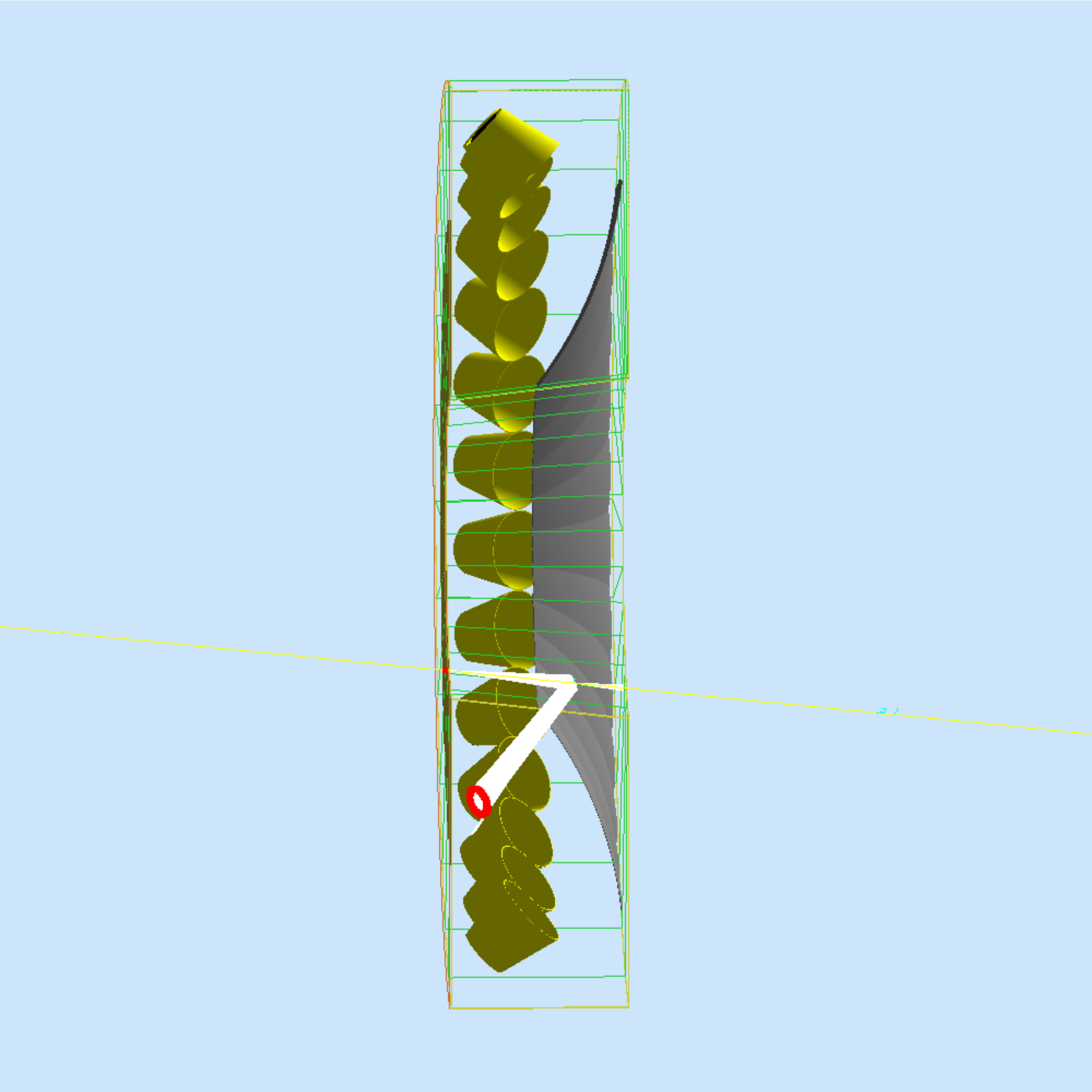
SoLID simulation with GEMC

Zhiwen Zhao

2015/09/11

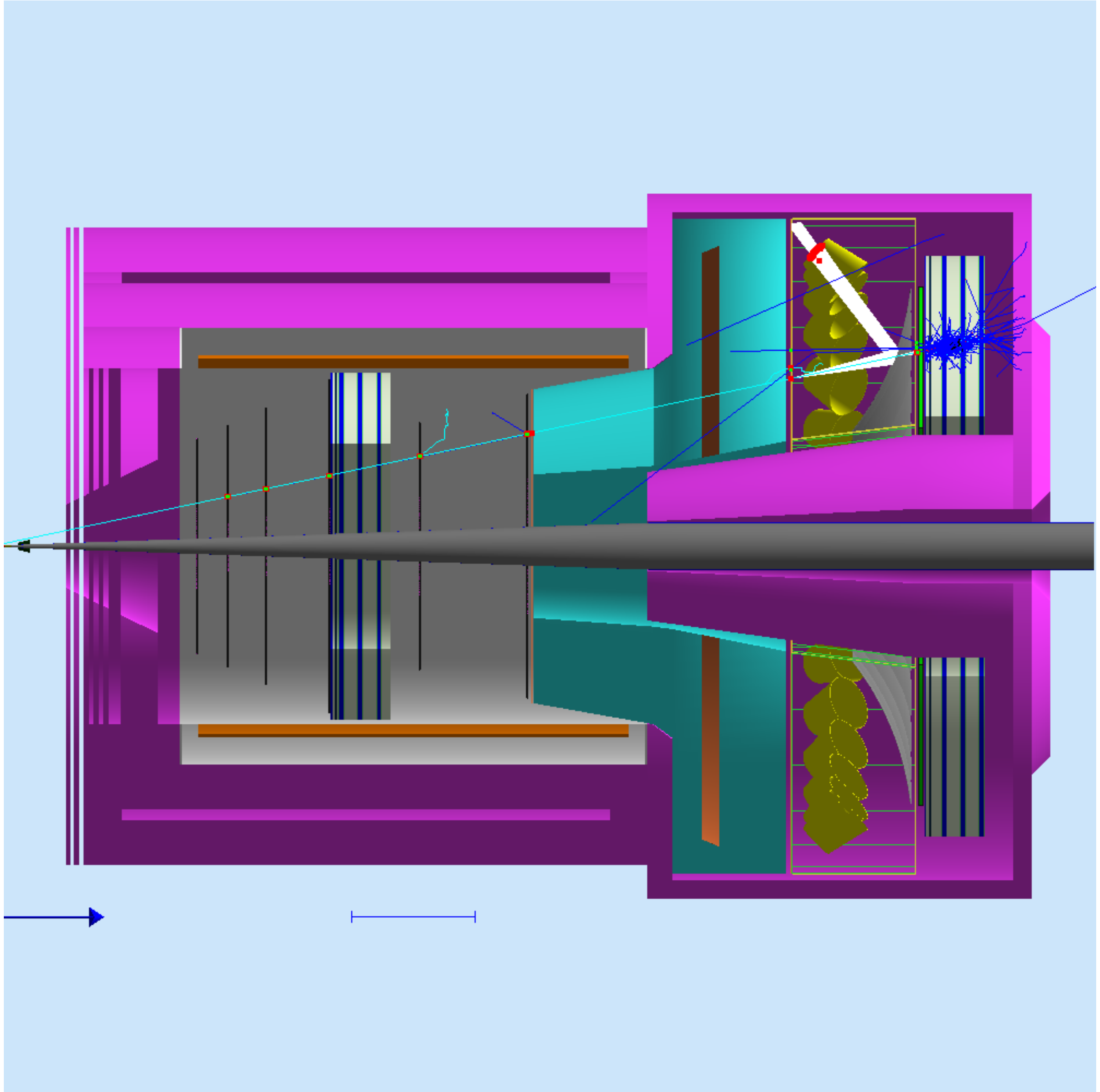
Standalone simulation in GEMC

- HGC



Full SoLID simulation in GEMC

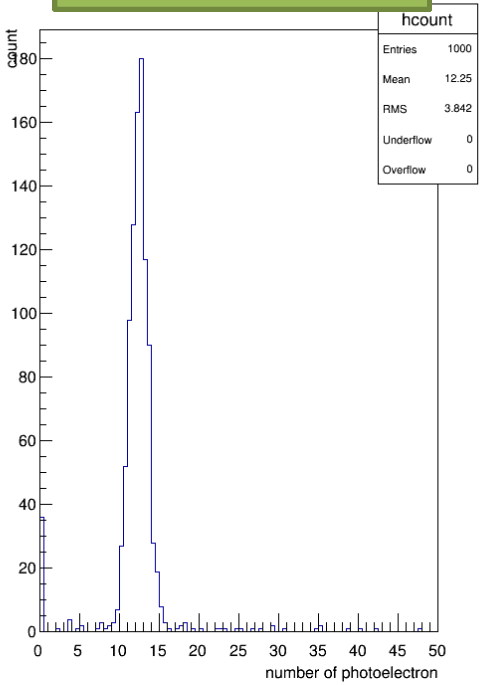
- Just drop-in with others
- Run with same field file, same condition, same output
- Nothing needs change



