D to do 3D rendering. Qt people are going probably to provide an Apple/Metal version of it. J.Allison made progresses around a G4 Qt3D vis driver.
- Anyway, I would strongly suggest to G4 to not put all its eggs in the same Qt-basket and still maintain an “academic way” to do GUI and graphics. If so, the effort done at Orsay may help in handling Metal for such G4 vis driver.
- (A g4tools (= part of inlib/exlib) G4 vis driver ? Not yet mature, but the idea makes its way…)

The X11 way on Macs

- Anyway now X11, OpenMotif, an X11 server, OpenGL/ Mesa, are available in a consistent way (for exemple through MacPorts) on macOS, then we can always run on Macs this way without any GUI and graphics Apple software.
- (Avoid to mix with XQuartz libs !).
- (Would Qt/X11 with Mesa/GL be running on these ?)

- (In fact the same is true on Windows by using CYGWIN).

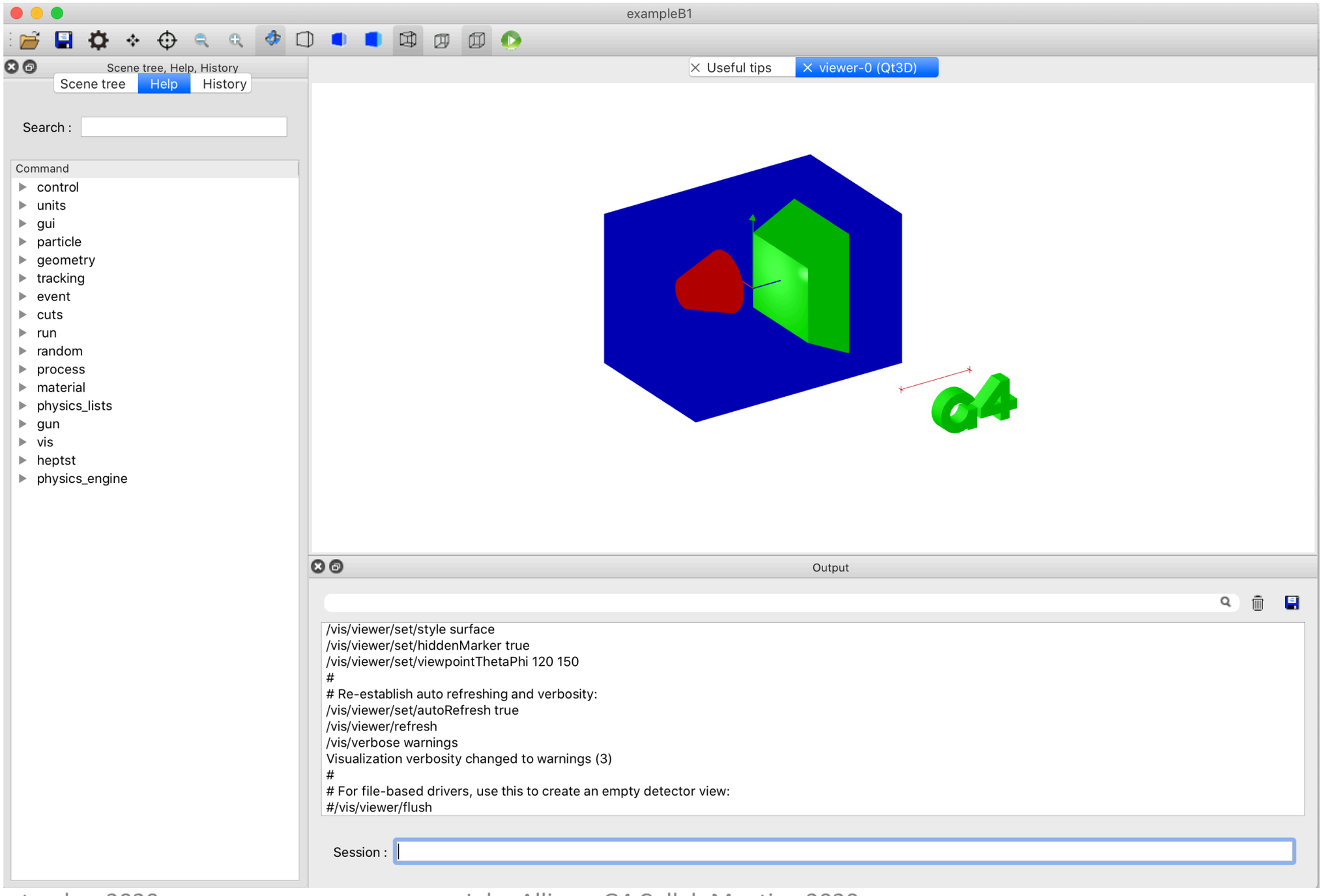
A Qt3D visualisation driver for Geant4

Qt3D

- Qt3D is a new graphical interface within the Qt project
  - Qt is free to open source projects
    - You have to register
  - From https://www.qt.io/blog/2016/06/16/introducing-qt-3d:
    - "The Qt3DRender module is a high-level interface to hardware accelerated graphics. At present Qt 3D uses an OpenGL backend but we have left the door open to be able to support more modern APIs such as Vulkan, Metal and DirectX 12 in the future."
  - Qt3D is still “under development”

