# Software Philosophy and a Proposed Reality for Water Cherenkov Simulation and Reconstruction

Brett Viren

Physics Department

BROOKHAVEN NATIONAL LABORATORY

DUSEL Collaboration Meeting @ BNL, October 2008

# Contents

# Assumptions About the Software

- We have limited available effort for software, what we have is geographically spread out and comes with assorted expertise.
- Lots of large and small ideas and designs details need vetting through simulation.
- Might use different design for each detector module.
- Lots of physics, lots of different analysis requirements.
- Experiment will run for a long time, we hope!
- Need quick, short-term results.

# Philosophical Choices

- Do we organize now for the long term or just hack away and hope for the best in the future?
  - I'm interested in working under the assumption of success and working towards a lasting software organization.
- If/when we organize, what is our software model?
  - Toolkit?
  - Framework? ($\leftarrow$ hint: this is the right answer)
  - Organic? (`*shudder*`)
- Do we write or do we steal?
  - Whoever said "It never pays to steal" doesn't do software.

# Step 1: Learn from the Daya Bay Experience

**Steal from LHC for the win!**

- CMT for the build system and environment setup
  - Widely used in our community, flexible and supported.
  - Allows cohesive collection of disparate software packages.
- Gaudi for the framework
  - Hides the tedious stuff, provides hooks for the important stuff.
  - Encourages modularity, collaborative development.
- DetDesc for detector geometry and material property description
  - Smart, XML based description language allows algorithmic or declarative descriptions.
  - Description independent from code - easy to test out new designs.
  - Unified geometry, used in both simulation and reconstruction contexts.
- GiGa to interface with Geant4
  - Lightly wraps G4 with a Gaudi skin.
  - Allows modular, collaborative simulation code development.

# Aside: Focus on Geometry

Initially, we need to simulate and explore grossly different detector ideas and understand optical properties:

- Specific design choices need vetting:
  - Veto region or no veto region?
  - How does photocathode coverage and granularity affect reconstruction of different event types?
- Greater path lengths, greater the effects of water transparency and light dispersion. Do they matter?
- What can improvements in photodetector resolutions do for reconstruction efficacy.

DetDesc XML geometry descriptions allows the flexibility to easily explore these questions $\longrightarrow$

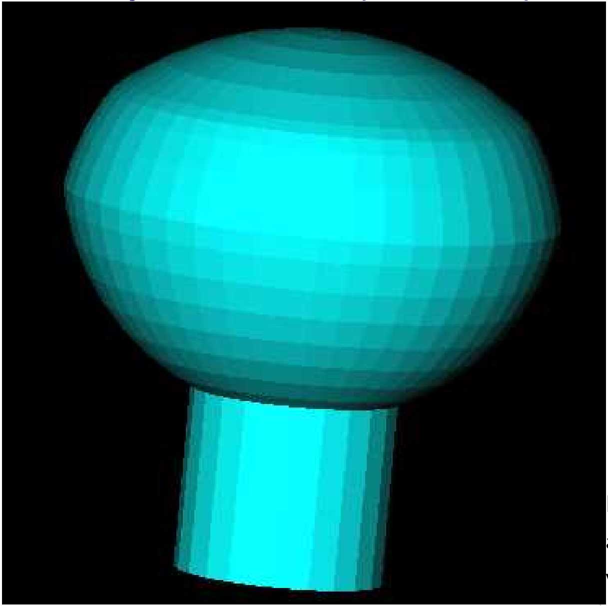
# Geometry XML - Example 1: Daya Bay AD

Daya Bay Antineutrino Detector = an oil cylinder logical volume with two daughters: an outer acrylic cylinder and an array of PMTs.

```
<logvol name="lvOIL" material="MineralOil">
  <!-- the shape: -->
  <tubs name="oil" sizeZ="ADoilHeight"
                outerRadius="ADoilRadius" />

  <!-- any physical daughter volumes: -->
  <physvol name="pvOAV"
           logvol="/dd/Geometry/AD/lvOAV"/>
  <physvol name="pvAdPmtArray"
           logvol="/dd/Geometry/AdPmts/lvAdPmtArray"/>
</logvol>
```

Parameters make any particular description more flexible.