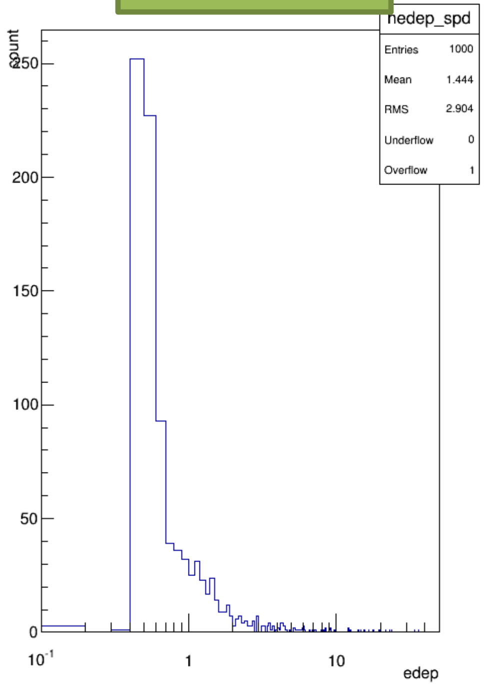
An example

- Shoot 4GeV pi- from target into forward detector

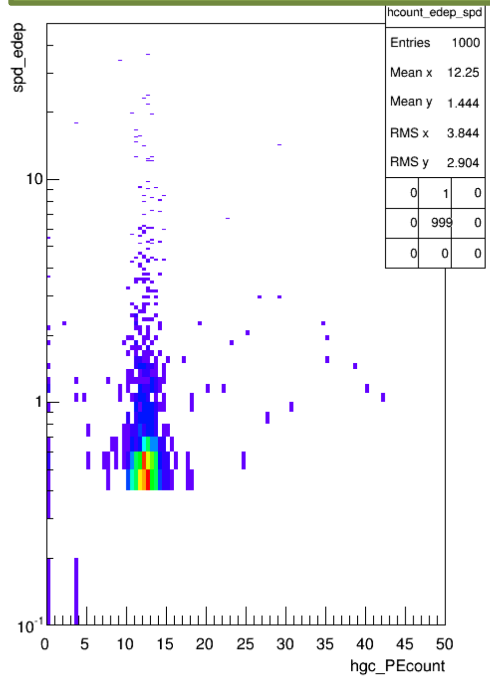
P.E. count in HGC



Edep in SPD



Relation between the two

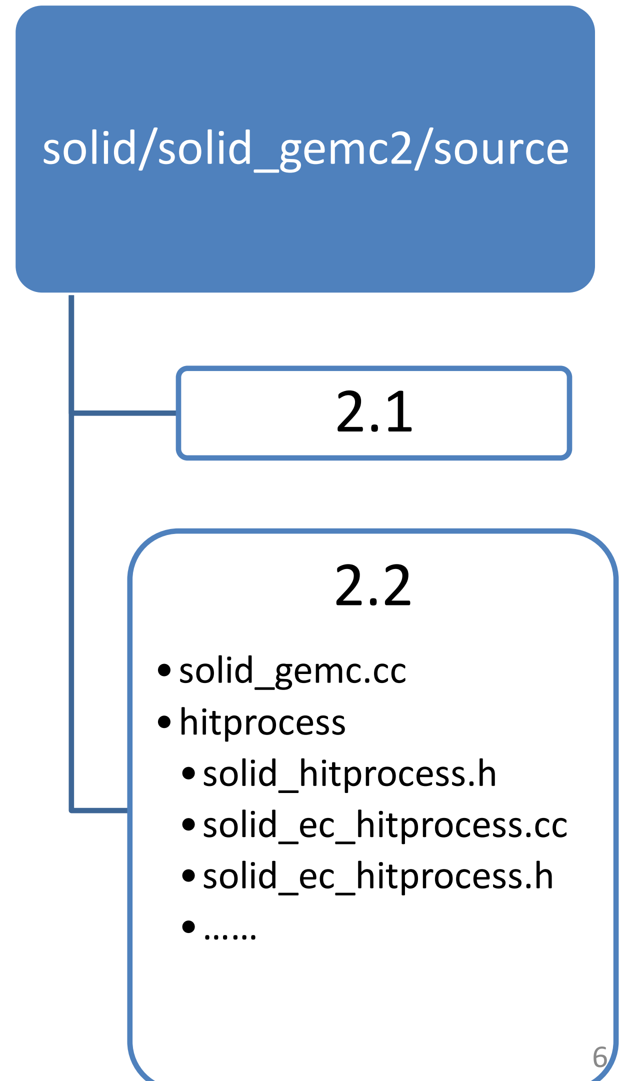


Everything external

- “Geometry, parameter, material, optical properties, output content” are all external
- They can be stored in file (regular version control) or database (CCDB)
- “parameter” can be anything, particularly useful to reconstruction/calibration/analysis
- “output content” defines data flow
- Both “Parameter” and “output content” can evolve

