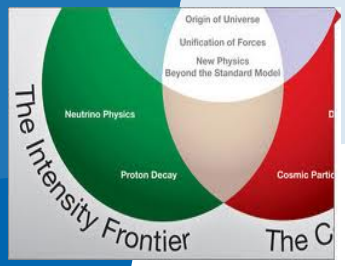
AMGATK update

Hans Wenzel

January 9th 2014

U.S. DEPARTMENT OF ENERGY

Fermilab



Outline

Calorimeter and Tracker Simulation

- Plans/Status from last time
  - Sampling calorimeter from last time
- Motivation
- Development environment
- How to tell art what we are producing
- Extending the gdml schema (colors)
- Examples:
  - PbF2 Crystal
  - Dual read out crystal calorimeter
- Plans

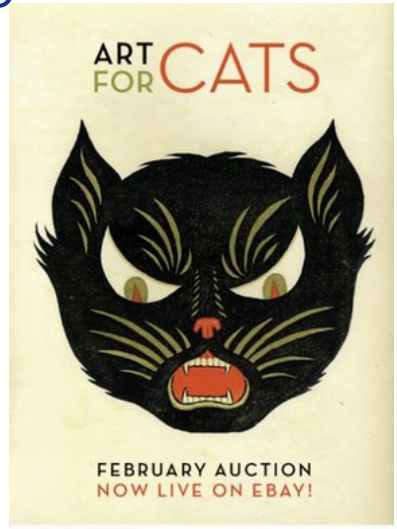
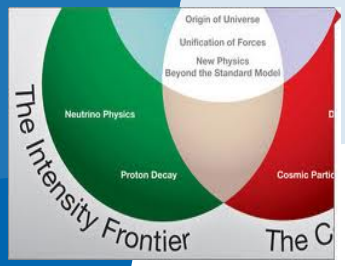