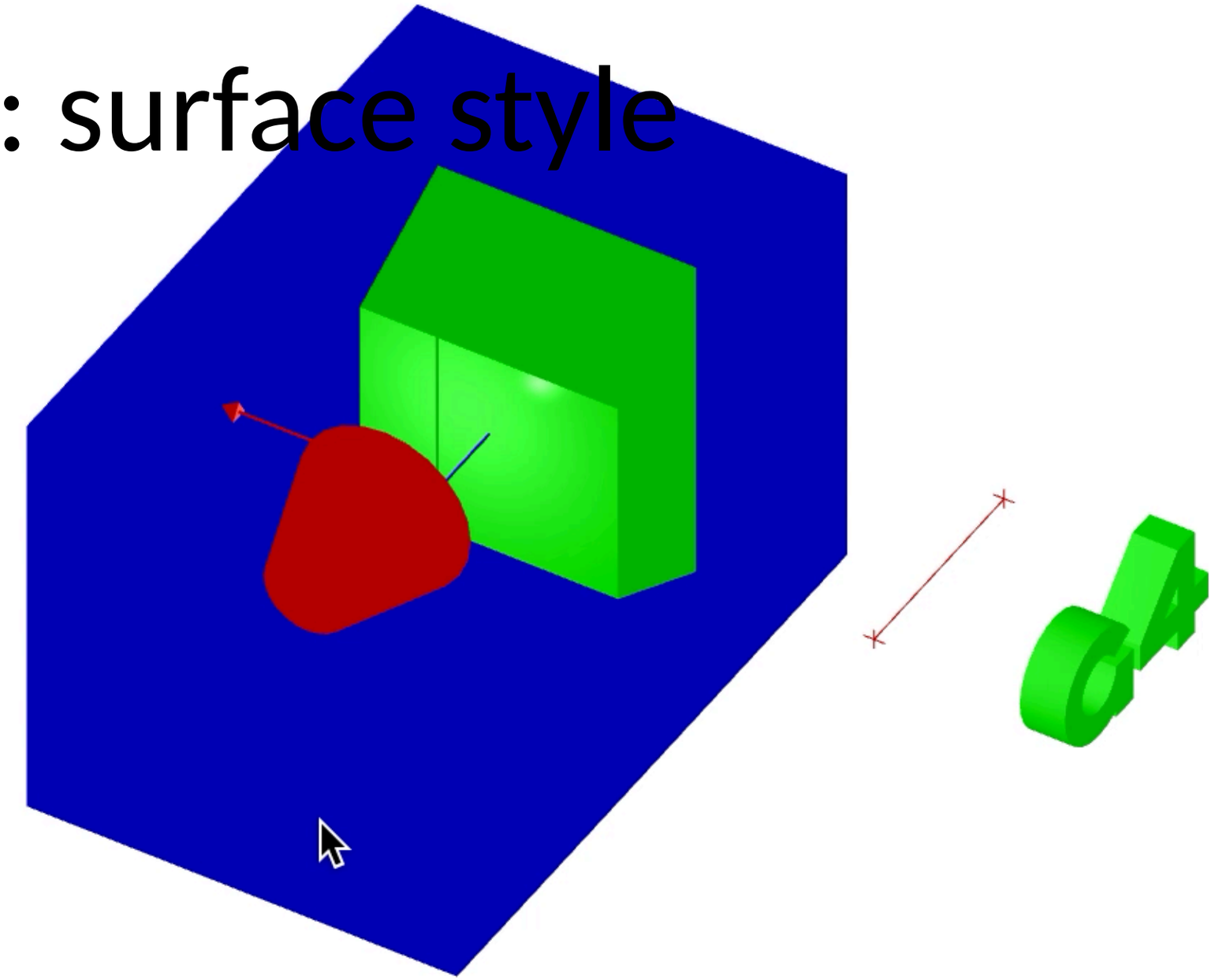
Experience so far

- Qt3D is pretty “low-level”
  - Documentation sparse; programming tough
  - It does not seem to have the very low level control of OpenGL
    - For example, I don’t see how to change line width
    - Or switch z-buffering off
    - But I guess it has to go for the highest common factor of all systems
- Geant4 Qt3D driver
  - Wireframe and surface drawing modes look OK
    - Hidden edge is weird
  - It doesn’t seem to do transparency very well
  - Trajectory drawing only works in sequential mode
    - I have not implemented changing threads yet
    - Markers (e.g., trajectory points) not yet implemented
  - Performance is good
  - In my personal repository, together with some examples
    - Until adopted you have to point to the G4visQt3D library and explicitly register it with the vis manager