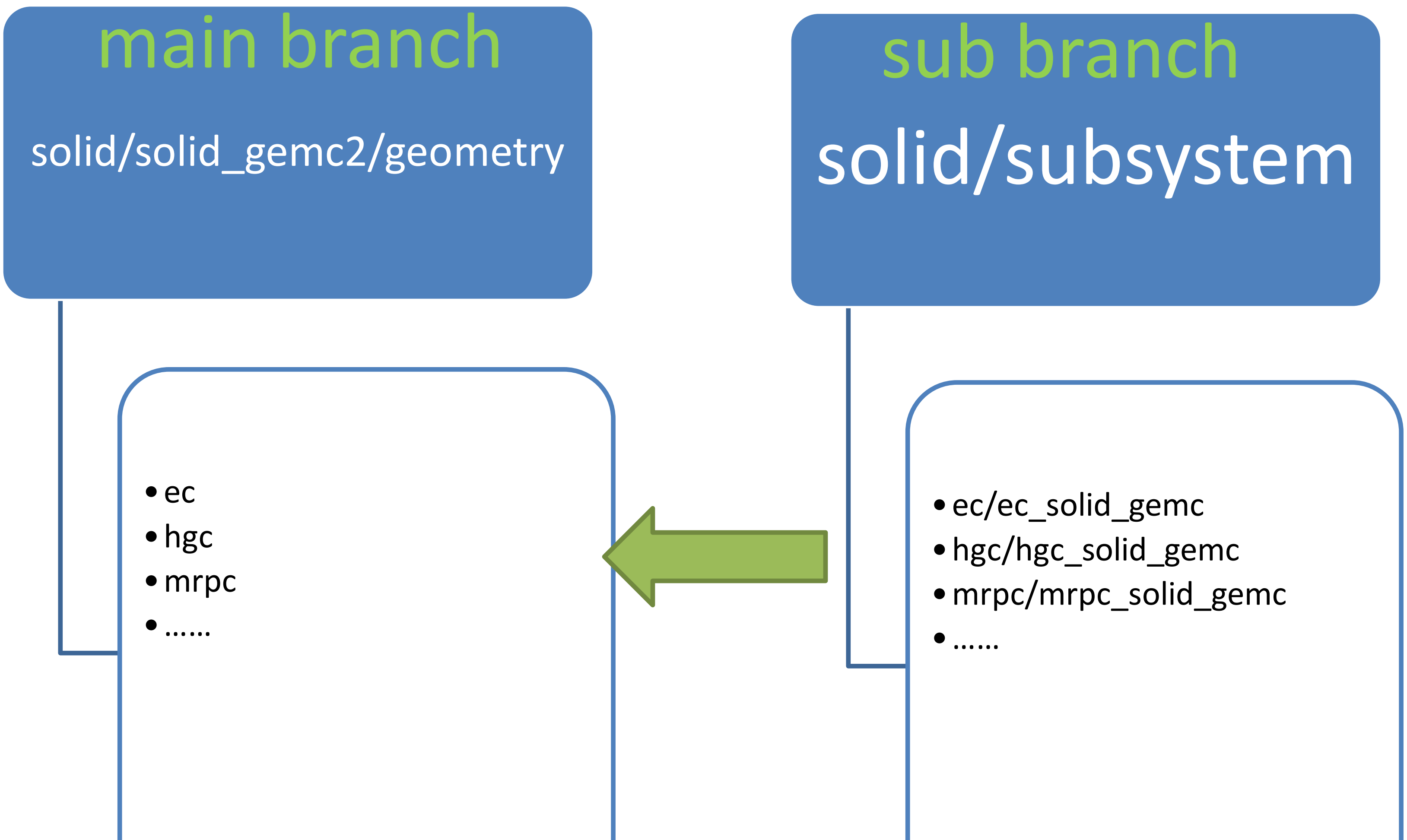
solid_gemc

- GEMC compiled as a lib during installation by default
- Then add customized hit process routing and other things and link to libgemc.so to keep all GEMC features
- This becomes "solid_gemc" with matching version of GEMC



Development model

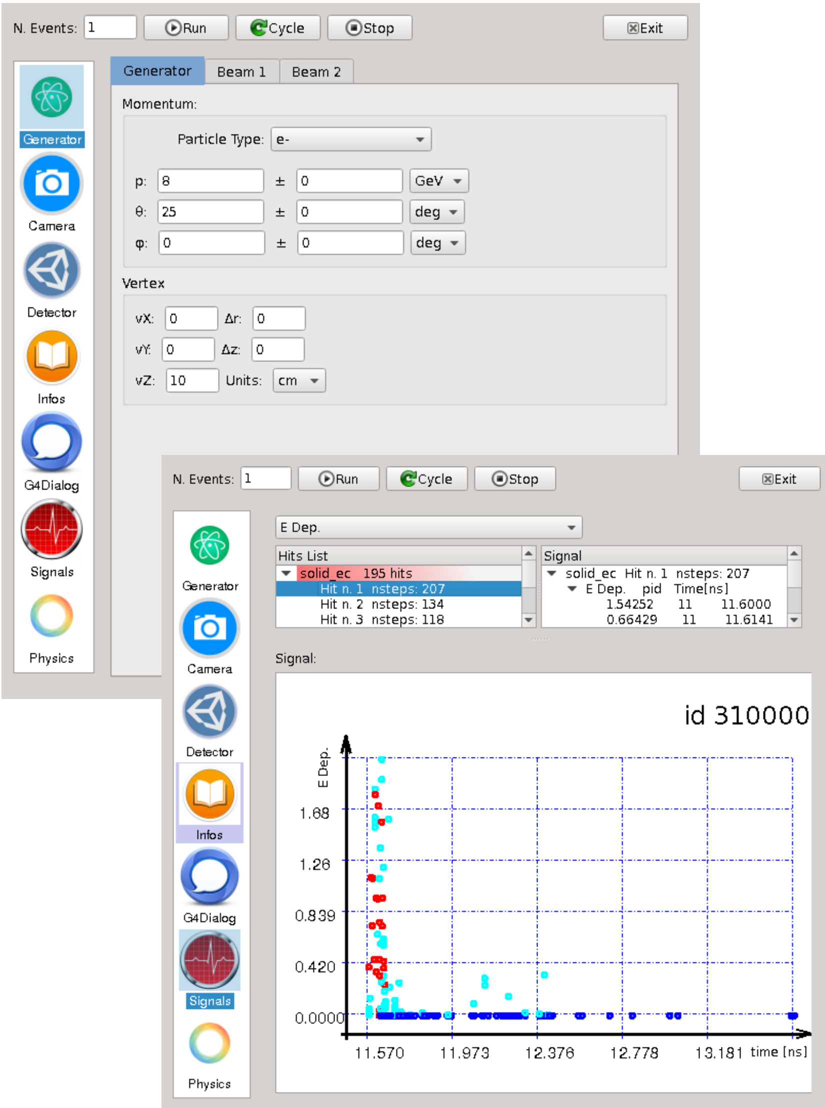
- test standalone in sub branch -> test with others from main branch in sub branch -> push from sub branch into main branch

