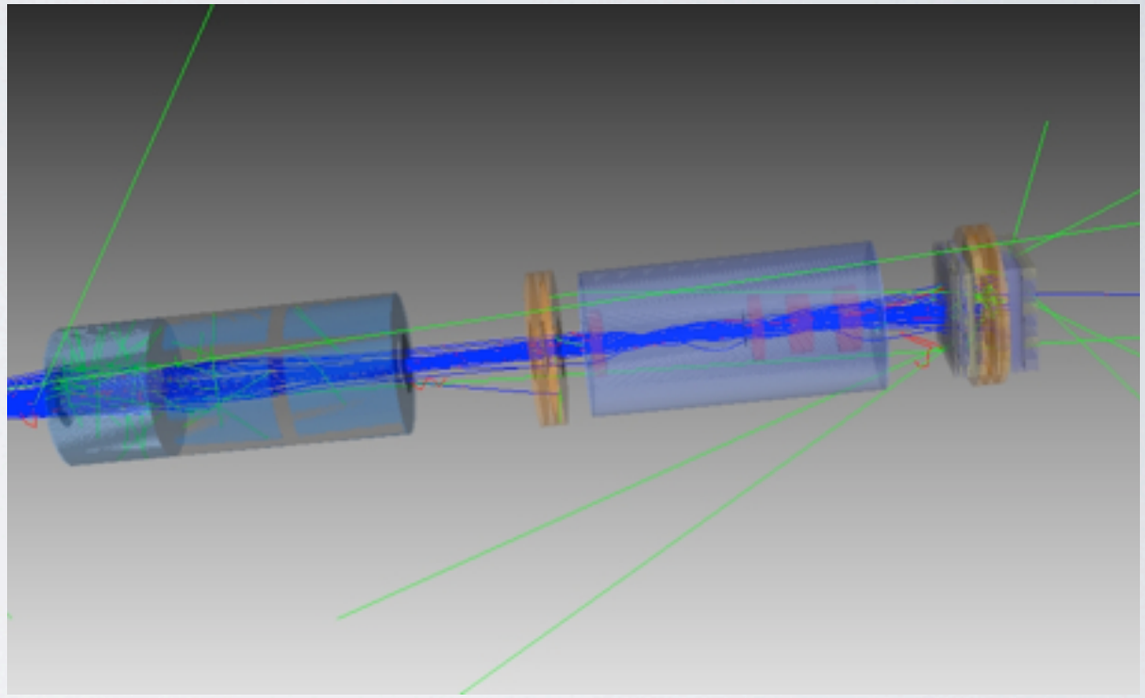
GEMC:
GEANT4 SIMULATION
AT JLAB

Simulation is the imitation of a real process or system

GEANT4 Library



Beam in fields, materials

- Particles through matter
- Several physics environment
- (Electro)Magnetic Fields
- Geometry Navigation

GEANT4 is a C++ Library

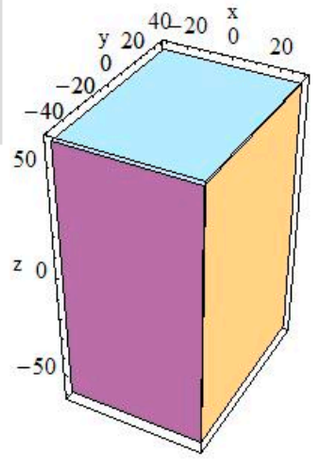
Does require some programming language experience
Does require some C++ knowledge
OO Programming skills much preferable

However

Physicist... we aren't necessarily the best programmers...
We do know the geometry and detector response well

GEANT4 is a C++ Library

```
G4Box(const G4String& pName,
            G4double  pX,
            G4double  pY,
            G4double  pZ)
```



[Rotate the Picture]

In the picture:

```
pX = 30, pY = 40, pZ = 60
```

by giving the box a name and its half-lengths along the X, Y and Z axis:

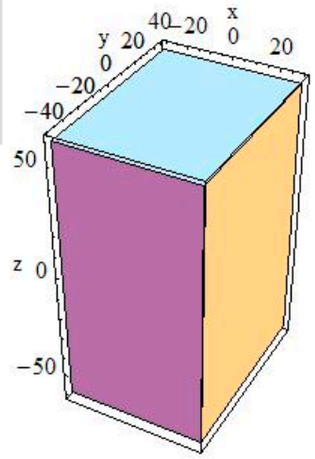
pX	half length in X	pY	half length in Y	pZ	half length in Z

- Write Geometry in the code
- Write Response Function in the code
- Implement Magnetic Fields in the code
- Incorporate Digitization
- Input and Output