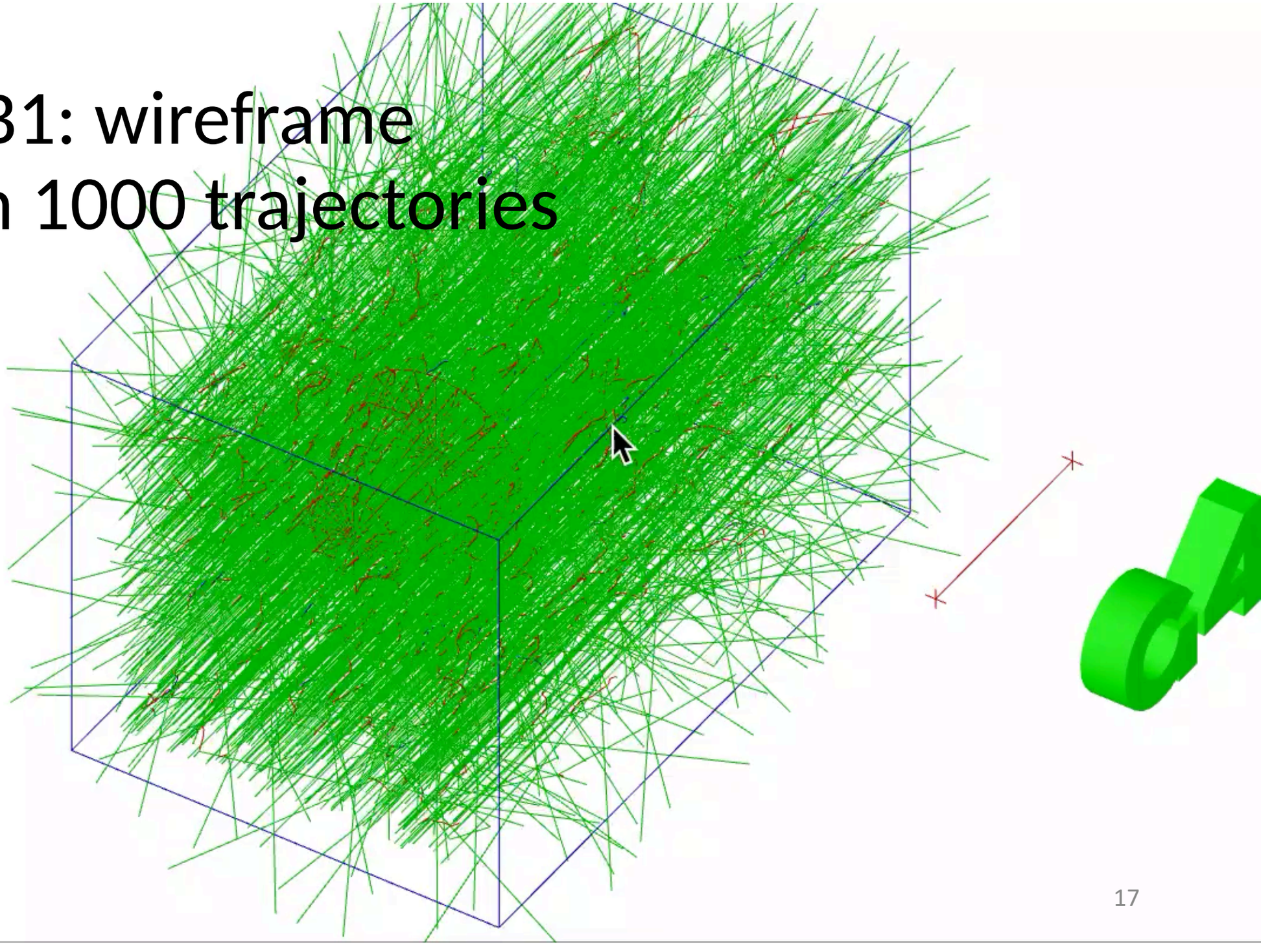

exampleB1
Scene tree, Help, History
Scene tree
Help
History
Search :
Command
control
units
gui
particle
geometry
tracking
event
cuts
run
random
process
material
physics_lists
gun
vis
heptst
physics_engine
Useful tips
viewer-0 (Qt3D)
Output
/vis/viewer/set/style surface
/vis/viewer/set/hiddenMarker true
/vis/viewer/set/viewpointThetaPhi 120 150
#
Re-establish auto refreshing and verbosity:
/vis/viewer/set/autoRefresh true
/vis/viewer/refresh
/vis/verbose warnings
Visualization verbosity changed to warnings (3)
#
For file-based drivers, use this to create an empty detector view:
#/vis/viewer/flush
Session :

exampleB1: surface style

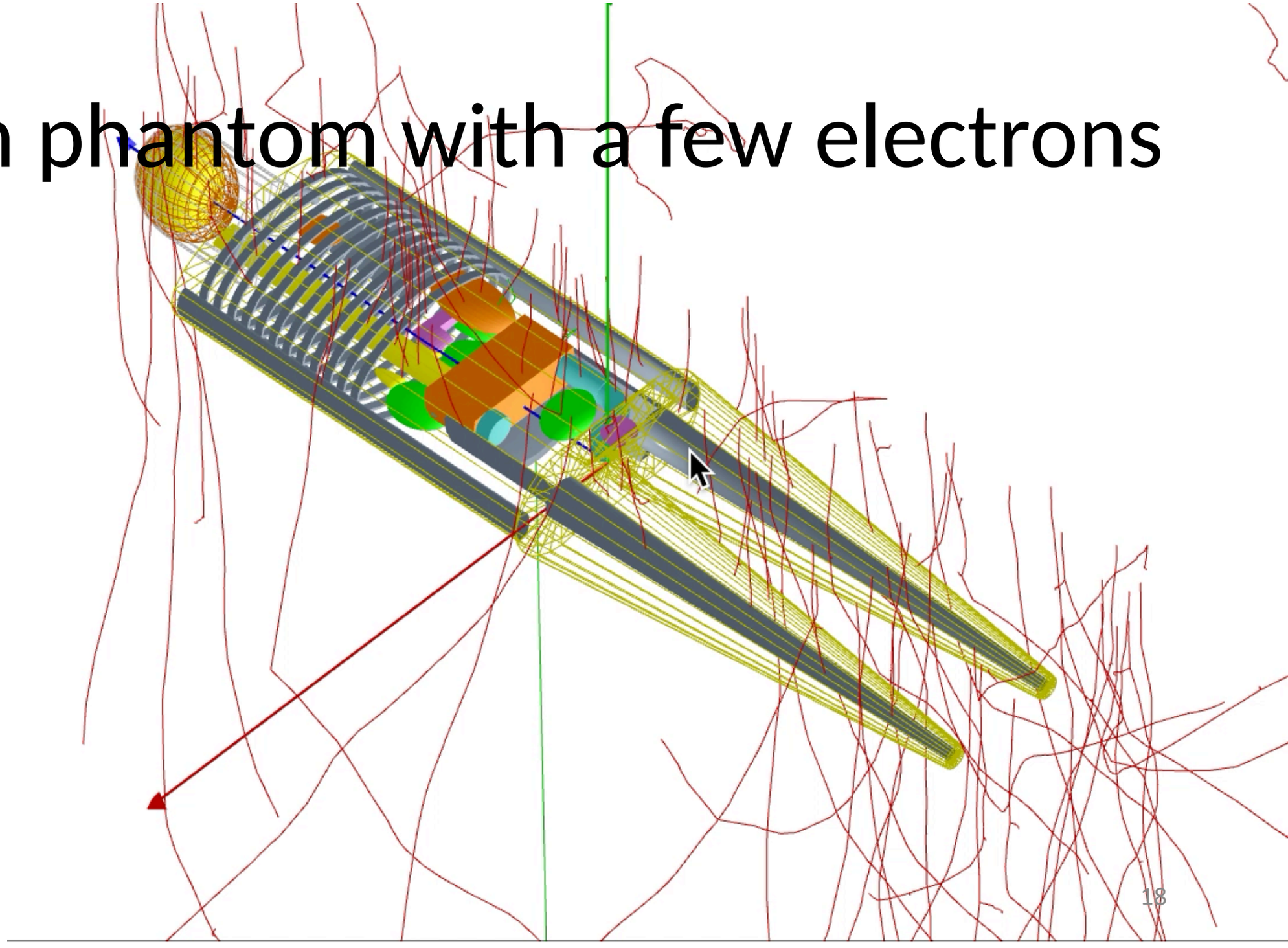
The blue “envelope” is supposed to be semi-transparent but it’s more like “see-through”



