LUXSim: A Component-Centric Approach to Low Backgroung Sims

Kareem Kazkaz, LLNL
on behalf of
LUX and the LUXSim Developers

AARM Workshop, Feb 2011, Minneapolis

LUXSim Developers

Kareem Kazkaz (LLNL, kareem@llnl.gov)
- Architecture, Management and Physics lead

David Malling (Brown)
- Materials and Matlab interface lead

Melinda Sweany (UCD)
- Geometry lead

Nick Walsh (UCD)
- Generator lead

Mike Woods (UCD)
- Detector response lead

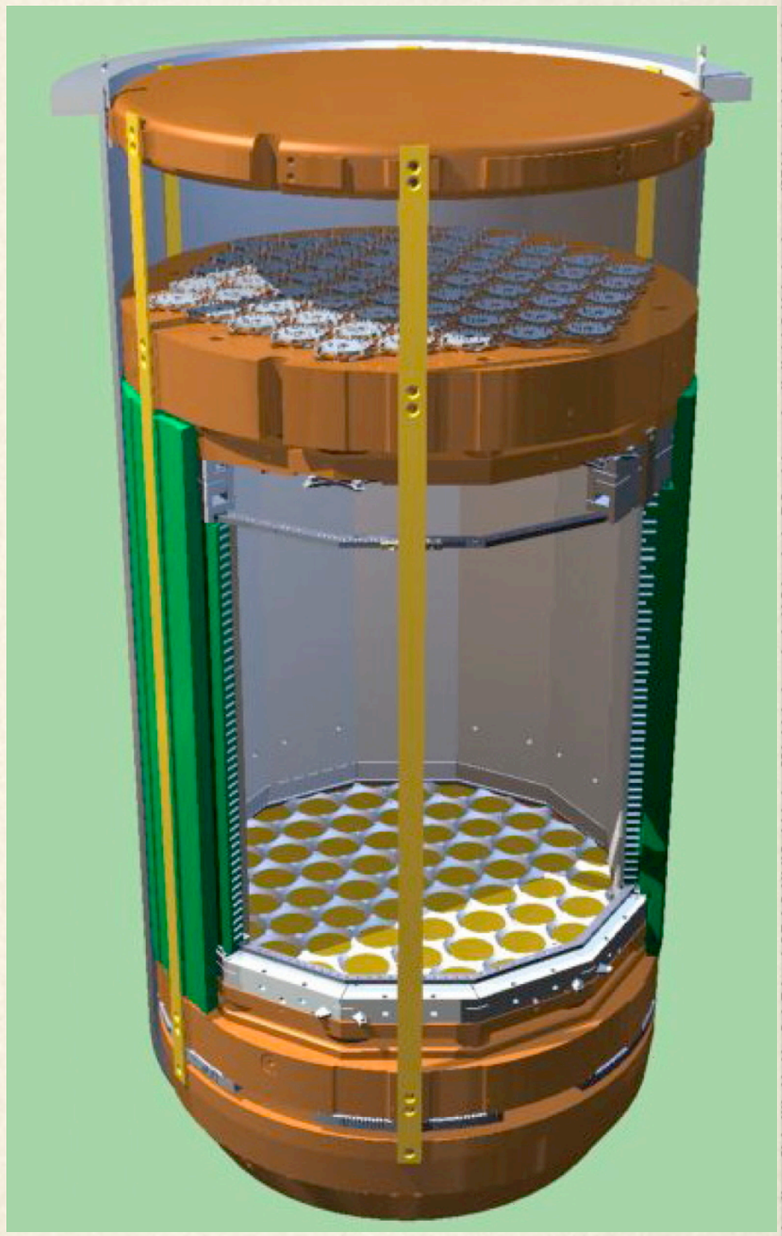
Chao Zhang (USD)
- I/O and ROOT interface lead

LUXSim Report

- The LUX One-Sheet
- Making GEANT4 better for Low-BG, Nuclear-scale Sims
- The Wishlist for LUXSim
- LUXSim Subsystems
- Comments and suggestions

The LUX Experiment

- Direct search for Dark Matter
- A dozen institutions, ~70 participants
- Liquid xenon TPC (300 kg total, 100 kg fiducial) at the Sanford Lab
- WIMP/nucleon cross section sensitivity of 7×10^{-46} cm^2 after one year of running
- 2 background events in the 5-50 keV region of interest (both e^- and nuclear recoils), ~7 signal counts if $\sigma = 10^{-45}$ cm^2
- See http://lux.brown.edu



LOW BACKGROUND EXPERIMENT

Traditional GEANT4

- Used for HEP experiments
- Shoot a beam of particles at a target
- In many cases, stop with statistical effects

Expanded functionality for nuclear / LE physics

- EM interactions down to 250 eV
- Neutron interactions to thermal energies
- Radioactive decays
- Event generation from a volume rather than a point
- And more…