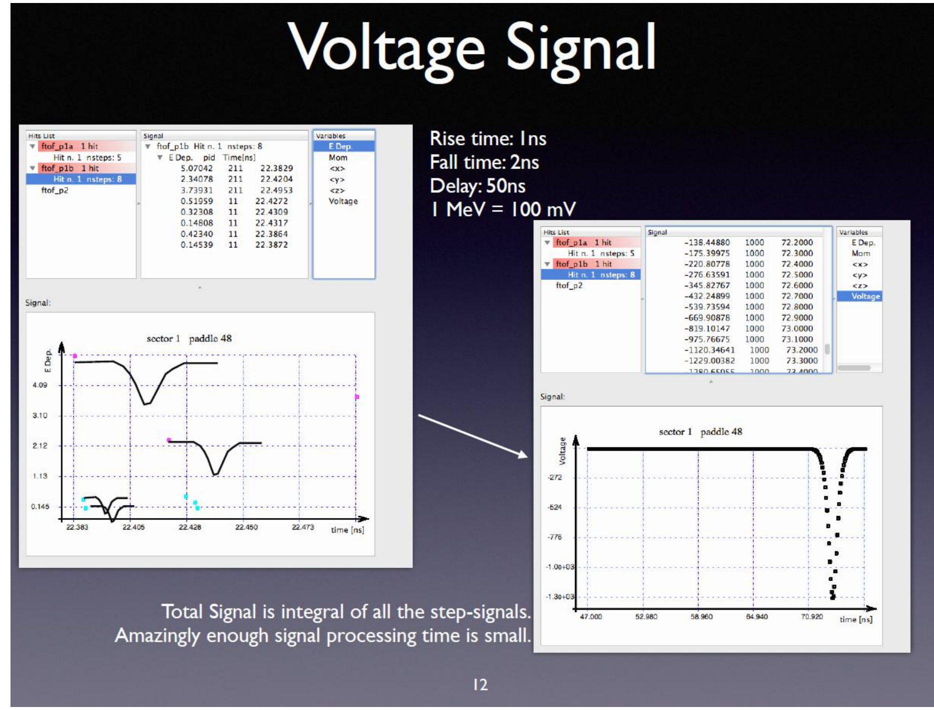


Other features of GEMC 2.x

- GEMC 1.x to 2.x, a major code rewrite
- use factory method as much as possible, easy to add plug-in and expand functionality
- Field map
  - Field info embedded in field map file, no separated definition needed
  - Map Reading 3-4x faster Swimming 30% faster
- Modular physics list (hadron+EM+optical)
- Built-in hit type “flux” and step-by-step hit info
- customized event generator input
- All codes are packaged by a single release number “jlab_version” with streamlined installation
- Took a lot suggestion from SoLID and MEIC simulation and we contribute to its code and structure also

Other features of GEMC 2.x





- Updated GUI
- voltage signal and FADC support

GEMC future

- Large user base: CLAS12, MEIC, SoLID, many smaller projects
- In github now, "github.com/gemc", has nice tool for debuging, feature adding, contributing
- Started "gemc collaboration" to gain input and help from developers and users

Summary

- status
  - EC, GEM, MRPC,SPD,HGC have initial implementation at various stages
  - LGC is under work (in a modified GEMC 1.x now)
- advantage
  - fast MC and full MC within one framework
  - standalone and full simulation within one framework
  - Things can evolve with new input format, new reconstruction/analysis framework

backup

GEMC Update: Outline

- GEMC 1.x to 2.x, a major code rewrite
- use factory method as much as possible, easy to add plug-in and expand functionality
- Field map
  - Field info embedded in field map file, no separated definition needed
  - Map Reading 3-4x faster Swimming 30% faster
- Modular physics list (hadron+EM+optical)
- Updated GUI
- geometry and parameters, material, optical properties, hit process and output are all external
- Built-in hit type “flux” and step-by-step hit info
- New features like customized event generator input, voltage signal and FADC support
- Took a lot suggestion from SoLID and MEIC simulation and we contribute to its code and structure also