# Geometry XML - Example 2: Simple PMT

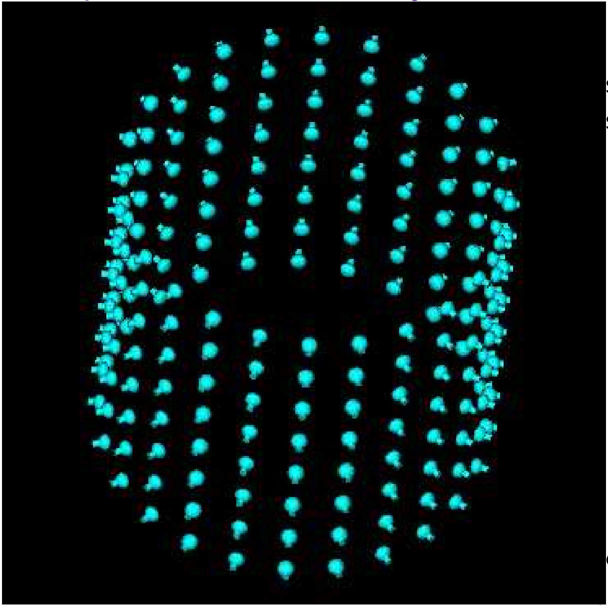


- Simple intersection of three hemispheres
- Glass, photocathode and vacuum volumes
- Photocathode / sensitive detector connection given in XML
- Validated against detailed GLG4Sim Hamamatsu model.

Note: polygons are just an artifact of this particular visualization program.

```
<logvol name="lvPmtHemi" material="Pyrex">
  <union name="pmt-hemi">
    <intersection name="pmt-hemi-glass-bulb">
      <sphere name="pmt-hemi-face-glass"
              outerRadius="PmtHemiFaceROC"/>
      <sphere name="pmt-hemi-top-glass"
              outerRadius="PmtHemiBellyROC"/>
      <posXYZ z="PmtHemiFaceOff-PmtHemiBellyOff"/>
     <sphere name="pmt-hemi-bot-glass"
              outerRadius="PmtHemiBellyROC"/>
      <posXYZ z="PmtHemiFaceOff+PmtHemiBellyOff"/>
    </intersection>
    <tubs name="pmt-hemi-base"
          sizeZ="PmtHemiGlassBaseLength"
          outerRadius="PmtHemiGlassBaseRadius"/>
    <posXYZ z="-0.5*PmtHemiGlassBaseLength"/>
  </union>
  <physvol name="pvPmtHemiCathode"
           logvol="/dd/Geometry/PMT/lvPmtHemiCathode"/>
  <physvol name="pvPmtHemiVacuum"
           logvol="/dd/Geometry/PMT/lvPmtHemiVacuum"/>
</logvol>
```

# Example 3: AD PMT Array



Powerfully, expressive descriptions possible. Only 4 steps needed:

1. Position one PMT at bottom of tank
2. Copy to make ring of 24 PMTs
3. Copy ring 8 times
4. Rotate everything 1/2 angular period for proper absolute alignment.

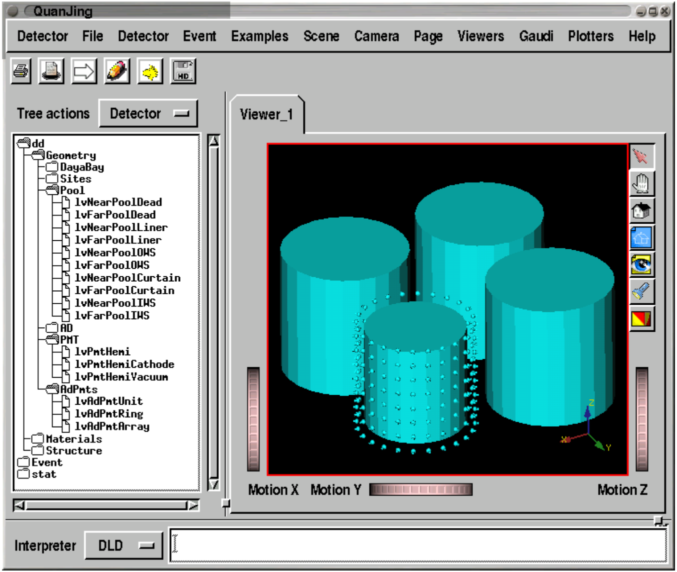
Result can be placed as one volume.

