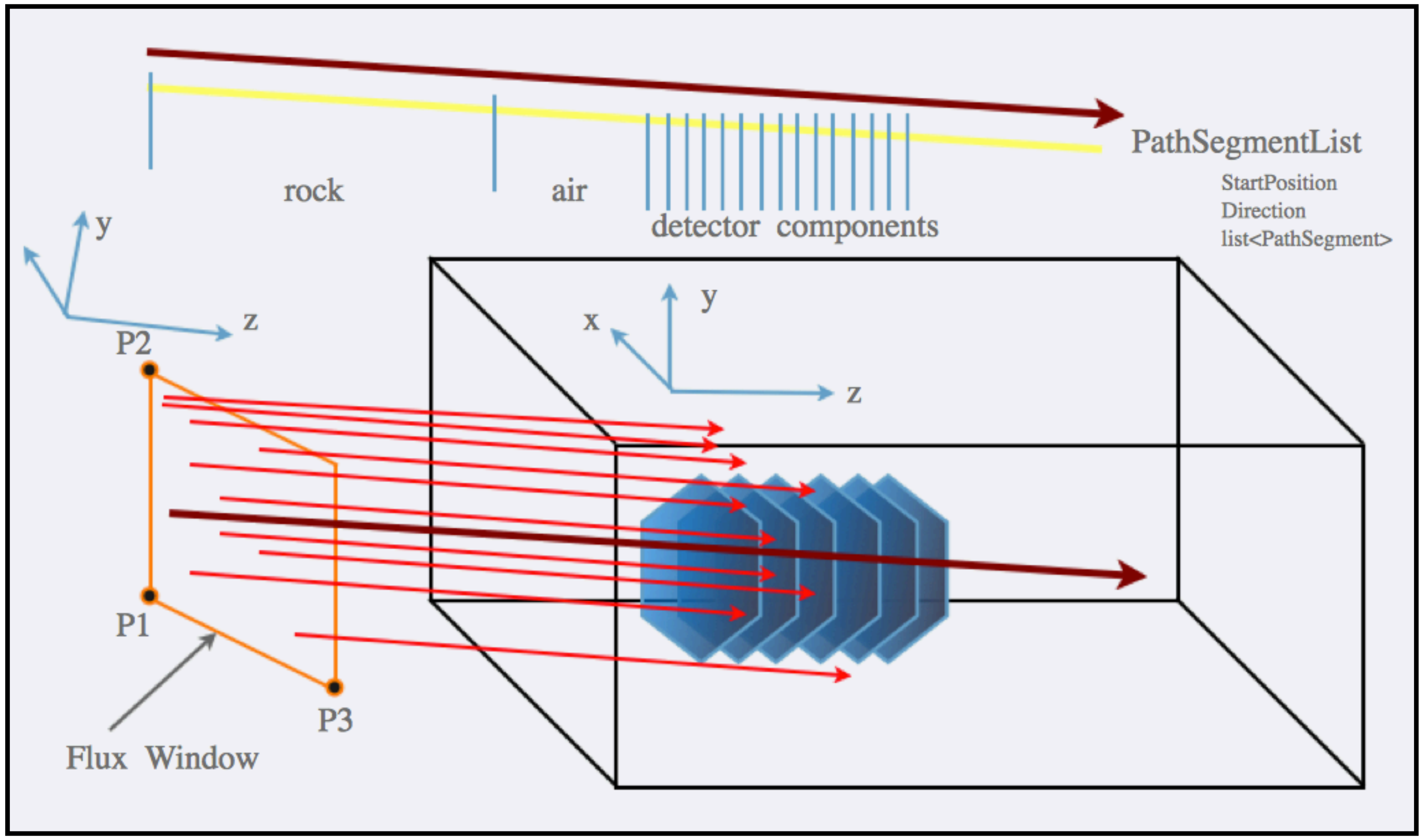
Using Fluxes and Geometries

Gabriel N. Perdue
Fermilab

Special thanks to Robert Hatcher for much of the material in this presentation.