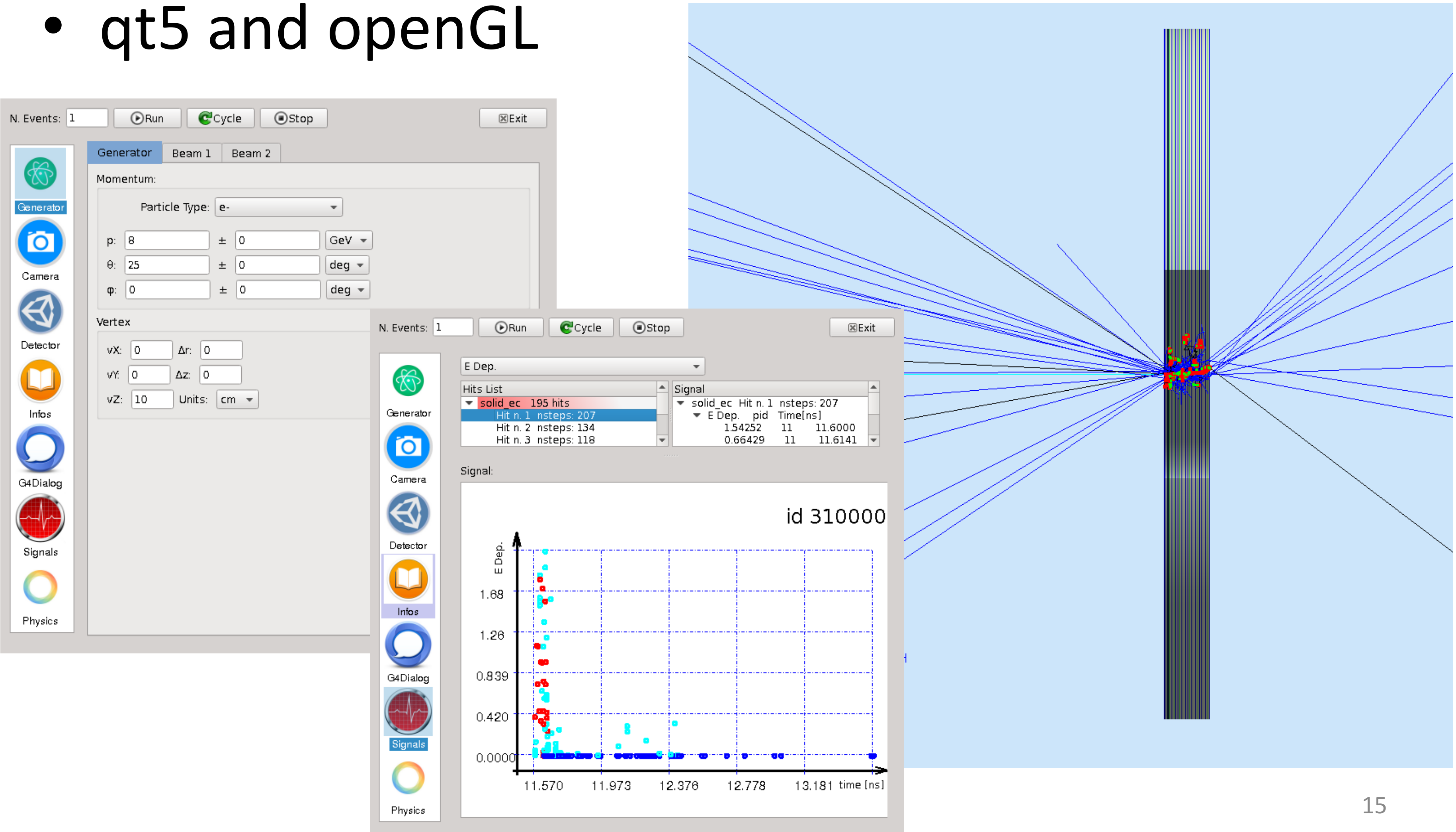
GEMC Update: installation

- software installation streamlined by a set of scripts
- part of the general jlab software framework (including jana, ccdb and more)
- Everything in release package, no SVN or github download
- a single version control by env "Jlab_version"
- Just an "App" to download on Mac, no install needed

Jlab_version	1.0	1.1	1.2	devel
GEMC	1.8	2.1	2.2	2.3?
geant4	9.5	9.5/9.6	10.0	10.1?

GEMC update: GUI

- qt5 and openGL



GEMC update: Input

- All generators are still fully independent, interface by txt file in customized LUND format
- Fully pass customized header info into output

GEMC

Home Documentation Gallery Support Downloads Web Interface

THE LUND FORMAT

The LUND format is a text file that can be used feed gemc with physics events.
It has a *header* to input the number N of generated particle for each event and other kinematic properties.