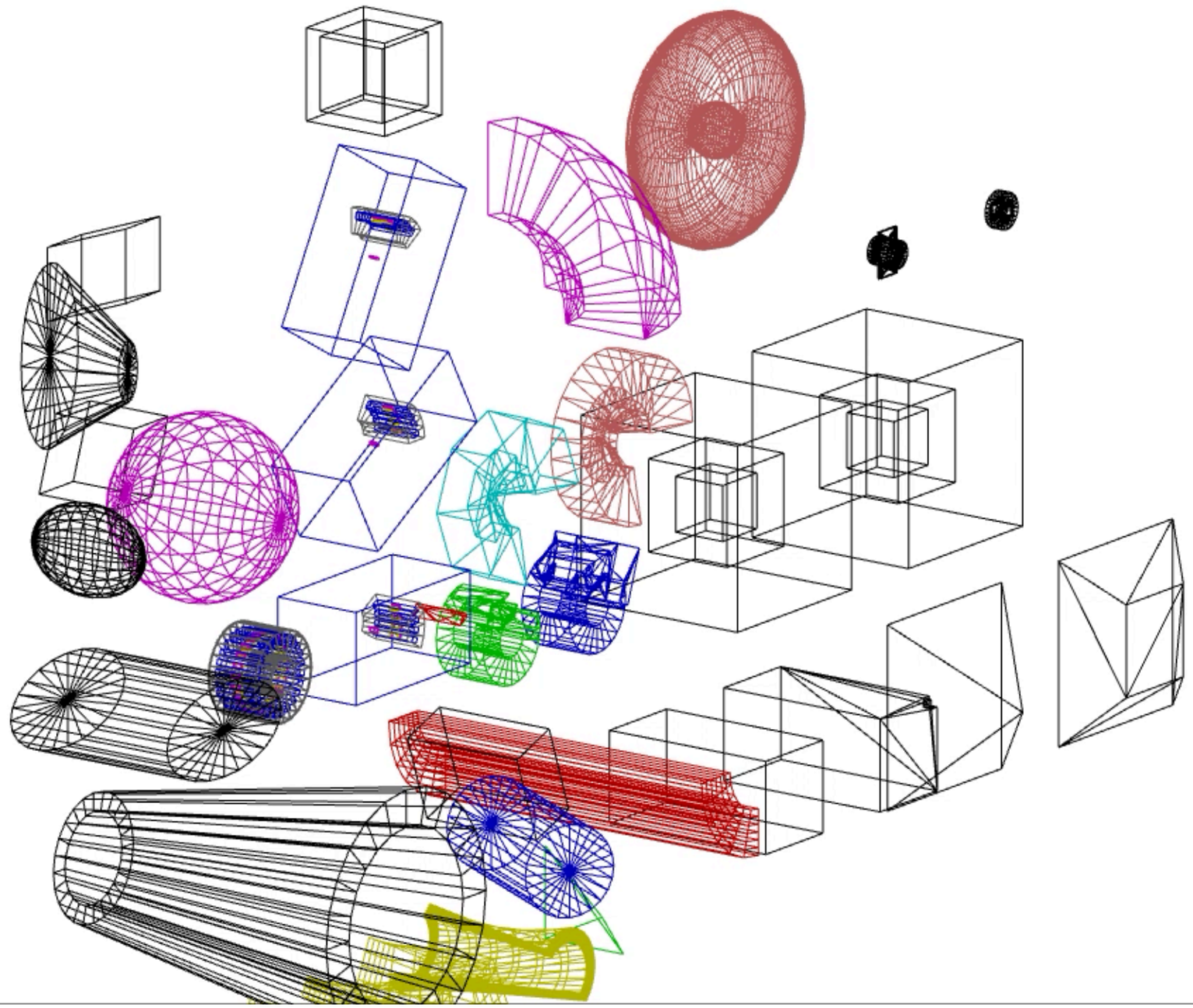
exampleB1: wireframe style with 1000 trajectories



Human phantom with a few electrons



test202

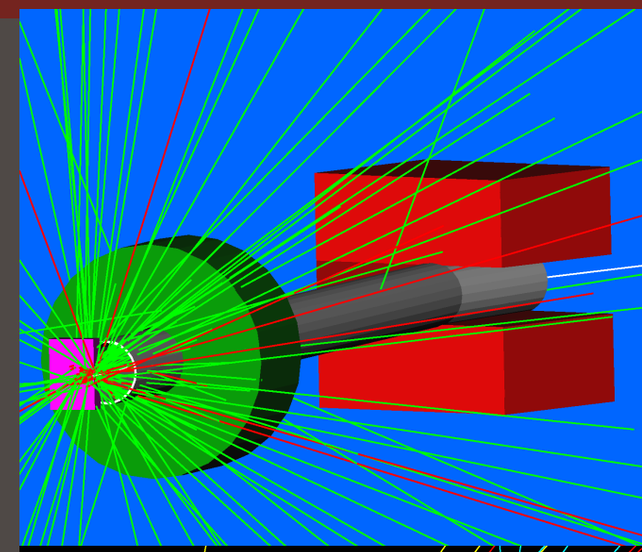


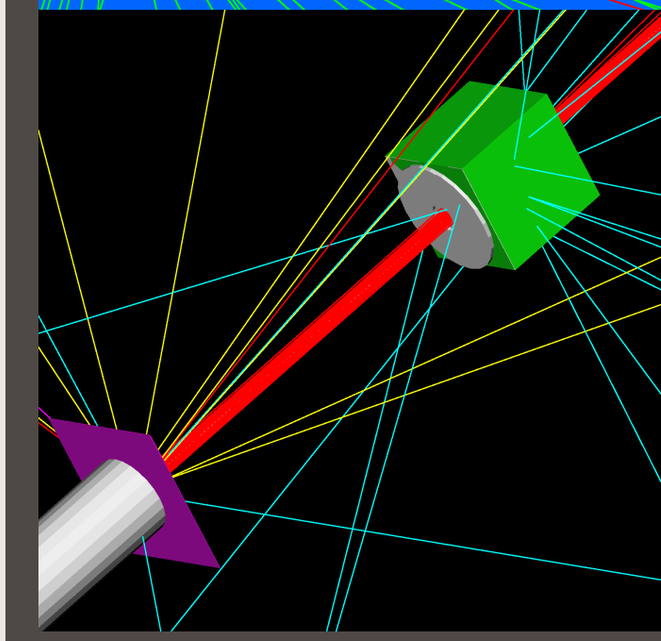
Summary

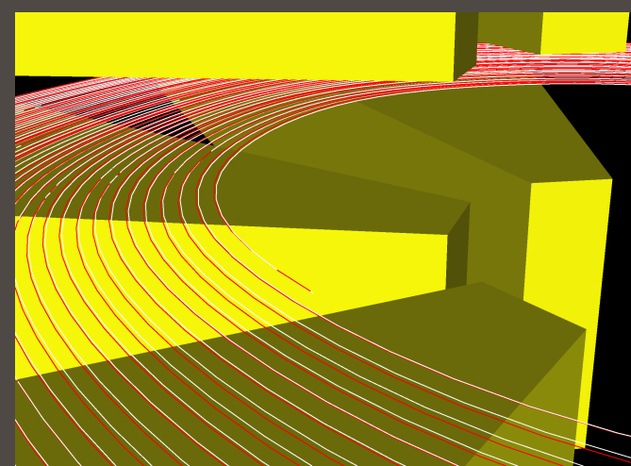
- A promising new driver
- Offers independence from evolution of graphics interfaces on all platforms