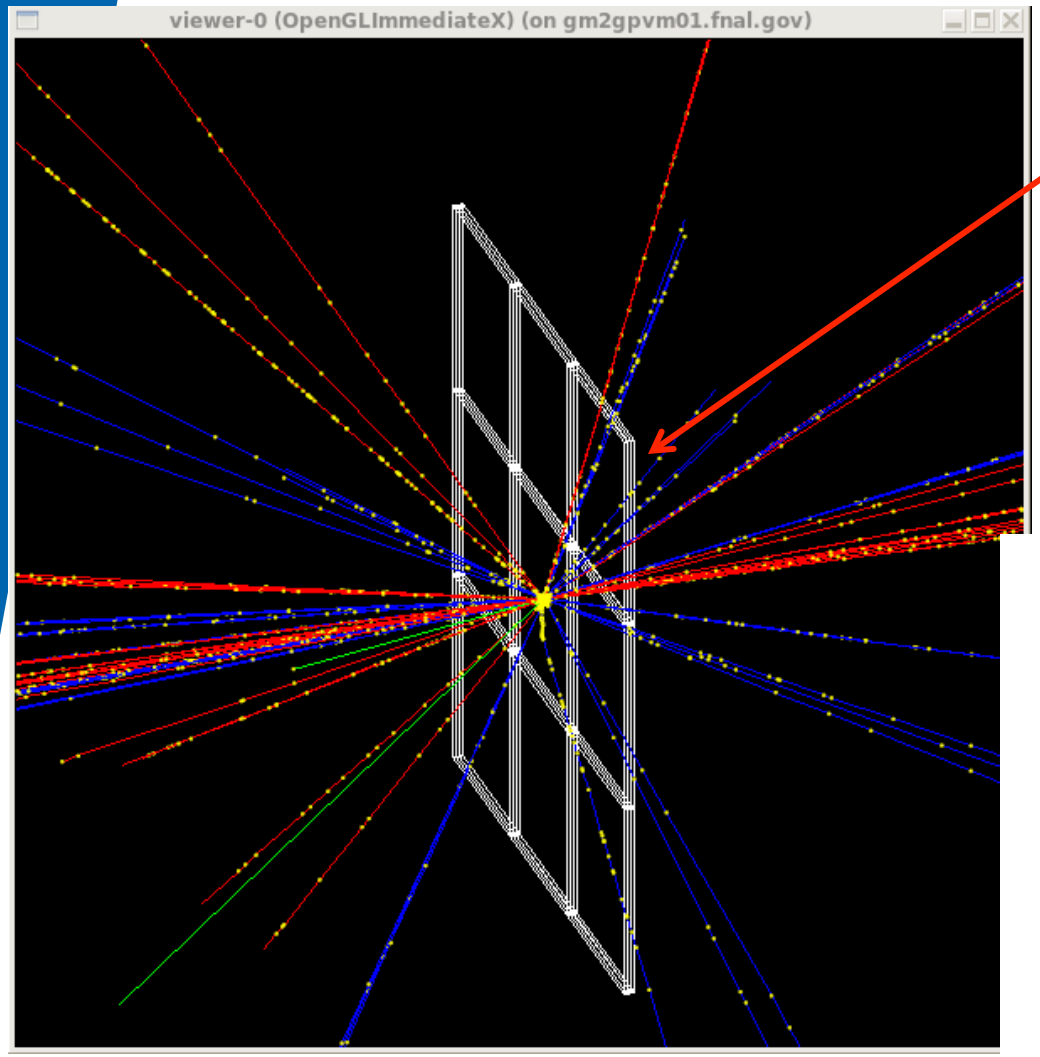
Image © Adam McCauley



Plan from last time

- Clean up code and put int artg4/artg4example git repository (done)
- Implement scheme to store/retrieve/analyze the data (Hits) (show)
- Import all SD/Hits classes, Analysis from Cats. Make sure we can do detector R&D (show examples)
- Add Color
- Integrate numi beamline physics list (not yet)

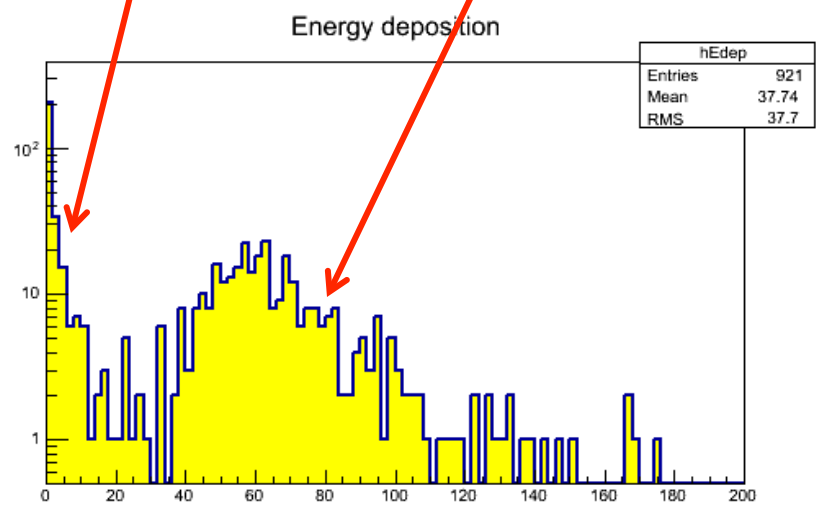
Example tiled PB/scintillator calorimeter from last time

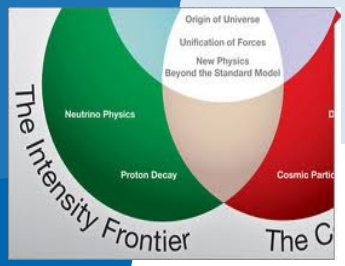


No visual Attributes?
4mm x 30cm x 30 cm Pb,
1mm x 30cm x 30 cm Sz

Pb Absorbers

Szintillator





Motivation: Modular system

build detector from predefined components

Currently implemented
PhotonSD
TrackerSD
CalorimeterSD
DRCalorimeterSD

Particle ID

Cerenkov
Radiator

Tracker

EM

Hadronic

μ

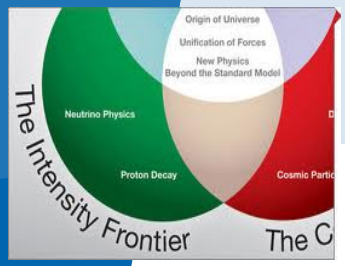
BeamDump

PhotonSD
produces ArtPhotonHits

TrackerSD
produces ArtTrackerHits

DRCalorimeterSD
produces ArtDRCalorimeterHits

StoppingCalorimeterSD
Pro. ArtCalorimeterHit



Development environment

Idea: Start with a working example

development machine:

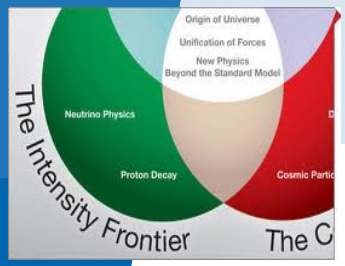
gm2gpvm01.fnal.gov

Software Versions:

GEANT4_VERSION=v4_9_5_p01
ROOT_VERSION=v5_34_01
art v1_02_04
artg4 v0_0_1

git repositories:

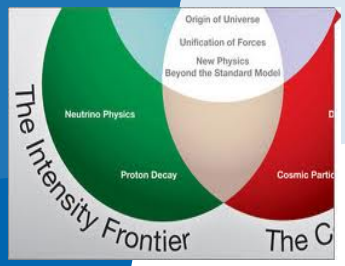
https://cdcvs.fnal.gov/redmine/projects/artg4/repository
https://cdcvs.fnal.gov/redmine/projects/artg4example/repository/show?rev=develop



Development environment

Setup not complete!