Do this for hundreds (thousands) of detector elements

GEANT4 is a C++ Library

```
G4Box(const G4String& pName,
            G4double  pX,
            G4double  pY,
            G4double  pZ)
```



[Rotate the Picture]
In the picture:

pX = 30, pY = 40, pZ = 60

by giving the box a name and its half-lengths along the X, Y and Z axis:

pX	half length in X	pY	half length in Y	pZ	half length in Z

Solid

Logical

Physical

Sensitivity

Fields

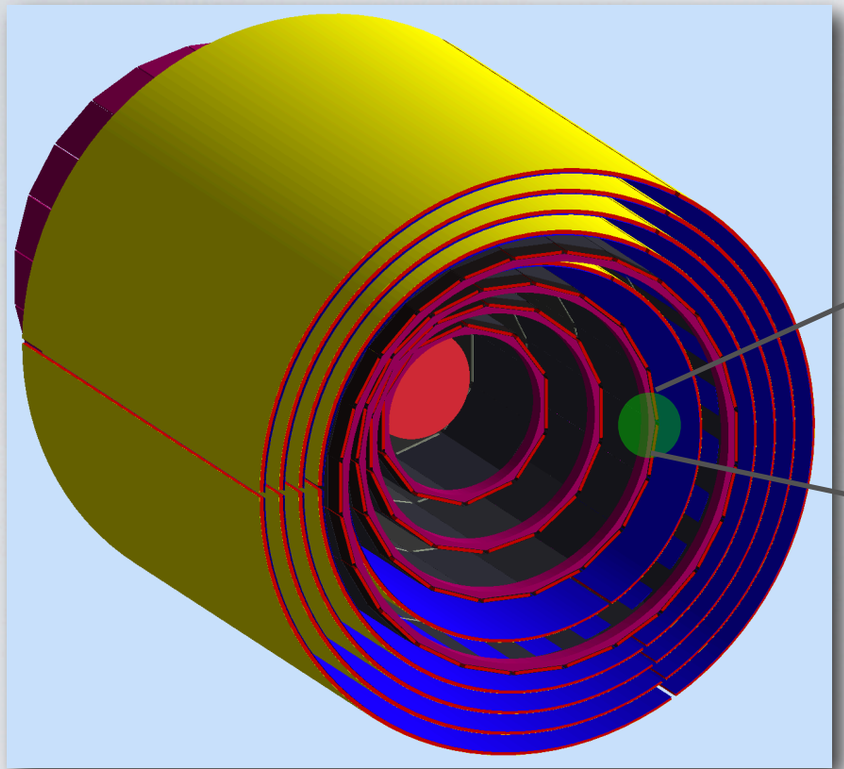
IN

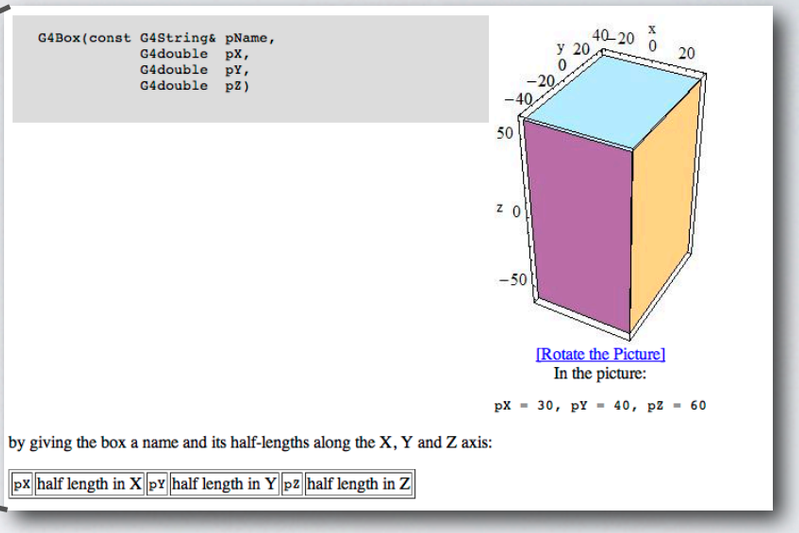
RUN

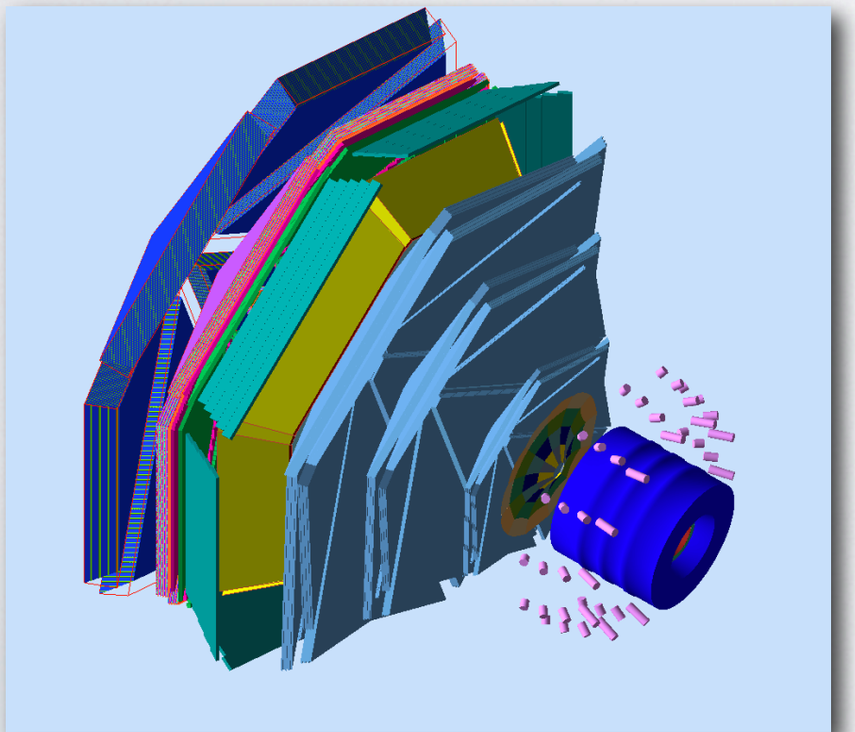
Out

These steps are well defined

Realistic Simulation







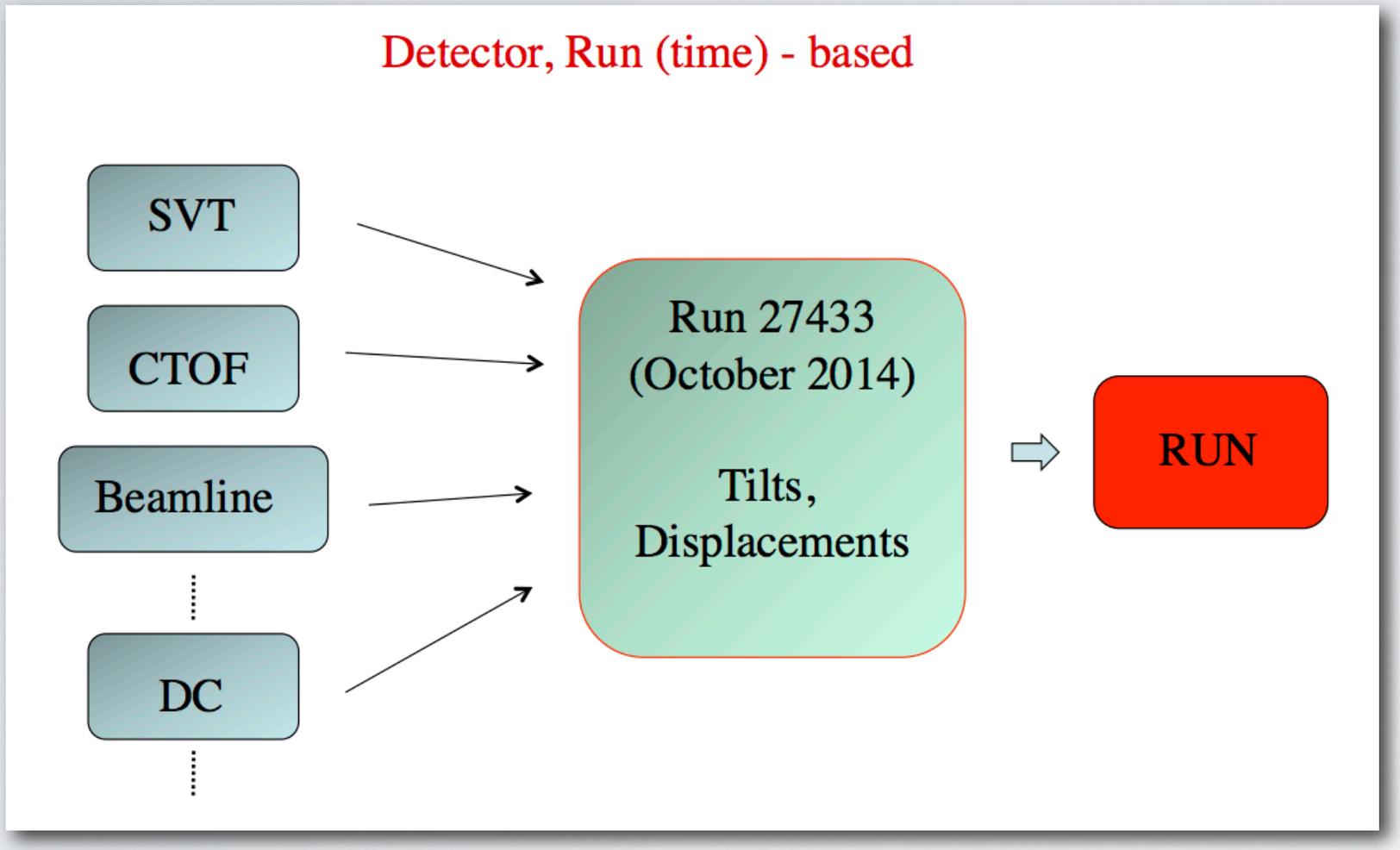
Thousands of detector elements