Then it has N lines describing each particle property, as described in these tables.

In BOLD, the quantities actually used by GEMC.

A * indicate a quantity not actually used by GEMC, but kept in the output stream. User defined meanings (not LUND) can be assigned to these variables.

Header Infos

Column	Quantity
1	**Number of particles**
2*	Number of target nucleons
3*	Number of target protons
4*	Target Polarization
5	**Beam Polarization**
6*	*x*
7*	*y*
8*	*W*
9*	*Q2*
10*	*nu*

Particle Infos

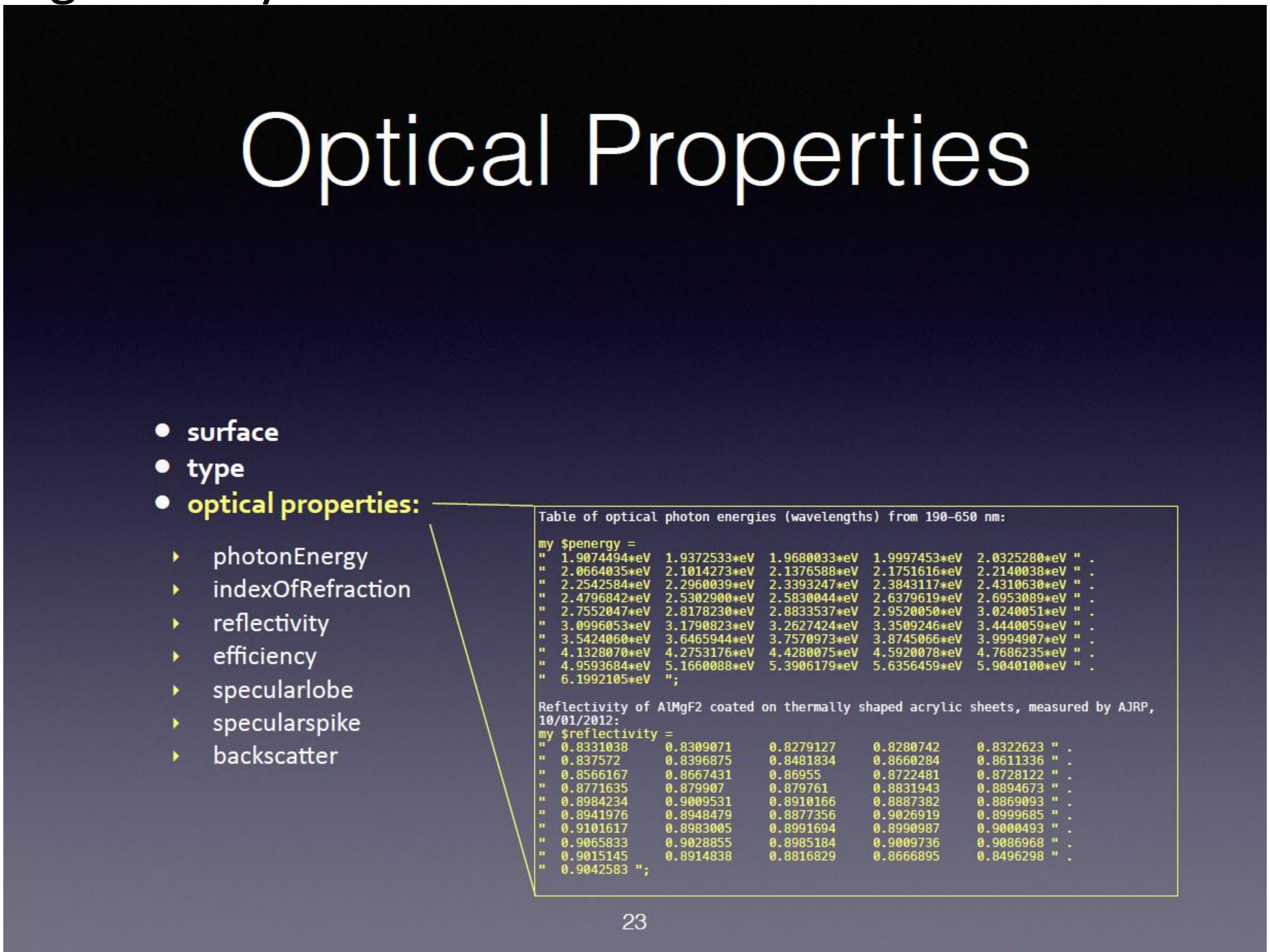
Column	Quantity
1	index
2	charge
3	**type**(=1 is active)
4	**particle id**
5	parent id (decay bookkeeping)
6	daughter (decay bookkeeping)
7	$\mathbf{p_x}$ [GeV]
8	$\mathbf{p_y}$ [GeV]
9	$\mathbf{p_z}$ [GeV]
10	E [GeV]
11	mass (not used)
12	**x vertex** [cm]
13	**y vertex** [cm]
14	**z vertex** [cm]