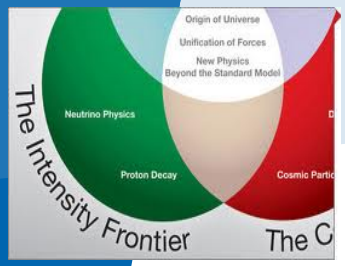
```
source /gm2/app/software/prod/g-2/setup
setup gm2 v201211_1 -q e2:debug
cd gm22
source localProducts_v201211_1_debug-e2/setup
cd build
source gm2d setup_for_development
export G4LEVELGAMMADATA=/gm2/app/software/prod/external/g4photon/v2_2/NULL/PhotonEvaporation2.2
export G4REALSURFACEDATA=/gm2/app/software/prod/external/g4surface/v1_0/NULL/RealSurface1.0/
```



Tell art what we are producing

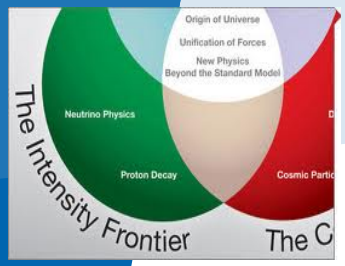
Change calling sequence

```
// We need all of the services to run @produces@ on the data they will store. We do this
// by retrieving the holder services.
art::ServiceHandle<ActionHolderService> actionHolder;
art::ServiceHandle<DetectorHolderService> detectorHolder;
  detectorHolder->initialize();
  //hjw:
  //detectorHolder -> callArtProduces(this);
  // Build the detectors' logical volumes
  detectorHolder -> constructAllLVs();
  // And running @callArtProduces@ on each
  actionHolder -> callArtProduces(this);
  detectorHolder -> callArtProduces(this);
```



Tell art what we are producing

```
void artg4example::GDMLDetectorService::doCallArtProduces(art::EDProducer * producer) {
    // Tell Art what we produce, and label the entries
    std::vector<std::pair<std::string, std::string> >::const_iterator cii;
    for (cii = DetectorList.begin(); cii != DetectorList.end(); cii++) {
        if ((*cii).second == "DRCalorimeter") {
            std::string identifier = myName() + "-" + (*cii).first + "-" + (*cii).second;
            producer -> produces<myDRCaloArtHitDataCollection>(identifier);
        } else if ((*cii).second == "Calorimeter") {
            std::string identifier = myName() + "-" + (*cii).first + "-" + (*cii).second;
            producer -> produces<myCaloArtHitDataCollection>(identifier);
        } else if ((*cii).second == "PhotonDetector") {
            std::string identifier = myName() + "-" + (*cii).first + "-" + (*cii).second;
            producer -> produces<myPhotonArtHitDataCollection>(identifier);
        } else if ((*cii).second == "Tracker") {
            std::string identifier = myName() + "-" + (*cii).first + "-" + (*cii).second;
            producer -> produces<myTrackerArtHitDataCollection>(identifier);
        }
    }
}
```



Contents of the root file

- root
- PROOF Sessions
- ROOT Files
  - TestingOptics.root
    - EventHistory;2
    - EventHistory;1
    - MetaData;2
    - MetaData;1
    - Parentage;2
    - Parentage;1
    - Events;2
      - EventAuxiliary
      - art::TriggerResults_TriggerResults__processA.
      - artg4example::myDRCaloArtHitDatas_artg4Main_GDMLDetectorService-CalorimeterVolume-DRCalorimeter_processA.
      - artg4example::myPhotonArtHitDatas_artg4Main_GDMLDetectorService-PhotonVolume-PhotonDetector_processA.
    - Events;1