Geometry, digitization Forever Changing

Development, Distribution Nightmare

Copy all parameters to Reconstruction, Event Display

What is an IDEAL simulation environment



We want to solve these problems:

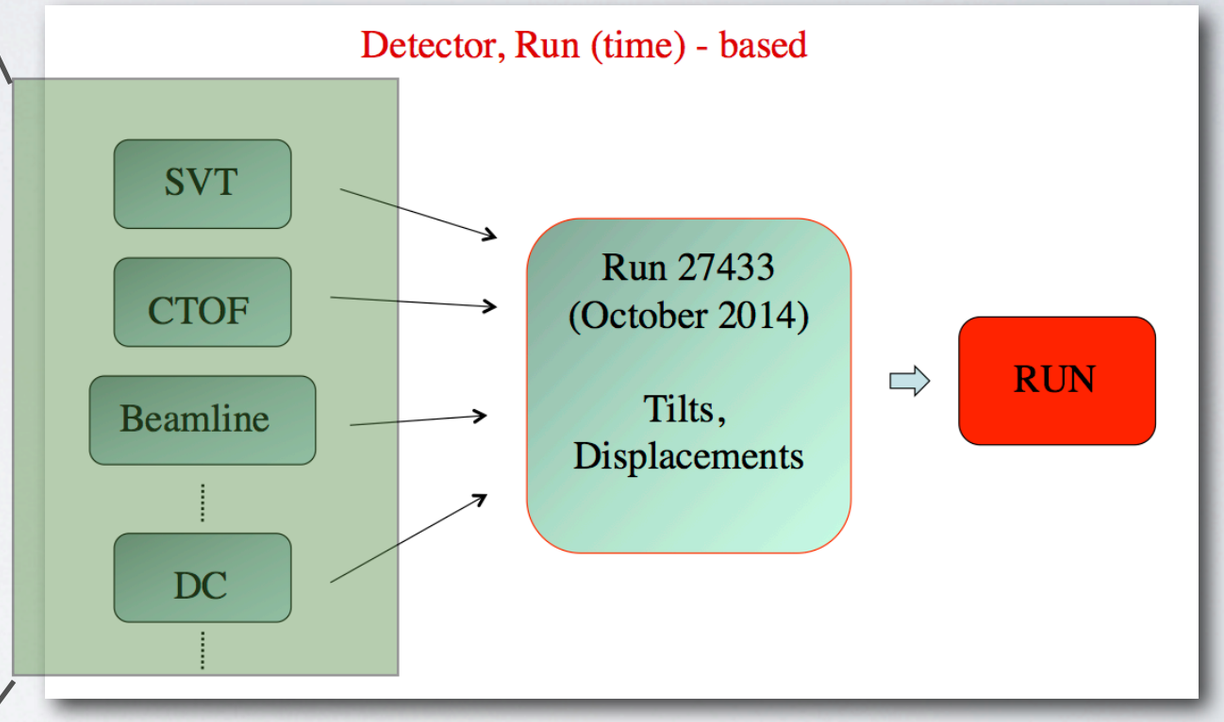
1. Users should have to deal ONLY with geometry, fields, response function parameters - NOT how they interface with geant4