TRIUMF

Canada's national laboratory for particle and nuclear physics
Laboratoire national canadien pour la recherche en physique nucléaire
et en physique des particules





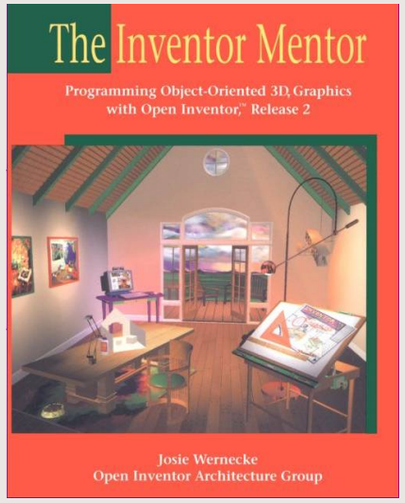


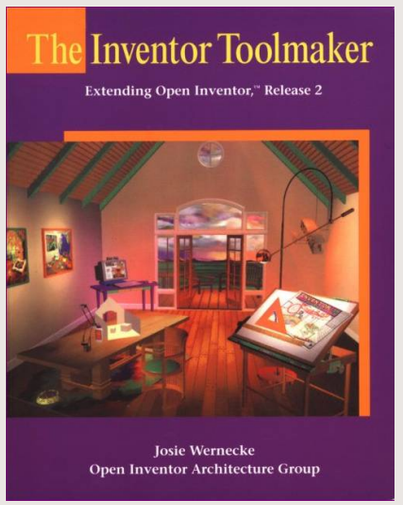
Progress on the Open Inventor Qt Visualization Driver

Frederick Jones, TRIUMF

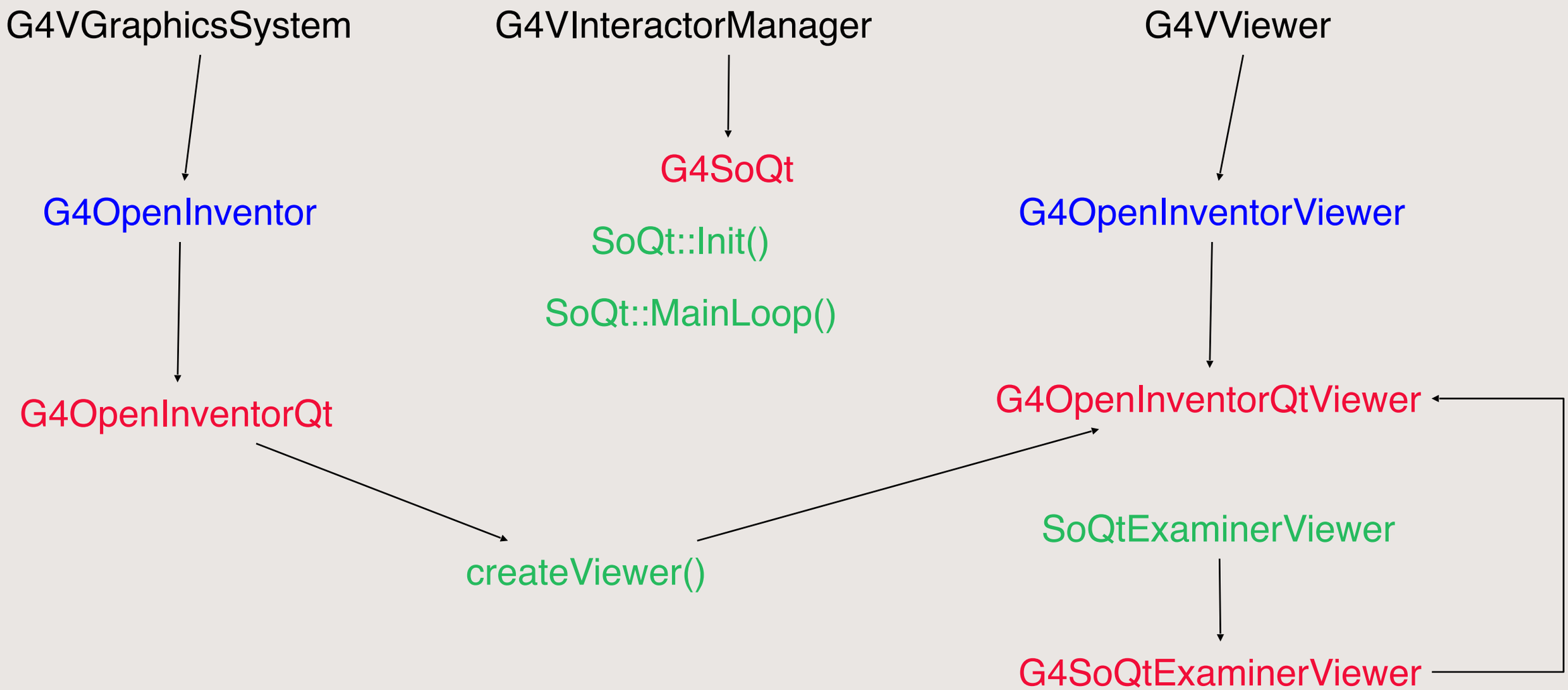
Background: Open Inventor Vis Drivers

- The Open Inventor (OI) drivers were first developed in 1996-98 principally by J.Kallenbach, G.Barrand and J.Boudreau.
- After Silicon Graphics developed OpenGL, Open Inventor was created both as a higher-level API for OpenGL applications and as an advanced 3D graphical display and management system, using OpenGL as the rendering layer.
- Some advantages of OI over basic OpenGL:
  - OI provides powerful viewers with effective interactive controls and many built-in functions. The view is continuously relocatable via the “Seek” function. Full support for animated displays, transparency, and other effects.
  - Scene graph technology allows great flexibility including overlayed 2D or 3D scenes and information retrieval via picking and mouse-hovering.
  - Multi-platform support. Application code is OS-independent. Works with different GUI toolkits: Xt/Motif, Windows, Qt, through specific libraries SoXt, SoWin, and SoQt. Many common “So” methods for writing event-driven viewer functions.
- The existing G4 drivers are based on SoXt (for Unix and MacOS with XQuartz installed) and SoWin.
- The current effort is to produce a driver based on SoQt, initially for Unix and MacOS, and later for Windows, thus potentially a common OI driver for all our supported platforms.