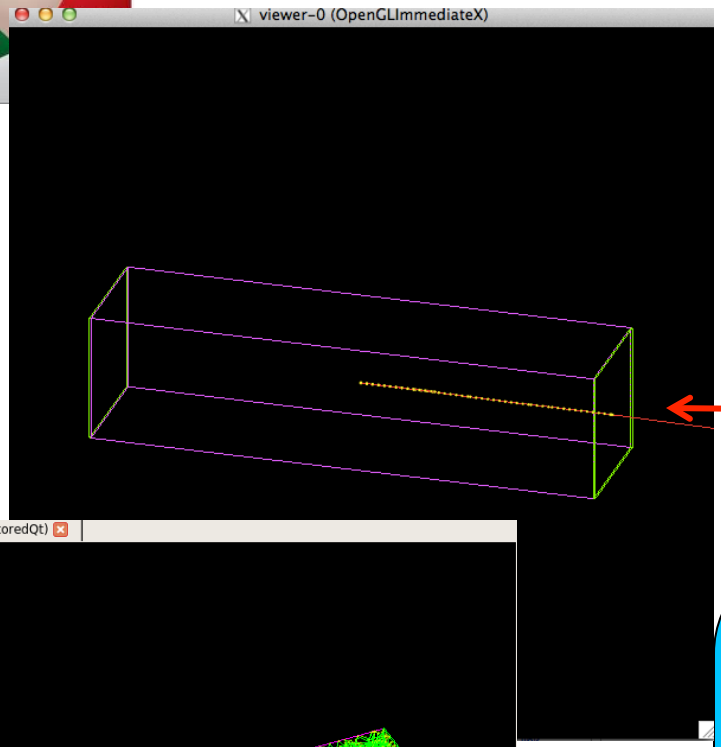
Add some color to your life:

```
<?xml version="1.0" encoding="UTF-8" ?>
<gdml_simple_extension xmlns:gdml_simple_extension="http://www.example.org"
                       xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
                       xs:noNamespaceSchemaLocation="SimpleExtension.xsd">
<!--
   //
   // PbF2 optical data from:
   // http://www.crystran.co.uk/
   // http://www.crystran.co.uk/lead-fluoride-pbf2.htm
   // the data values where estimated using:
   // Engauge Digitizer - Digitizing software
   // http://digitizer.sourceforge.net/
   // Note! the formula used to calculate the absorption length from the transmission
   // is not correct (too pessimistic) since it doesn't account for
   // fresnel reflection.
   //
-->

   <extension>
     <color name="test_color"  R="0.1"  G="0.2"  B="0.3"  A="1.0" />
     <color name="magenta"     R="0.0"  G="1.0"  B="0.0"  A="1.0" />
     <color name="green"       R="1.0"  G="0.0"  B="1.0"  A="1.0" />
   </extension>
```

```
auxiliary auxtype="SensDet" auxvalue="DRCalorimeter"/>
<!-- color extension element -->
<colorref ref="green"/>
</volume>
<volume name="PhotonVolume">
    <materialref ref="Si" />
    <solidref ref="PhotonBox" />
    <auxiliary auxtype="SensDet" auxvalue="PhotonDetector"/>
<colorref ref="magenta"/>
</volume>
```

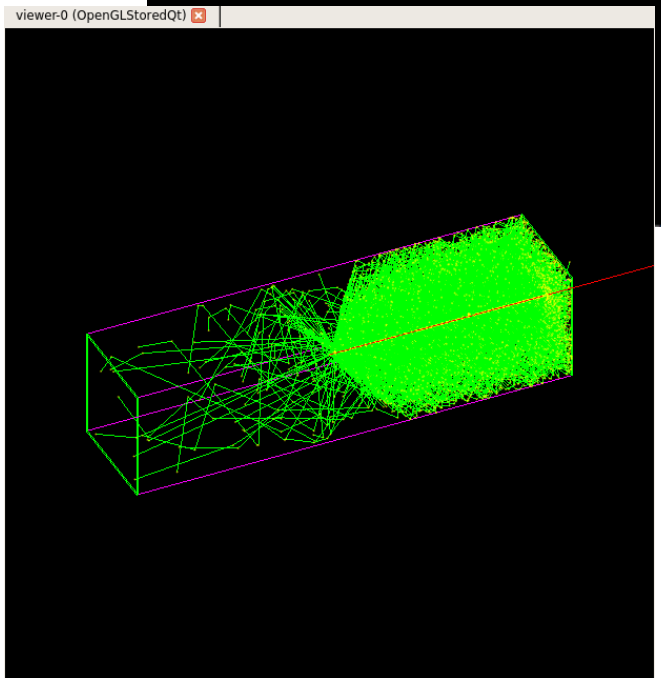
Gdml extension + modified parser for colors overkill
But good to know e.g. for Implementing Segmentation/ readout geometry or Fields
Extensions ignored by standard parser

Cerenkov light in PbF2 Crystal

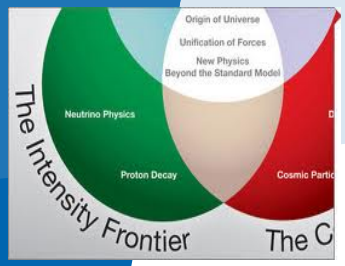


PbF2: no scintillation,
Good Cerenkov radiator,
used for em calorimeters
(e.g. g-2)

100GeV μ+

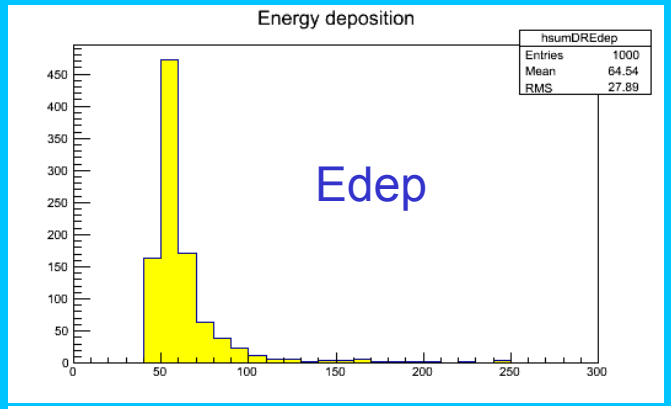



```
    PhysicsListHolder: {}
    PhysicsList: {
        PhysicsListName: "FTFP_BERT"
        DumpList: false
        enableCerenkov: true
        enableScintillation: false
        enableAbsorption: true
        enableRayleigh: false
        enableMieHG: false
        enableBoundary: true
        enableWLS: false
}
```