Example of 2 DVCS events (ePγ)

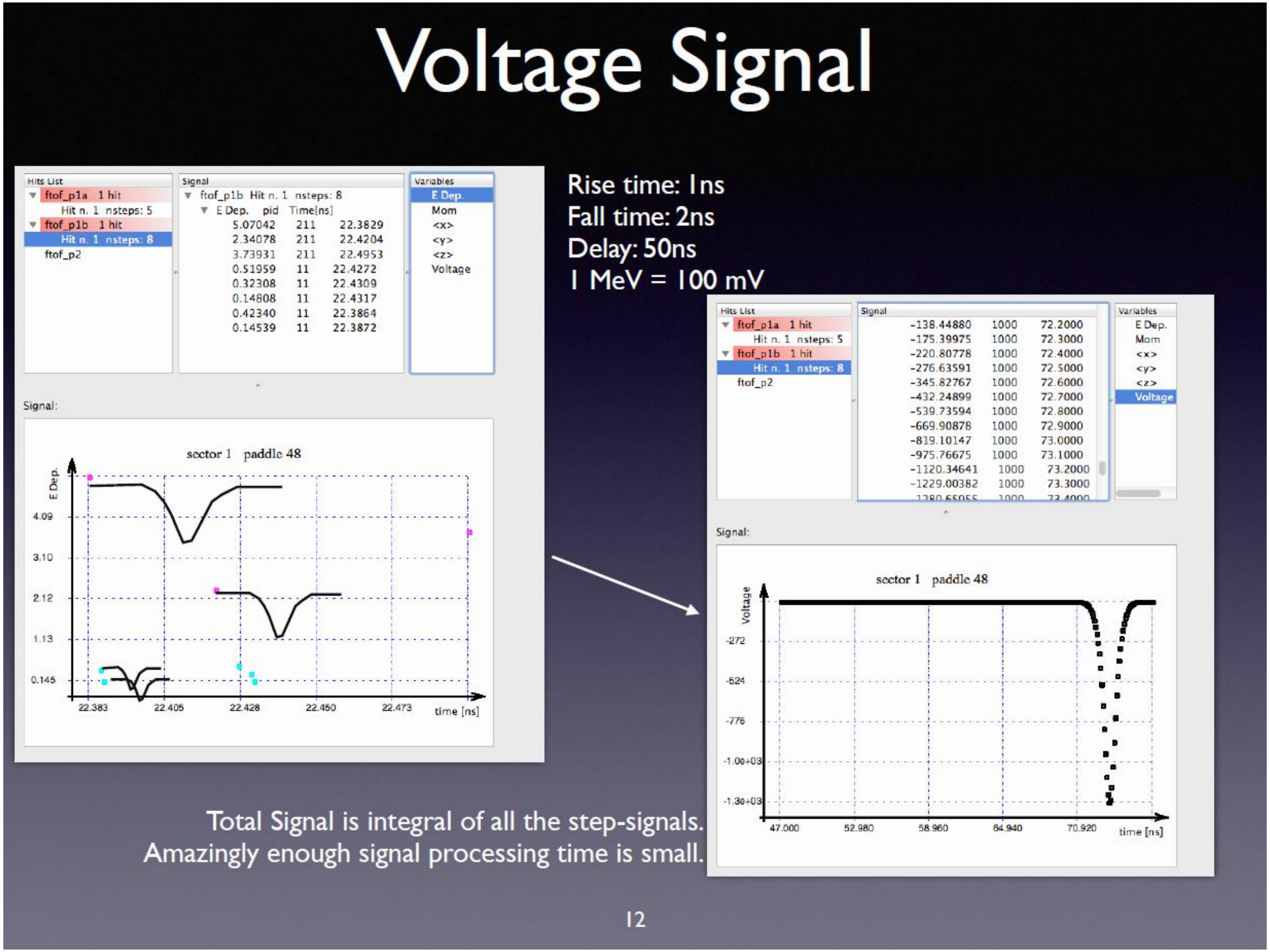
```
3  1.  1.  0 -1 0.209   0.336   6.373   1.448  -1.000
 1 -1.  1   11  0  0    0.9636  -0.1675   7.2357   7.3015   0.0005   0.0000 0.0000 0.0000
 2  1.  1 2212  0  0   -0.6536   0.0604   0.3367   1.1935   0.9383   0.0000 0.0000 0.0000
```