Class Structure for Open Inventor Qt Viewer



Four new classes are required.

Minimal other changes: register OIQt in G4VisExecutive and make G4VInteractorManager::secondaryLoop() virtual.

G4SoQtExaminerViewer is the Qt equivalent of G4SoXtExaminerViewer

OI Driver Developments at TRIUMF

- For our beam physics applications we found the OI viewer to be indispensable because of its interactive controls and ability to easily relocate the view (seek function). But we needed further functionality.
- We started a background project to add the following features:
  - A Bookmark facility to save/restore views and camera parameters, allowing random access to them or sequential “slide shows” of interactive 3D views.
  - Stepwise navigation or animation through the geometry along a trajectory or reference path, with precise view rotations.
  - Superimposed text layer allowing mouse-over readout of solids and trajectories (very useful to obtain specifics of a track).
- The new “extended viewer” Vis Driver (OIXE) was completed in 2015. It relies on the Coin3D implementation of OI which is available for all OS and has been made a requirement for the Vis-OI category.
- Requirement of the current development: the Qt-based driver (OIQt) will contain all the added features of the OIXE driver.
- SoXt to SoQt migration process: Unqualified “So…” calls mostly go over unchanged, but code based on “SoXt…” methods must be rewritten using Qt widgets and event handling. Many new menus, buttons, dialogs and callback functions have been implemented.

Extensions to Open Inventor Viewer

Save bookmark

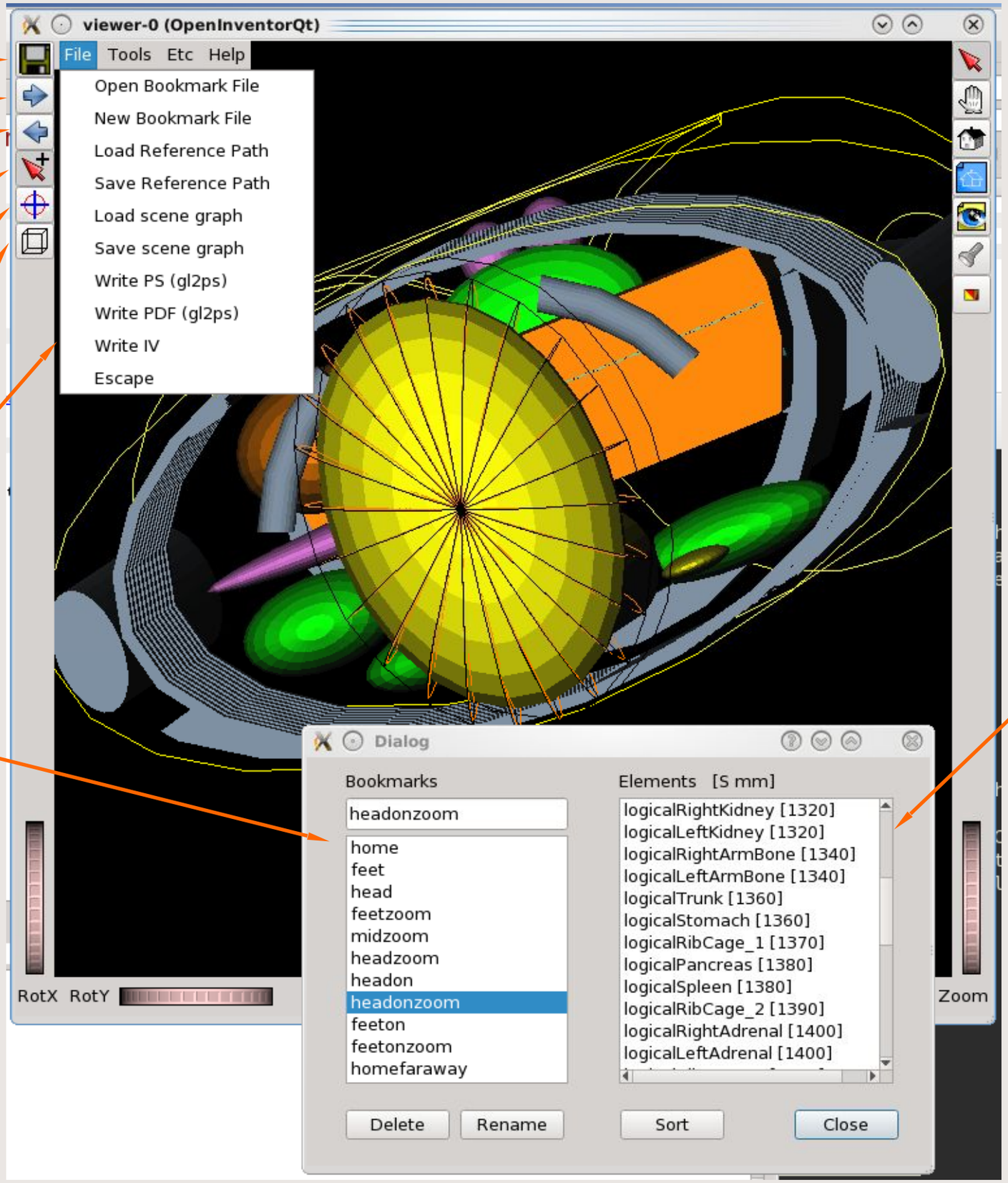
Next bookmark

Previous bookmark

Extended picking & mouse-over display

Pick reference path (trajectory)

Toggle wireframe



Built-in viewer controls

Input/Output menu

Bookmark List
Click for random access or use viewer buttons for sequential access

Element List
Ordered by distance along reference path. Click to navigate to element or use arrow keys to travel between elements and rotate around them.