LUXSim Wishlist

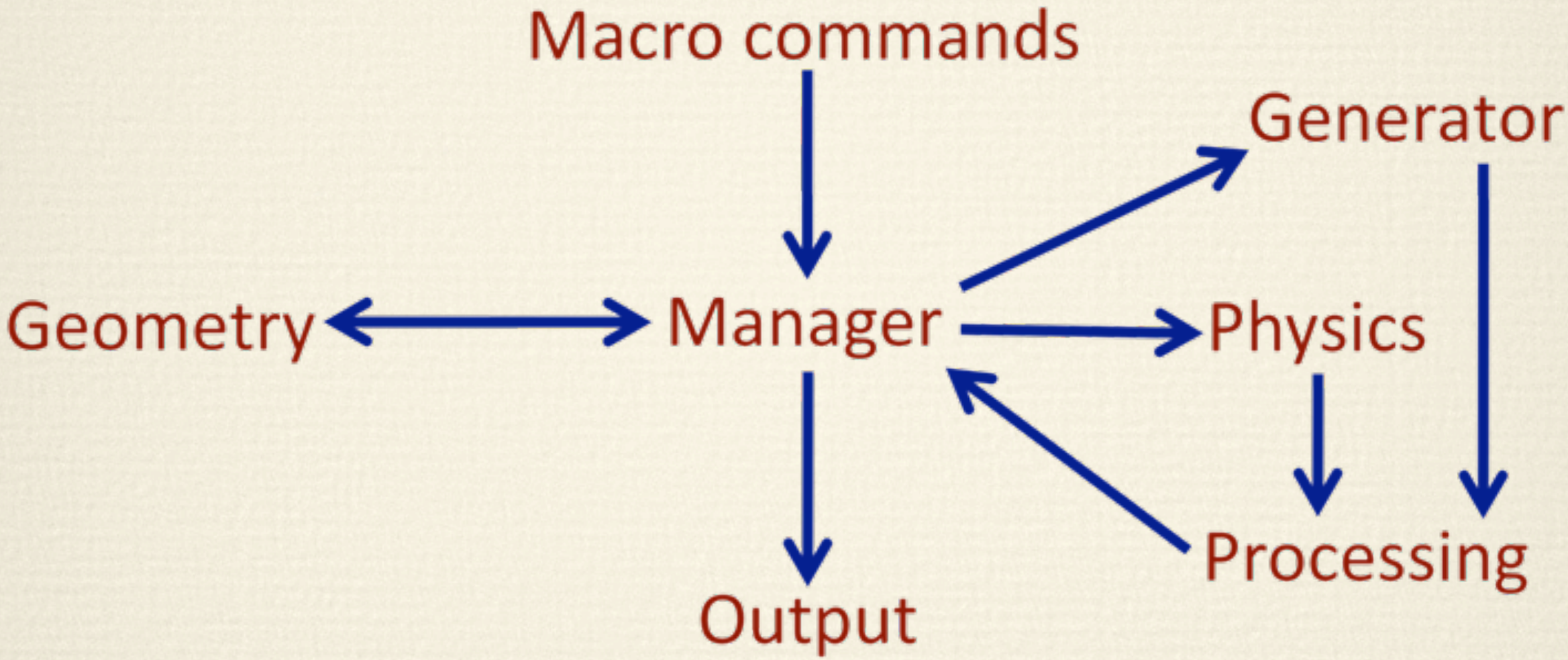
Expanding on basic GEANT4 functionality

- Multiple volume sources (same or distinct volumes)
  → Appropriate generators
- Framework automation
- No hard-coded "sensitive detectors"
- Multiple levels of event recording in the output files
- Set event recording and physics at run time

Also want...

- Reduce or eliminate user coding (reduces bugs)
- Reproducibility / Reliability (header information)
- Simplicity for the users (developers burn brain cells)
- Agnostic output format (Matlab and ROOT converters)

Simulation Structure



Manager Subsystem

Sits at the center of the web, contains pointers to all subsystem classes (and provides Get methods) to provide easy inter-class communications

Controls the running of the simulation (/LUXSim/beamOn)

- Registration methods for all other classes
- Generates and maintains header information
- Full record of all detector components
- Performs calculations for source activities and determines which component has the next decay

  PrimaryGeneratorAction::GeneratePrimaries calls the manager
- Holds physics switches to pass to the physics list

Geometry Subsystem

The usual standard, object-oriented, nested volumes...

BUT

We created a new class: LUXSimDetectorComponent, which inherits from G4PVPlacement and has

- Automatic registration methods with the manager class
- Support for multiple sources (stored in a vector)
  - Automatic activity ratio calculation
  - Provides primary vertex: class determines its own center, rotation, extent, all keyed on a pointer to the physical volume to avoid ambiguity
- Data record (energy deposition vector), including record level

So in any existing geometry, do a global search-and-replace of "G4PVPlacement" with "LUXSimDetectorComponent"

Generator Subsystem

Geared for low-background simulations:

- Cosmic muons and spallation neutrons (Mei & Hime)
- Single nuclides (^{40}K, ^{137}Cs, ^{xx}Co)
- Decay chains (^{238}U, ^{232}Th, still need ^{235}U, ^{228}Th, Ra)
- AmBe, (α,n), ^{252}Cf neutrons

All events generated chronologically in the simulation

- Avoid throwing out events before secular equilibrium
- Avoid having to time-order events after the fact

Can load any and all sources onto any and all detector components simultaneously, and specify activity for each

at run time, with no new coding

Output Subsystem

Set the energy record level of each individual component...

0 - No information (default)
1 - Total energy only
2 - Individual steps if deposition > 0
3 - All individual steps
4 - Just the first step, then kill the track

...and set the optical photon record level...

0 - No information (default)
1 - Just total number
2 - All information
3 - Just total, and kill the track
4 - All info, and kill the track

at run time, with no new coding

Output File

Want to be able to reproduce a data file, or at least know how it came into being. So we record a header in the data file:

- GEANT4 & LUXSim version
- SVN diffs
- Date/time stamp
- Operating system and computer name
- Randomization seed
- Macro commands

This allows for reconstruction of the data file, and protects us from making changes and forgetting what they were.

Questions?
Suggestions?