2. All parameters should be in a database in common with reconstruction, calibration, visualisation

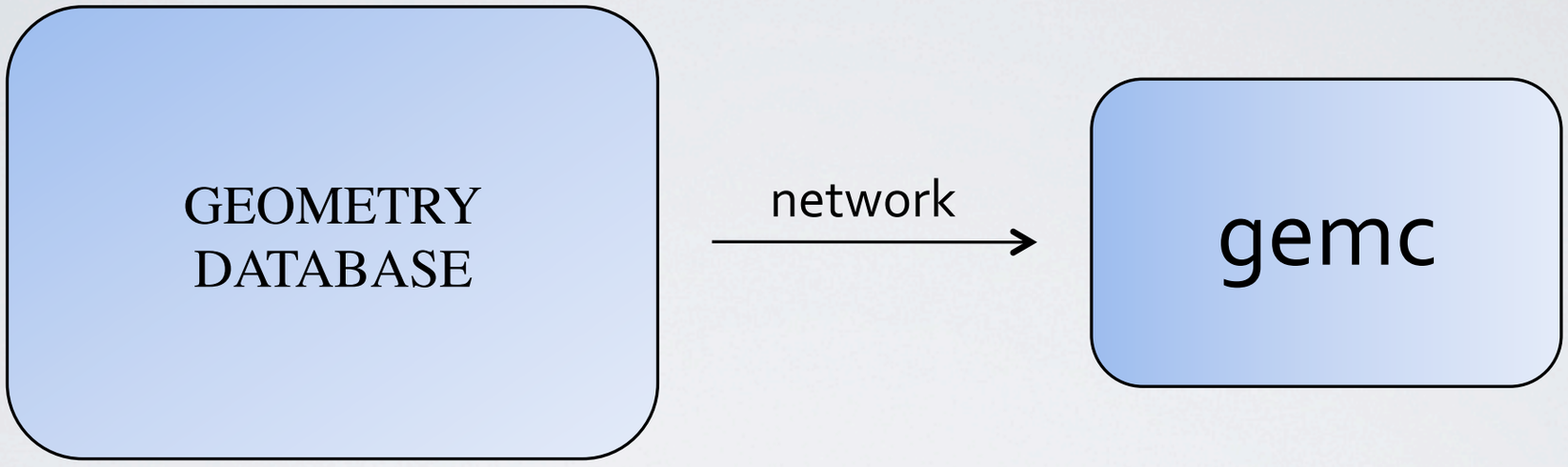
GEant4 Monte-Carlo

Database
MYSQL, GDML
(soon SQLITE)