Why does GENIE need Geometry?

- Real fluxes and geometries are never uniform.
  - Experiments need to generate interaction vertices in the correct locations.
  - Fluxes vary in intensity and energy profile across the detector.
  - Detector structures (and the surrounding area!) have specific structures and boundaries.



High-Level View

- `genie::GMCJDriver` controls event generation.
  - Initialized with a `genie::GeomAnalyzerI` instance.
    - Using `genie::ROOTGeomAnalyzer` specialized class.
    - `genie::PointGeomAnalyzer` also exists.
  - Also initialized with a `genie::GFluxI` object.
    - Basic interface must supply rays with [p4, x4, flavor, weight].
    - Get a new ray with every call to `GenerateNext()`

Fluxes

- Many choices (including making your own):
  - User-specified histograms (no spatial variation, only energy and flavor)
  - Encapsulations of common parameterizations (e.g., atmospheric)
  - Simple, generic ntuple format (`GSimpleNtpFlux`*)
  - Experiment (NuMI, T2K) or institution specific.
- Wrap any of the above in a "flavor blender" adapter (`GFlavorMixerI`) - this is how you handle far detectors in an oscillation experiment.
- Some drivers have exposure counters (e.g., time, protons on target).

*FNAL beamlines committed to migrating to this common ntuple format (dk2nu).

Job Driver

- With geometry and flux, the job driver then computes the maximum path lengths in materials.
  - This allows overall scaling for more efficient generation.
  - The sampling may depend on the flux source (e.g., atmospheric is different from beam) as well as the geometry.
- The driver also reads the cross section spline file.

Event Generation

- `genie::GMCJDriver` controls event generation.
  - Draw a neutrino ray from the flux driver.
  - Walk through the geometry along the ray's path.
    - Accumulate `PathSegments` as new volumes are traversed, taking note of material, length, and end-points.

Event Generation

- Use `PathSegments` + isotope cross-sections to determine overall interaction probability.
    - Compare to scale for maximum path-lengths.
- If an interaction occurs (use the rejection method), decide on material and vertex.
- Finally, generate event kinematics, hadronize, final state interactions, etc.

Geometry Basics

- GENIE uses a ROOT representation of the geometry for anything more complex than a point target.
  - Standard GENIE apps need a file containing either:
    - Saved ROOT geometry (mygeo.root)
    - GDML (XML DSL) description (mygeo.gdml)
  - User programs must construct a `genie::geometry::ROOTGeomAnalyzer`
    - Takes a ROOT `TGeoManager` as constructor argument.

ROOT & GDML Geometry

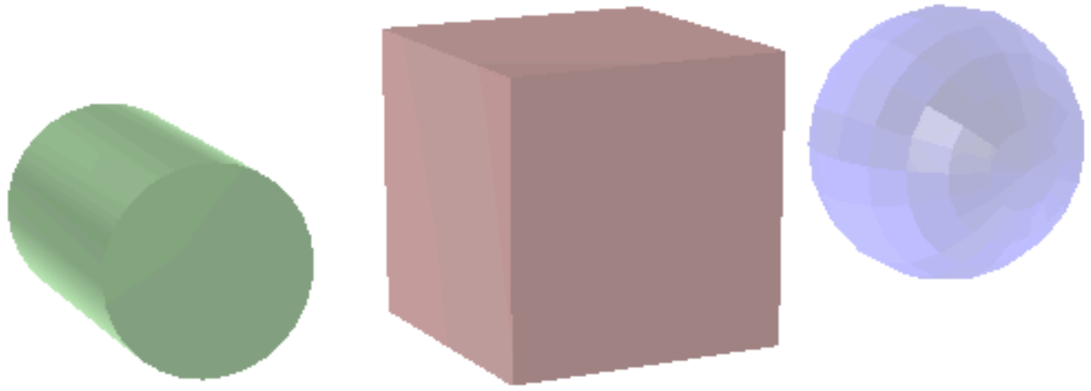
- `TGeoManager` class holds material, shape, volume, placement info.
  - General documentation:
    - ftp://root.cern.ch/root/doc/18Geometry.pdf
  - The default units are in cm when building it via calls to ROOT classes.
  - GENIE may be told what units the geometry is in.
- GDML == Geometry Description Markup Language
  - http://gdml.web.cern.ch/GDML/gdml.html
  - Both ROOT and Geant4 can read/write in this format.
    - **Reuse THE SAME geometry for event generation and detector simulation.**