GEMC Update: Optical

- All properties defined external in “table” format similar to geometry



GEMC Update: Voltage Signal



GEMC Update: Outlook

- FADC banks will emulate FADC output, including translation tables
- Hit process routines will be a plugin - completely independent from GEMC, to be loaded at run time. The routines can be stored together with the geometry / materials scripts.
- multithreading, following G4
- C++ 11

EC simulation with solid_gemc

- solid - Revision 746: /subsystem/ec/ec_solid_gemc
- ..
- config_solid_PVDIS_ec_forwardangle.dat — configuration file for generating
- readme
- solid_PVDIS_ec_forwardangle.gcard — input file to run simulation
- solid_PVDIS_ec_forwardangle.pl — file to generate geomtry,bank,hit
- solid_PVDIS_ec_forwardangle__bank.txt — generated bank
- solid_PVDIS_ec_forwardangle__geometry_Original.txt — generated geometry
- solid_PVDIS_ec_forwardangle__hit_Original.txt — generated hit
- solid_PVDIS_ec_forwardangle__materials_Original.txt — generated material
- solid_PVDIS_ec_forwardangle__parameters_Original.txt — parameters defines geometry
- solid_PVDIS_ec_forwardangle_real.pl — file to generate geometry
- solid_ec_bank.pl — file to generate bank
- solid_ec_hit.pl — file to generate hit
- solid_ec_materials.pl — file to generate material
- solid_slice.vis

- Purple and orange files needed to generate files for simulation
- Red files needed to run simulation
- Red and orange files are in "table" format, they can be in
  - txt file
  - Database (mysql now, CCDB soon)
  - expanded to more sources

An example line of a txt file

solid_PVDIS_ec_forwardangle_real_shower | root |solid_PVDIS_ec_forwardangle_real_shower | 0*cm 0*cm 350*cm | 0*deg 0*deg 0*deg | ff0000 | Tube | 110*cm 365*cm 21.728*cm 0*deg 360*deg | G4_AIR | no | 1 | 1 | 1 | 1 | 1 | no | no | no

Customized hit process routine "solid_ec"

```
insert_bank_variable(\%configuration, $bankname, "bankid", $bankId, "Di", "$bankname bank ID");
insert_bank_variable(\%configuration, $bankname, "pid",    1,   "Di", "ID of the first particle entering the sensitive volume");
insert_bank_variable(\%configuration, $bankname, "mpid",   2,   "Di", "ID of the mother of the first particle entering the sensitive volume");
insert_bank_variable(\%configuration, $bankname, "tid",    3,   "Di", "Track ID of the first particle entering the sensitive volume");
…………………
insert_bank_variable(\%configuration, $bankname, "id",     24,  "Di", "id number");
insert_bank_variable(\%configuration, $bankname, "hitn",   99,  "Di", "Hit Number");
```

solid_ec_bank.pl

```
dgtz["pid"] = (double) aHit->GetPID();
dgtz["mpid"] = (double) aHit->GetmPID();
dgtz["tid"] = (double) aHit->GetTId();
…………………
dgtz["id"] = id;
dgtz["hitn"] = hitn;
```

solid_ec_hitprocess.cc

- Totally flexible to any raw and digitized hit processing and output, fit any level of simulation and digitization need
- As far as the two match each other to give consistent result, "solid_gemc" need recompile if any change

Hit process control

solid_ec_hit.pl

```
$hit{"name"} = "solid_ec";
$hit{"description"} = "solid ec hit definition";
$hit{"identifiers"} = "id";
$hit{"signalThreshold"} = "0*MeV";
$hit{"timeWindow"} = "400*ns";
$hit{"prodThreshold"} = "1*mm";
$hit{"maxStep"} = "1*cm";
$hit{"delay"} = "10*ns";
$hit{"riseTime"} = "1*ns";
$hit{"fallTime"} = "1*ns";
$hit{"mvToMeV"} = 100;
$hit{"pedestal"} = -20;
```

- Fine tuning hit processing without source code change
- No need to recompile solid_gemc

External parameters

solid_PVDIS_ec_forwardangle__parameters_Original.txt

Nlayer | 194 | | Nlayer | - | - | - | - | - | -

Thickness_lead | 0.05 | cm | Thickness_lead | - | - | - | - | - | -

Thickness_scint | 0.15 | cm | Thickness_scint | - | - | - | - | - | -

Thickness_gap | 0.024 | cm | Thickness_gap | - | - | - | - | - | -

Thickness_shield | 1.0274 | cm | Thickness_shield | - | - | - | - | - | -

Thickness_prescint | 2 | cm | Thickness_prescint | - | - | - | - | - | -

Thickness_support | 2 | cm | Thickness_support | - | - | - | - | - | -

z_shower | 350 | cm | z_shower | - | - | - | - | - | -

Rmin | 110 | cm | Rmin | - | - | - | - | - | -

Rmax | 365 | cm | Rmax | - | - | - | - | - | -

Sphi | 0 | deg | Sphi | - | - | - | - | - | -

Dphi | 360 | deg | Dphi | - | - | - | - | - | -

- It can take source from survey data
- And can be part of calibration database

Other things

- Record simulation condition
  - GEMC record all input options into EVIO file
  - We can think of ways to record detector related input (as SVN or github version or database entry with index like run number? It will depends on where we store them)
- Output file format
  - "evio2root", convert evio to root tree, included with framework
  - "clas-root", read evio like a root tree, will include
  https://userweb.jlab.org/~gavalian/clas12docs/sphinx/html/rootio/introduction.html
- Documentation
  - Nice GEMC tutorials available
  - Doxygen for source code
  - Wiki https://hallaweb.jlab.org/wiki/index.php/Solid_Software