The Geometry
Sensitivity
Digitization
Output Format
Magnetic Fields
Materials



GEMC Geometry

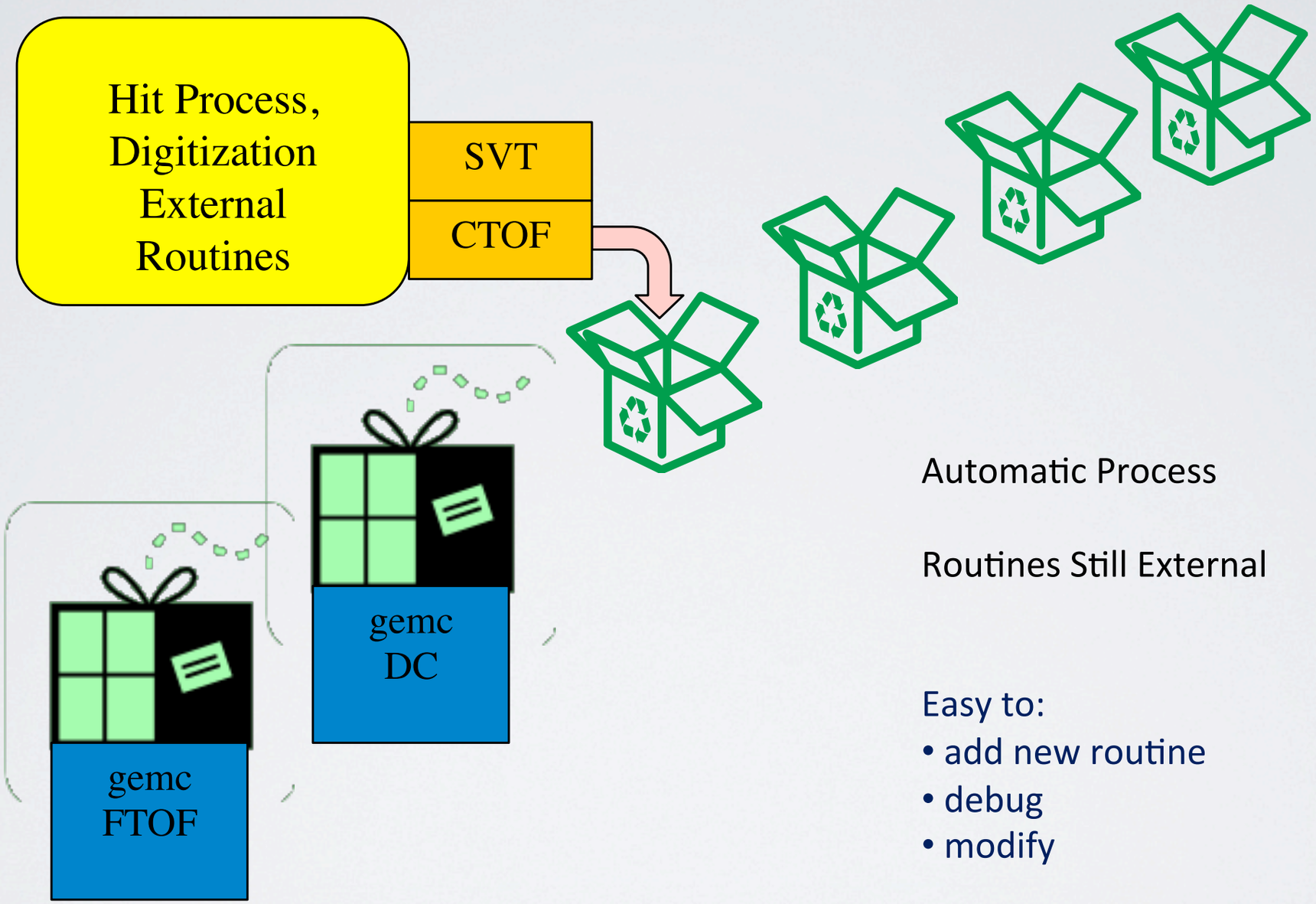


Geometry Stored on a MYSQL DB
"CLAS12 Official" geometry at JLAB

Solves the previous problem PLUS:

No need to recompile code

GEMC Digitization



Automatic Process

Routines Still External

Easy to:
- add new routine
- debug
- modify

OO approach ensure bugs are virtually non-existant

Digitization Flowchart

EDEP, Time

(Paddle, DC cell, Strip)

Calibration:

Cable Lengths
Att. Lengths
t vs x
nphe
paddle to paddle

DB

Geometry,
Detector Type

Not necessarily
in Calibration

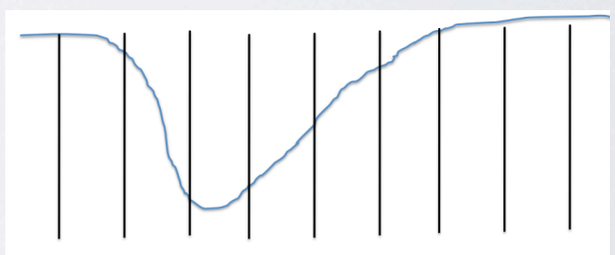
photomultiplier type
(quantum efficiency)

DC hit distance to cables

ADC, TDC

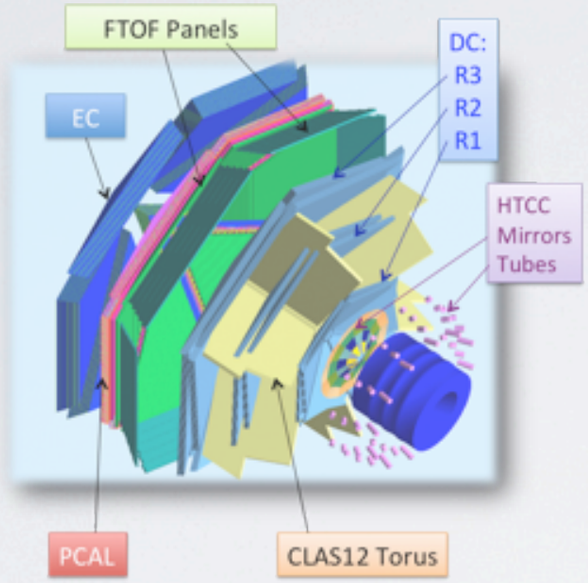
Signal F(t)