Result:

Bulk (DRCalorimeter):



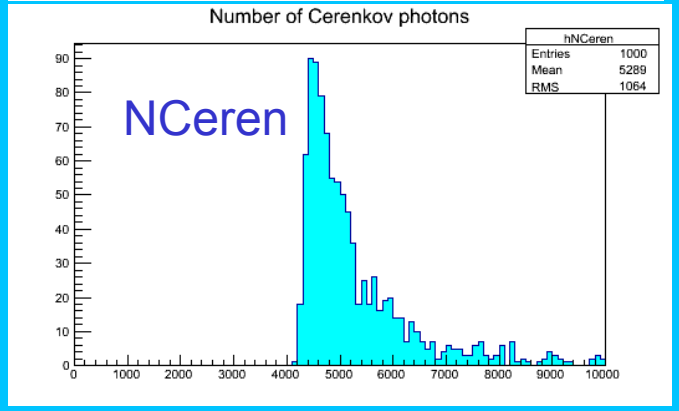
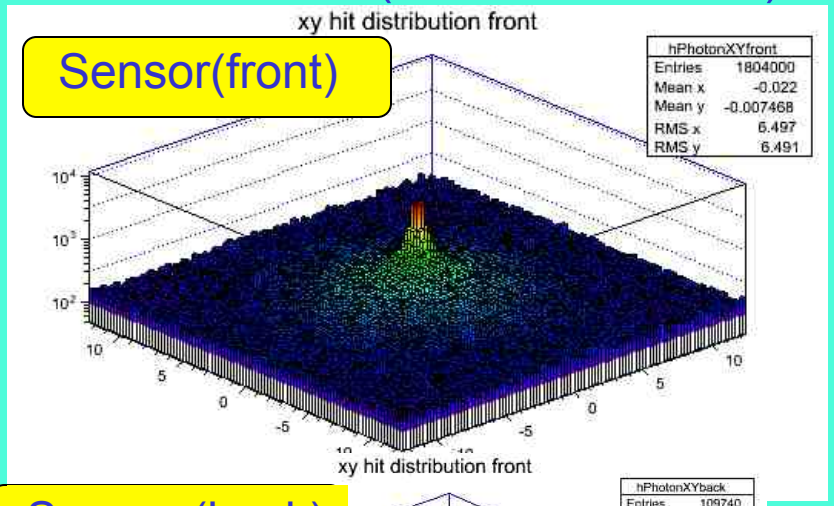
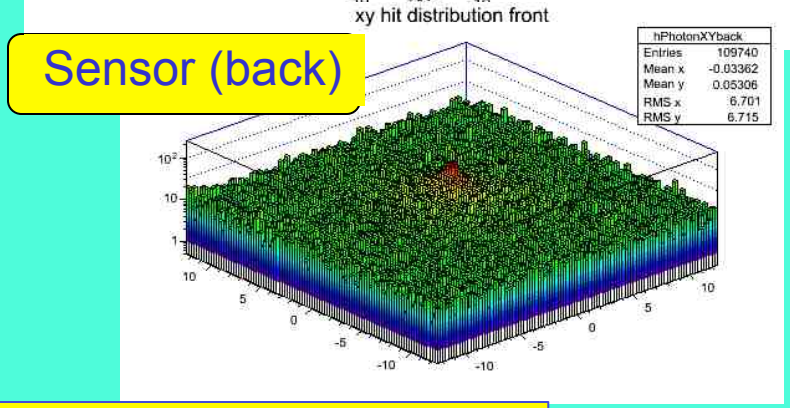
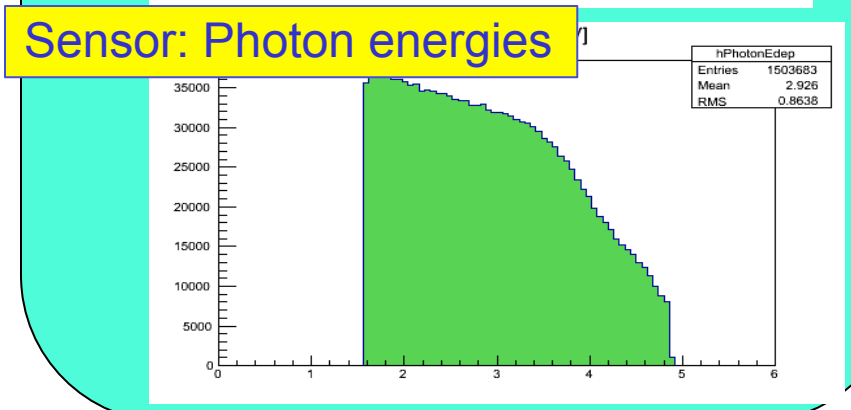