Geometry Concepts

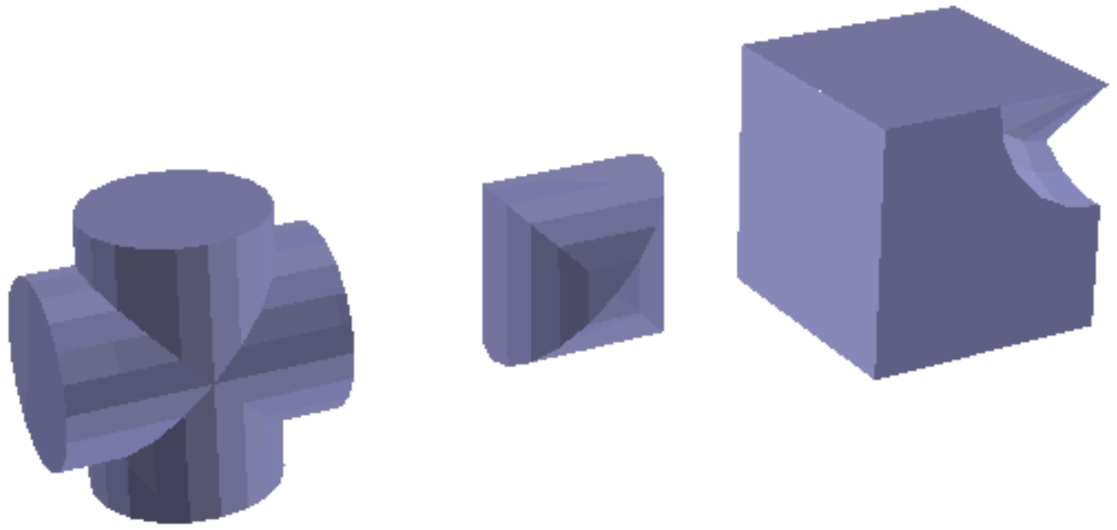
- Constructive Solid Geometry
  - Individual shapes (box, sphere, tube, extrusion, etc.)
  - Boolean combinations
    - Union == inside A or B
    - Intersection == inside A and B
    - Subtraction == inside A but not B