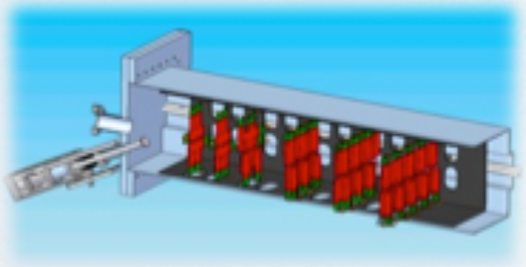
FADC



GEMC USE

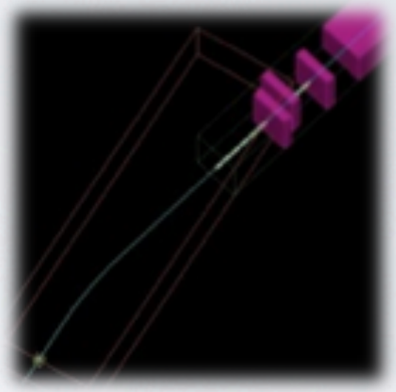
CLAS12 Detector

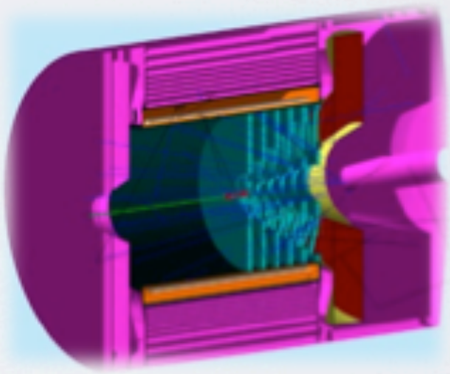




HPS Beamline

GLUEX Beamline

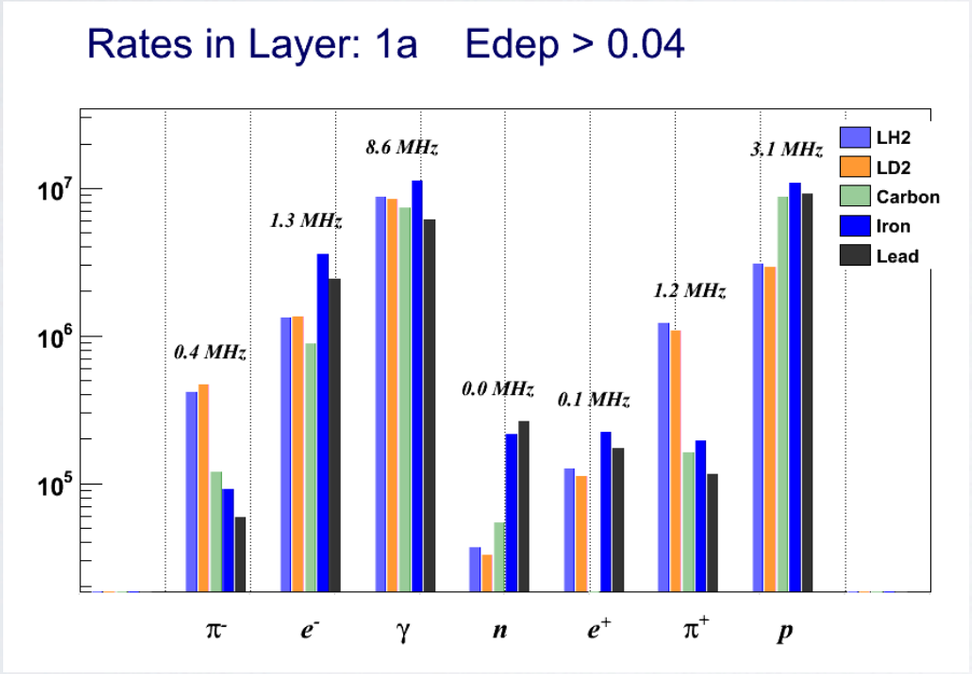


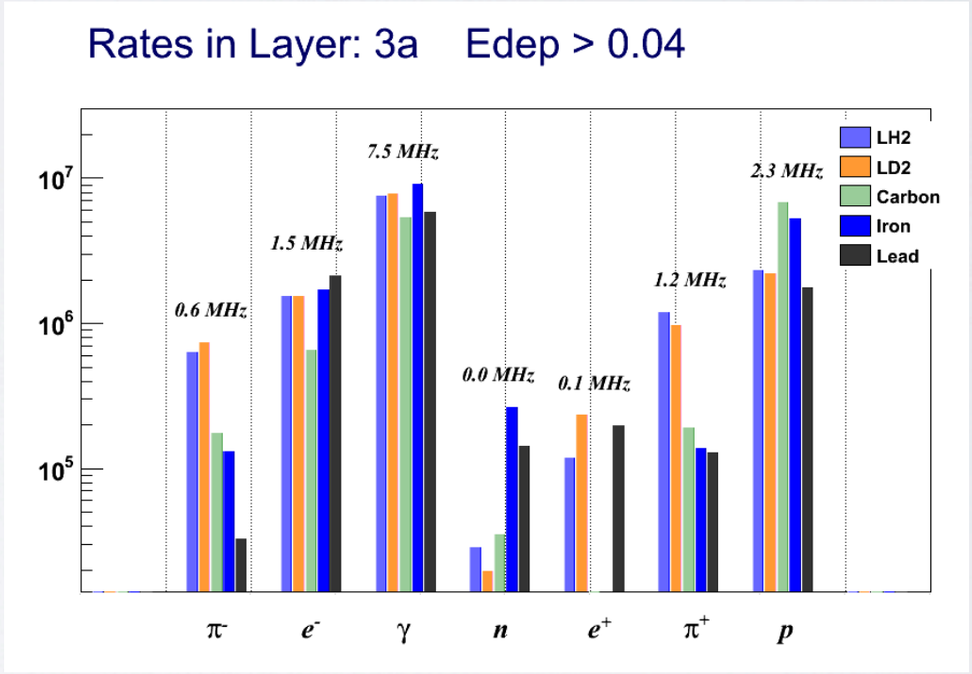


Solid: Hall-A 12

GEMC USE

E > 0.04 MeV
All targets
Rate written on top is on LH2





GEMC USE

1 sensor = 3.26 g, 46.88cm^2

SL1:	3 sensor	10 sectors
SL2:	3 sensors	14 sectors
SL3:	3 sensors	18 sectors