Photo Sensor (PhotonDetector)

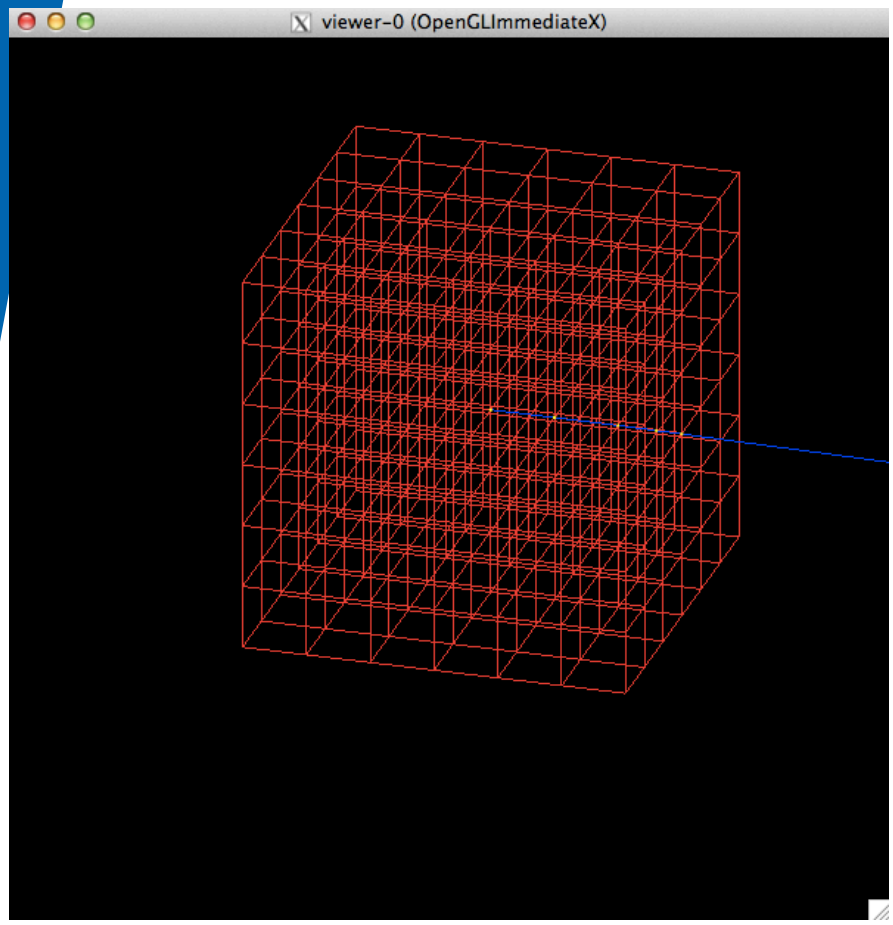


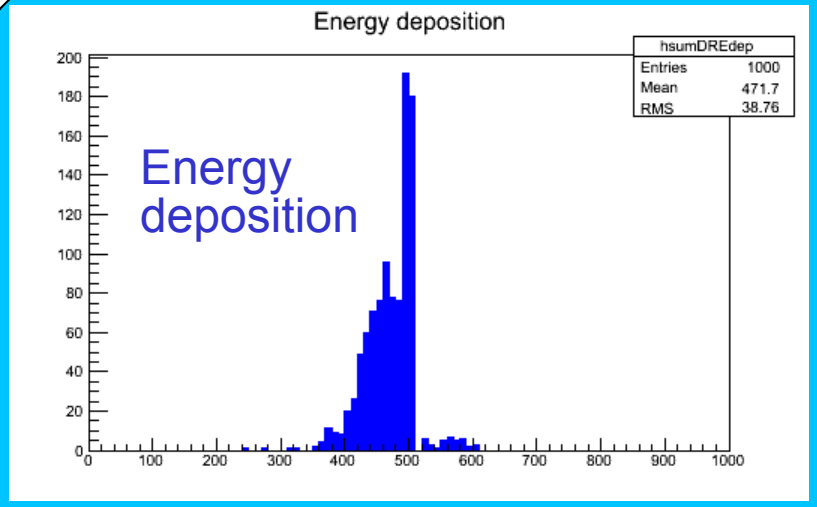


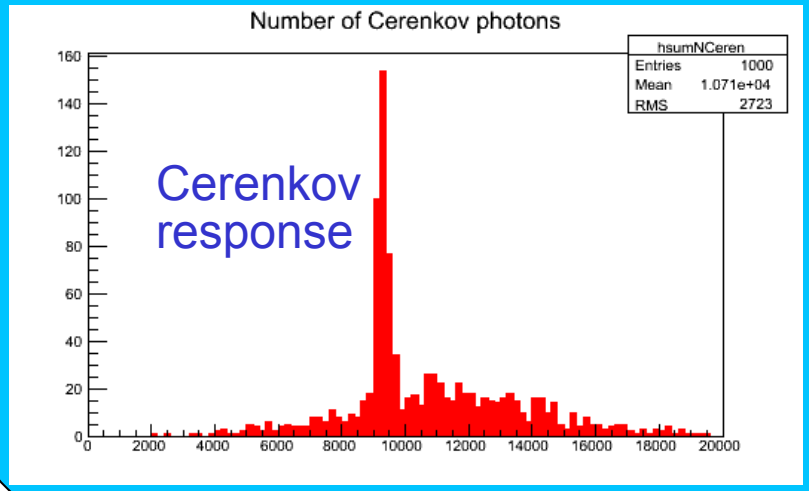


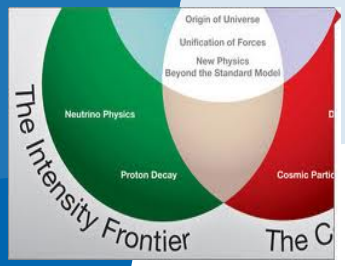
Dual Read out Calorimeter

500 MeV Proton in dual readout PbF2 crystal calorimeter





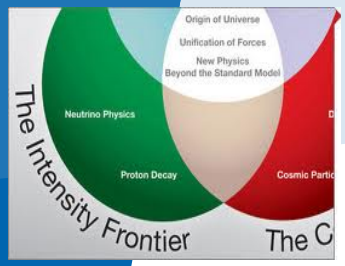




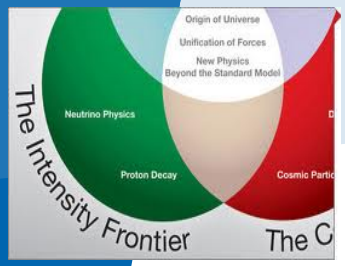
Plans

- Packaging
  - Move to latest version for products that we depend on.
  - Own product artg4tk
  - ….

- Learn how to use GPS artify simple particle Gun
- Add e.g. Birks to calorimeters
- Try out grid …
- Provide more examples.

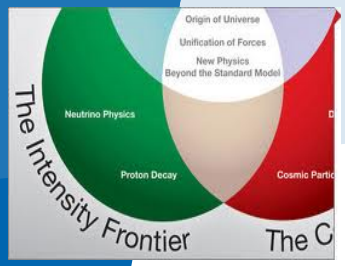

Origin of Universe
Unification of Forces
New Physics
Beyond the Standard Model
Neutrino Physics
Proton Decay
The Intensity Frontier

backup



Motivation

- Fast prototyping → make working with geant4 easy without obstructing its functionality.
- Avoid specialized application, changes to the detector don't require compilation → modular system.
- Don't reinvent the wheel (artG4, nova, LArSoft have solutions so art and Geant4 play together), CaTS provides example of modular system. GDML is a language to describe geometries….
- Make use of the components of CaTS → allow calorimeter R&D and other detector R&D projects to migrate to the new system.
- Grid: does relocate able UPS make life easier?
- Easier packaging, installation?
- Develop art expertise
- make use of software developed for art?



artg4

One producer that handles Geant: ArtG4Main

To make it generic, ArtG4Main delegates lots of responsibiliti
SERVICES that are ONLY used by ArtG4Main.
The configuration files says what Services to load

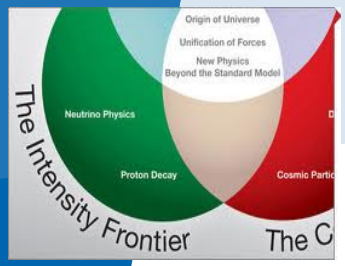
ArtG4Main Producer

GDML_Detector_Service (creates the World and SD's)