Summary: Progress on OIQt Development

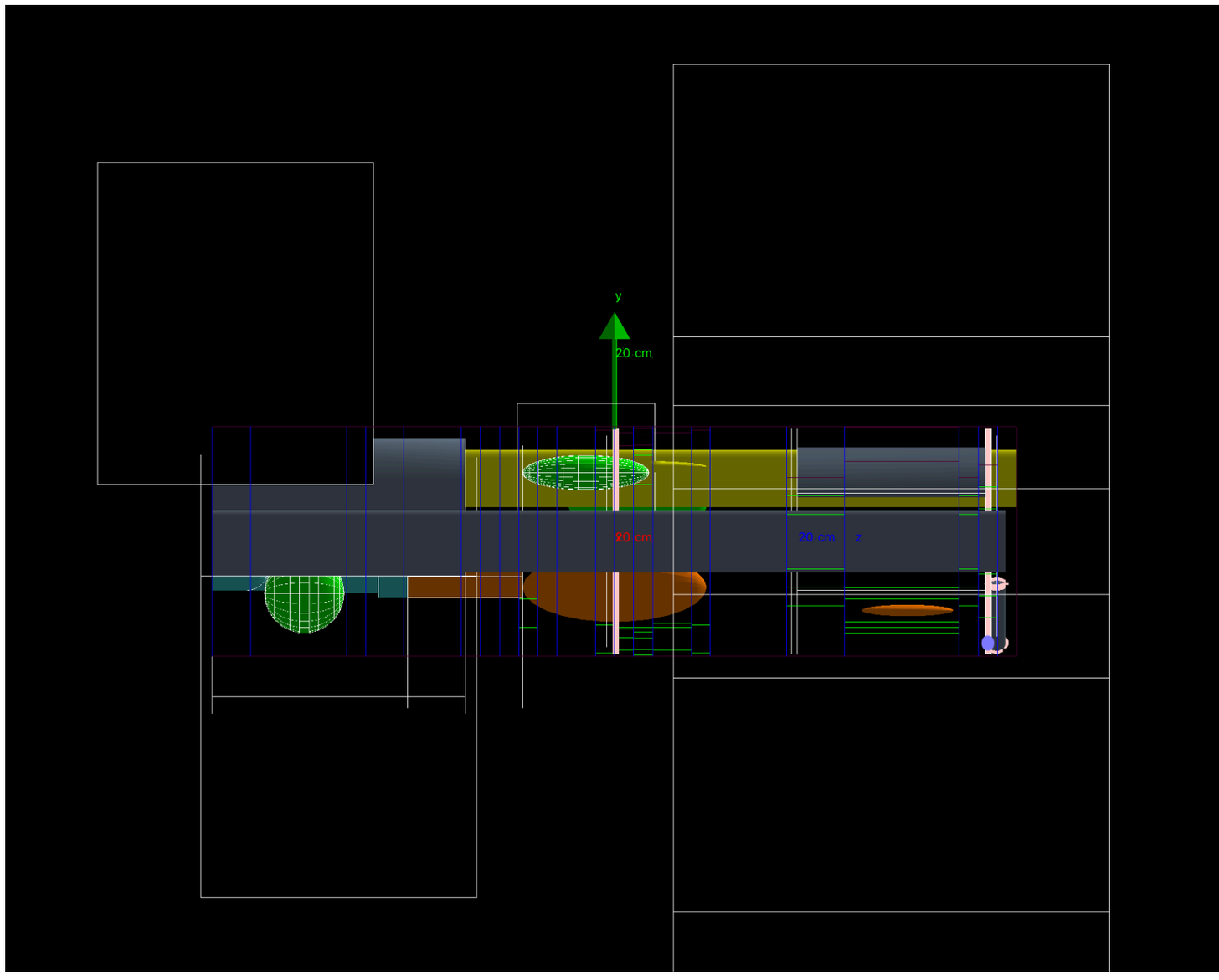
- Phases completed:
  - Corrections to CMake configuration and addition of GEANT4_USE_INVENTOR_QT option.
  - A prototype OIQt viewer (fully functional but no add-ons) was released in Geant4 10.7 Beta.
  - All functions migrated to Qt widgets.
  - All Bookmark functions implemented.
  - Element navigation, rotation, and geometry fly-through.
  - Superimposed scene, extended picking, and mouse-over readouts for volumes and trajectories.
- Phases remaining:
  - Add some viewer I/O functions such as PS and PDF output.
  - Improve layout of the Lists Dialog (Qt Designer).
  - General shake-down to remove glitches and resolve some inter-operability issues (e.g. embedding in the Qt UI).

Special thanks to John Allison for his interest in this project and for essential feedback and support.