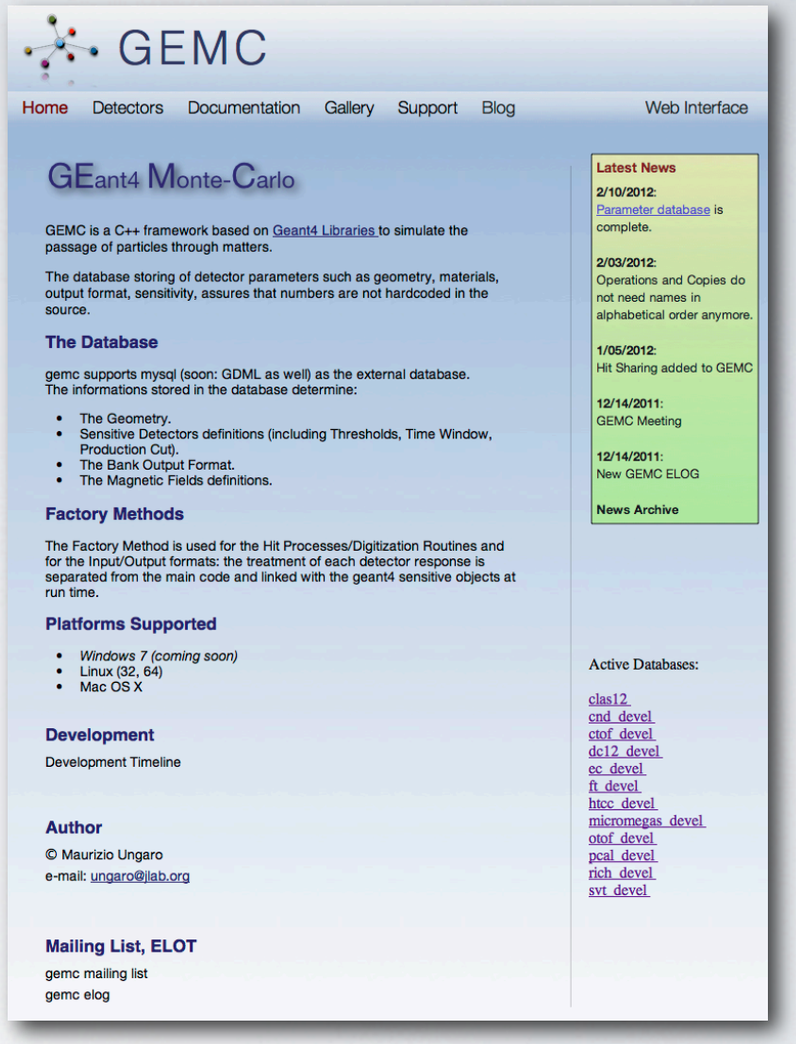
Layer 1a

target	GeV/s	GeV/(s cm2)	mrad/s	mrad/(scm2)	rad/year	rad/(year cm2)
lh2	20325	15.054	6.244	0.00462	196939	145
ld2	20332	15.060	6.247	0.00462	197013	145
C	32220	23.865	9.899	0.00733	312193	231
Fe	52182	38.650	16.032	0.01187	505612	374
Pb	66000	48.885	20.278	0.01501	639498	473

Documentation

gemc.jlab.org

Doxygen
Installation
HowTos
Detectors



https://clasweb.jlab.org/elog-gemc/

Mailing List: gemc_software

Bug Report: Mantis

Doxygen