# Building the AD PMT array with parameterized placement

```
<logvol name="lvAdPmtUnit">                <!-- step 1 -->
  <physvol name="pvAdPmtUnit" logvol="/dd/Geometry/PMT/lvPmtHemi">
    <posXYZ x="AdPmtRadialPos" z="-0.5*(AdPmtNrings-1)*AdPmtZsep"/>
    <rotXYZ rotY="-90*degree" />
  </physvol>
</logvol>

<logvol name="lvAdPmtRing">                <!-- step 2 -->
  <paramphysvol number="AdPmtNperRing">
    <physvol name="pvAdPmtInRing:1" logvol="/dd/Geometry/AdPmts/lvAdPmtUnit" />
    <posXYZ/>
    <rotXYZ rotZ="AdPmtAngularSep" />
  </paramphysvol>
</logvol>

<logvol name="lvAdPmtArrayZero">           <!-- step 3 -->
  <paramphysvol number="AdPmtNrings">
    <physvol name="pvAdPmtRingInCyl:1" logvol="/dd/Geometry/AdPmts/lvAdPmtRing"/>
    <posXYZ z="AdPmtZsep"/>
  </paramphysvol>
</logvol>

<logvol name="lvAdPmtArray">               <!-- step 4 -->
  <physvol name="pvAdPmtArray" logvol="/dd/Geometry/AdPmts/lvAdPmtArrayZero">
    <posXYZ/>
    <rotXYZ rotZ="0.5*AdPmtAngularSep"/>
  </physvol>
</logvol>
```

Fully parameter driven, no hard coded values. Allows easy numerology games.

# Direct visualization: quanjing.py (nee' Panoramix)




- Interactively inspect Gaudi data stores
- Rotate, pan, zoom
- Interrogate volumes
- Drill down through mother/daughters
- Transparency & color effects
- GUI built from simple XML
- Easy to plug in user code
- Can call C++ or Python from interpreter
- Display event data

## Step 2: Steal from the Daya Bay Experience

- The joy of stealing is tempered with the work of integrating.
  - Daya Bay has a mostly agnostic install script that takes you from nothing to a fully working system with one command (and about 3 hours depending on your network, CPU and disk speeds).
- Kinematics generator subsystem
  - Suite of pluggable modules to build up initial kinematics.
  - Direct C++ classes or external HepEVT generators
  - Could easily integrate NUANCE or GENIE
- ROOT based I/O mechanism using standard Gaudi converter idiom
  - Can support time window based analysis (eg. finding $\mu$-decay)
- Electronics & trigger packages may be available (I didn't write them).

For my part, an intentional effort was made to make Daya Bay software generic, **expressly for the purpose of using it on DUSEL WC**.

# Effort needed

Assumes individual has strong competence in required tools and at least a high level understanding of the Gaudi framework (read manual).

- Make the install script truly Daya Bay agnostic, build a full set of non Daya Bay packages.
  - Effort: 1-4 months, post-doc or above recommended
  - Tools: shell scripting, CMT, unix build tools, building Free Software from source
- Develop a first, trivial model of a current strawman WC detector in DetDesc XML.
  - Effort: 1 week to 1 month, student or above
  - Tools: XML, DTD, scripting, 3D transformations, previous simulation work would help
- Port the few Daya Bay packages (generators, Geant4 support, I/O)
  - Effort: 1-4 months (depending on scope), post-doc or above
  - Tools: C++, Python, ROOT, Geant4 and DetDesc.

I plan on doing work after ∼6 months time. In the mean time I'll help anyone to get started.

# Getting organized

Regardless of the above, the software effort needs to get organized. The first two apply to all DUSEL groups.

- Need official mailing list(s)
  - Leverage "free", expert support using **lists.bnl.gov**
  - Aside: also hosts `gaudi-talk@lists.bnl.gov` which has brought together the many Gaudi-using experiments into a helpful group.
- Public web presence and private, internal organization pages:
  - Leverage "free", expert support using **wiki.bnl.gov**
- Agree on software repository mechanism, centralized vs. decentralized, but definitely need to be inter-operating.
  - Suggest embracing **GIT**. SVN is a distant second.
- Bug/issue tracking
  - **Trac** very helpful for Daya Bay, becoming a FOSS standard.
  - Can integrate with unit and validation testing

# Summary

- Need to determine DUSEL's near and long term software strategy and development model.
- Proposal to steal from Daya Bay and the LHC experiments' efforts (thanks!)
  - Some "agnosticizing" needed.
  - Mostly a "turn-key" solution.
  - A rough effort estimation given.
- Independent of this, some organizational infrastructure is needed and suggestions given.