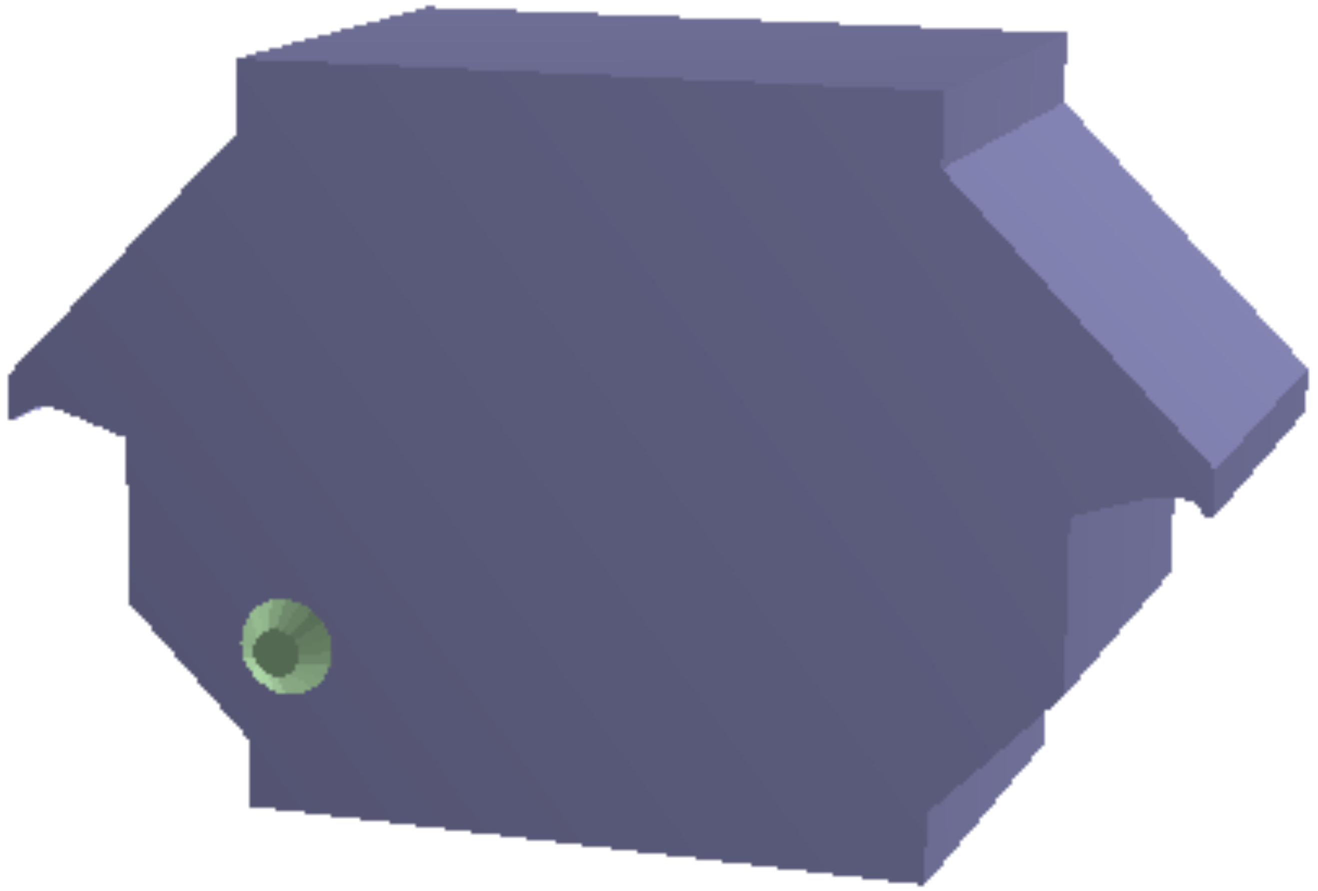
http://en.wikipedia.org/wiki/Constructive_solid_geometry

Geometry Concepts



Unions,
Intersections,
Subtractions





Extrusion

GDML Files

- Separate sections for:

  <define>: constants

  - Predefined values, rotations, positions.

  <materials>: what things are made of

  - Made of combination of elements + density.

  <solids>: basic shapes & boolean combinations

  <structure>: geometrical hierarchy

    <volume> == shape + material

    <physvol> == volume + placement/rotation

  <setup>: top-level volume, e.g. of the world

- Options for splitting complex geometries into sub-files.

ROOT & GDML Geometry

- Limitations in ROOT's GDML parser:
  - Not supported: replica volumes, loops, matrices
    - Perhaps in the future...
  - Failures to parse often provide very cryptic error messages.