Debugging the geometry of the human phantom

/vis/drawLogicalVolume logicalTrunk

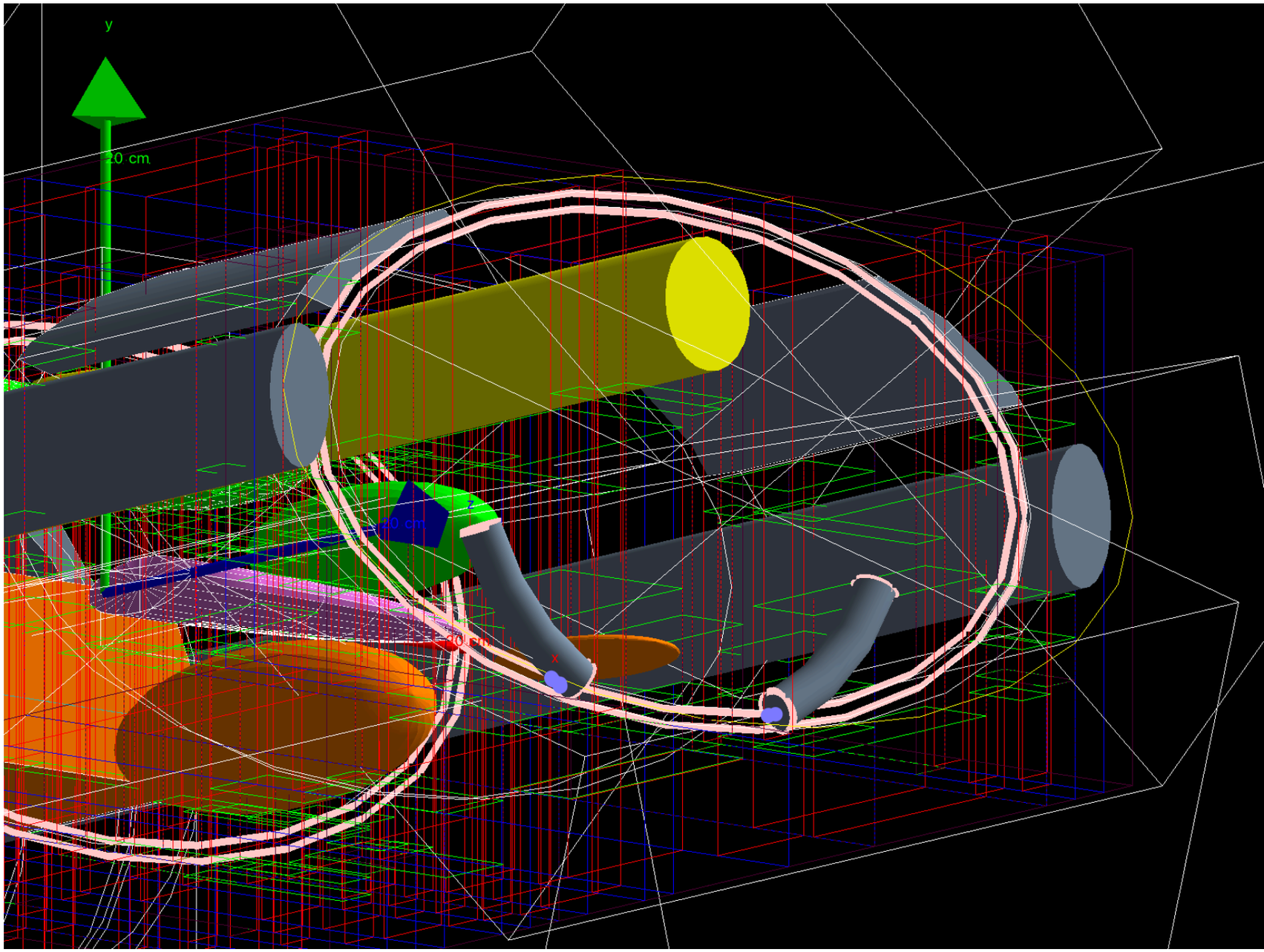
- You have to know the name of the relevant logical volume
  - Use /vis/drawTree
- This shows all daughters
  - Overlapping volumes in pink
  - Overlapping points in pale blue



Clavicles

Shows overlapping clavicles

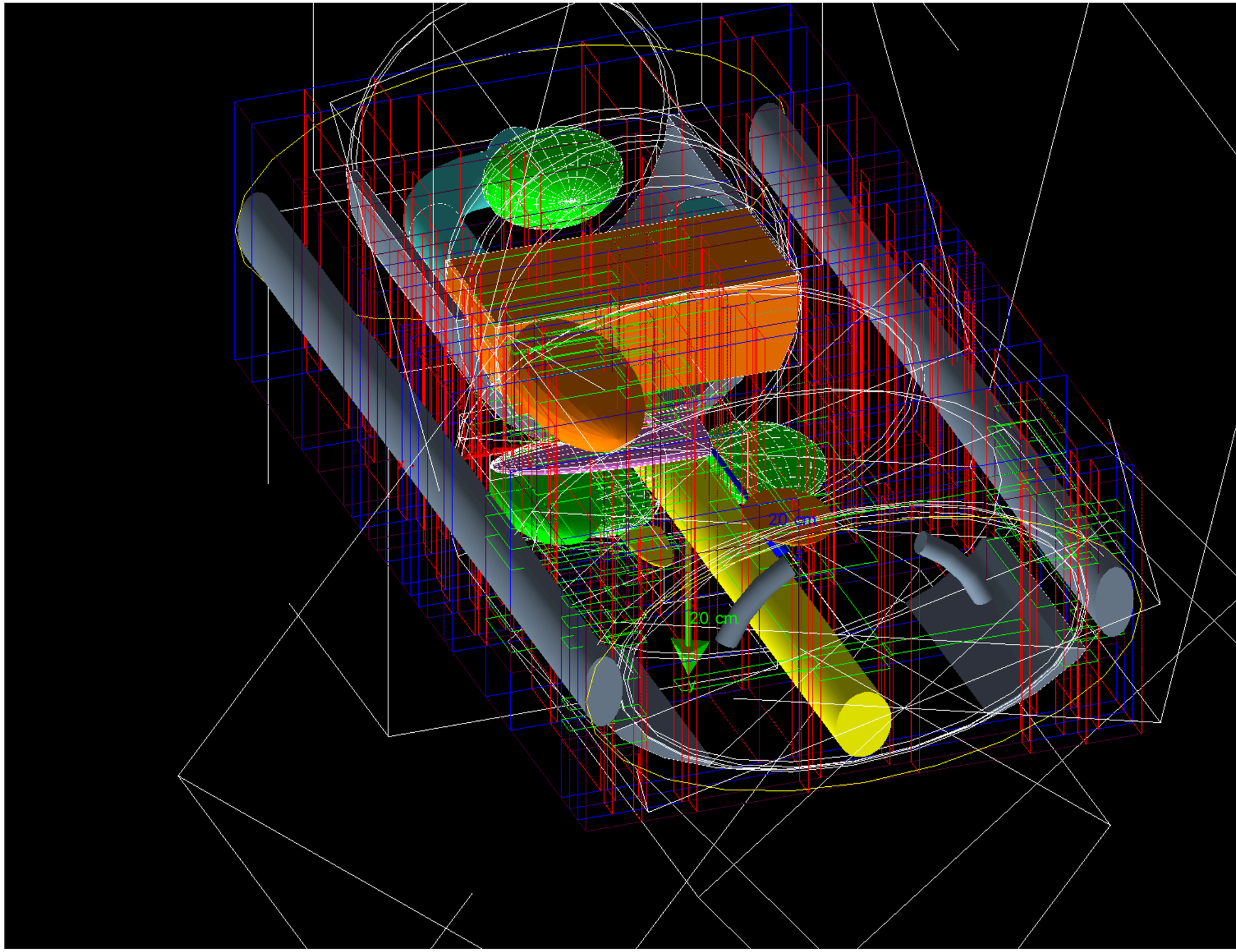
- Easily fixed
  - Move 2 cm +y



All OK

Shows
geometry
voxels

and Boolean
components



Questions ?