ROOT & GDML Geometry

- git clone https://github.com/GENIEMC/GeometryAndFlux.git
  - Copy and paste (if you like) from the README there rather than from the talk pdf (quote characters may be mangled in the pdf).
- Reading and viewing a file in ROOT:

```
root [0] TGeoManager::Import("mysimplegeom.gdml");

root [1] gGeoManager->GetTopVolume()->Draw("ogl");
```

- "ogl" == Open GL

Flux

```
$ genie -b -q myfakefluxgen.C
```

- Or, `.C+` to compile, load, and run as a shared library.
- Generate a GSimpleNtpFlux file.

```
$ root myfakeflux.gsimple.root

root [0] flux->Draw("E");

root [1] flux->Draw("vtxy:vtxx","","COLZ")

root [2] TBrowser tb
```

PDG Info

```
$ root

root [0] TDatabasePDG* tpdg = TDatabasePDG::Instance();

root [1] TParticlePDG* apdg = tpdg->GetParticle("pi0");

root [2] int pdg_pi0 = apdg->PdgCode();

root [3] cout << "pi0 is " << pdg_pi0 << endl;

root [4] cout << "pi+ is " << tpdg->GetParticle("pi+")->PdgCode() <<
endl;
```

Use the Flux

```
$ cat flux.vec

$ export GENIEXSECFILE=$XSECSPLINEDIR/gxspl-vA-
v2.8.0.xml
```

- Or `gxspl-vA-v2.8.0.xml if it wasn't unzipped...`

```
$ gevgen -r 42 -n 100 -p 14 -e 0.5,6.0 -f
flux.vec -t 1000260560[0.82],1000080160[0.16],
1000010010[0.02] --cross-sections $GENIEXSECFILE
--event-record-print-level 0 --message-thresholds
Messenger_laconic.xml

$ genie gntp.42.ghep.root

genie [2] TBrowser tb;
```

Interactive GENIE

```
genie[3] TTree *mytree = 0;

genie[4] _file0->GetObject("gtree", mytree);

genie[5] Long64_t nEntries = mytree->GetEntries();

genie[6] cout << nEntries << endl;

genie[7] genie::NtpMCEventRecord* myentry = new genie::NtpMCEventRecord();

genie[8] mytree->SetBranchAddress("gmcrec", &myentry);

genie[9] mytree->GetEntry(0);

genie[10] myentry->PrintToStream(cout);

genie[11] myentry->event-Particle(0)->Print(); cout << endl;

genie[12] myentry->event->XSec()

genie[13] myentry->event->Vertex().Print()
```

- Also:

```
$ gntpc -i ./gntp.42.ghep.root -f gst
```

gevgen_simple

- ROOT script to run the event generation with a custom geometry and flux.

```
$ ./build_gevgen_simple.sh   # <<-- probably need to change for OSX
```

- Instructions also present in the 'README.txt'
- The script assumes an environment variable has been set for the cross section file.

```
$ export GENIEXSECFILE=$XSECSPLINEDIR/cross_sec.xml
```

- The geometry file provided uses nitrogen, oxygen, aluminum, argon, and copper (not all present in the default spline - nitrogen, aluminum, argon40 (argon38 is there?), and copper are missing).
  - Exercise: either generate the spline for those materials or edit the file to use iron in place of aluminum and copper, and oxygen in place of nitrogen and argon. If you have argon splines, you could just argon and vacuum.

gevgen_simple

```
$ ./gevgen_simple -h

$ ./gevgen_simple -n 10 -g mylardetector_simple.gdml
```

- 'mylardetector_simple.gdml' is actually a liquid oxygen detector (because oxygen is definitely in the spline file that comes with GENIE 2.8).
- 'gevgen_simple.C' is a simple GENIE app. It basically shows how to specify a flux and a geometry. By default is uses 'myfakeflux.gsimple.root' for the flux. The app currently only supports the "simple" flux format being adopted at NuMI, but you can see how to adapt it, etc.

Thanks!

Backup

Flux Concepts

- Flux driver supplies neutrino rays.
  - Each ray has:
    - Starting point (careful, SI units == meters)
    - 4-momentum (GeV)
    - Flavor (PDG code)
    - Weight