MyEventAction

SteppingAction

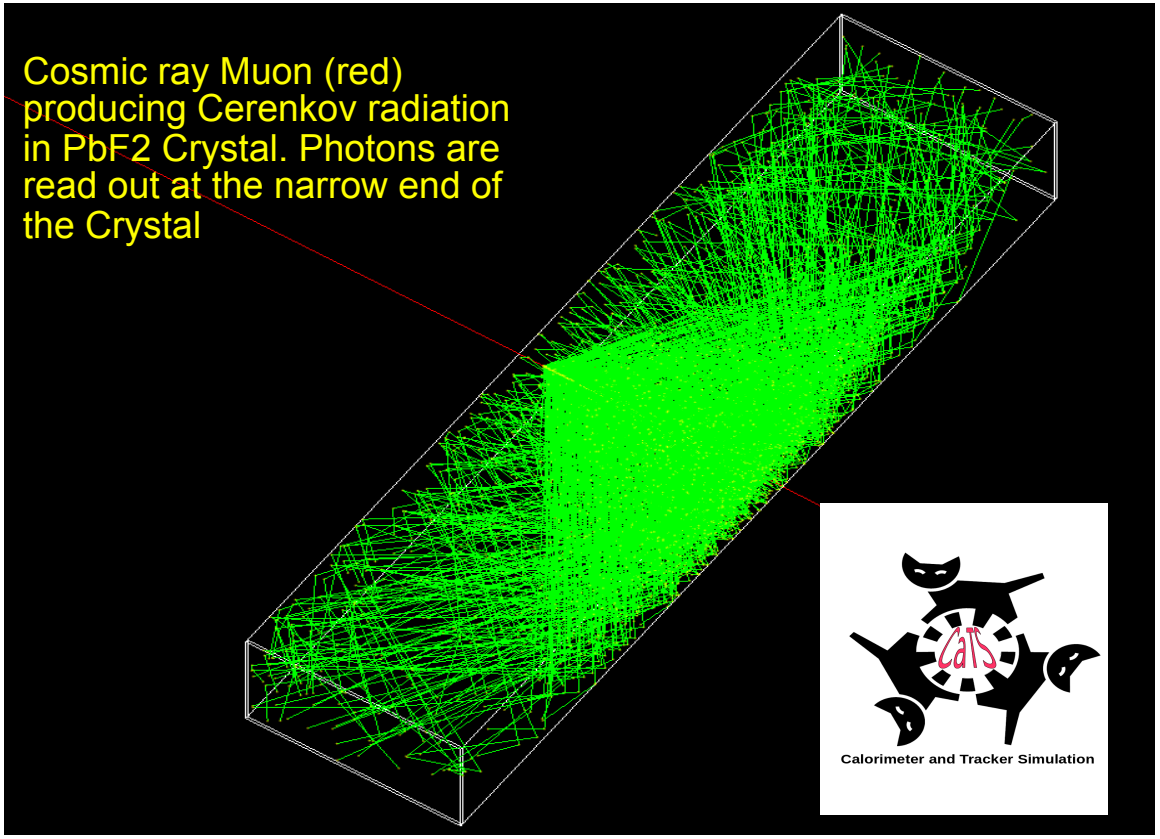
MyPhysicsList

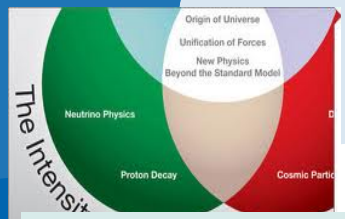


GDML

- Gdml developed to completely describe geometries, matched to the corresponding geant 4 C++ classes.
- Supported by geant 4 collaboration.
- Easily extendable to include e.g. sensitive detectors (done), visualization attributes (available), segmentation (not done).
- Several browsers exist that allow to visualize (debug) the Geometry.
- Converters available to change into different formats.
- clear separation of detector description (gdml) and run time configuration
- Sensitive Detector:
    - Knows how to create Hits and how to add hitlist to the event.
    - Is attached to a logical volume (as specified in gdml file)
- To retain provenance gdml file is stored in run record.
- But: no scheme for segmentation/readout geometry (needed?)

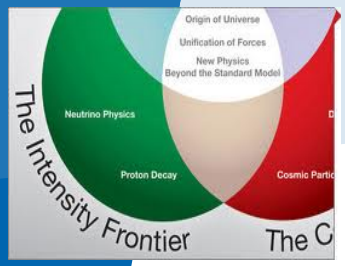
Example PbF2 Crystal





Components

Detector Description:	GDML_Detector_service used to build G4 geometry and SD. Xml based gdml input file, with extensions for SD's and visual attributes (e.g. crystalcal.gdml) (Geometry, Materials, optical properties, sensitive detector), we provide working examples, no recompilation necessary. Added mechanism to add gdml file to the run record to retain provenance.
Persistency	Provided by art, uses Root reflexion (gccxml) to automatically, create dictionaries for all classes we want to write out (e.g. Hits)
Input modules:	GPS, Particle Gun, HEPMC (Pythia)
Physics Lists:	choice of all Reference Physics Lists which can be extended to include optical physics processes (Cerenkov, Rayleigh, Scintillation etc.) → make more modular, add numi-beamline (Julia)
Sensitive Detectors and Hits:	TrackerSD, CalorimeterSD, DRCalorimeterSD (also registers Cerenkov photons), DRTSCalorimeterSD (DR+time slices), StoppingCalorimeterSD, PhotonSD: sensitive detector that registers optical photons.
User Actions:	Art Service: examples of user actions (EventAction, RunAction, StackingAction,SteppingAction...) will be provided
Code repository, Redmine project & Instructions:	Use artg4/artg4examples git repository/redmine project.
Histogram manager	Art analyzer, examples provided for various SD.



Acknowledgement

Thanks to:

Paul, Marc, Chris, Rob, Krzysztof, Daniel, Adam

Calorimeter and Tracker Simulation

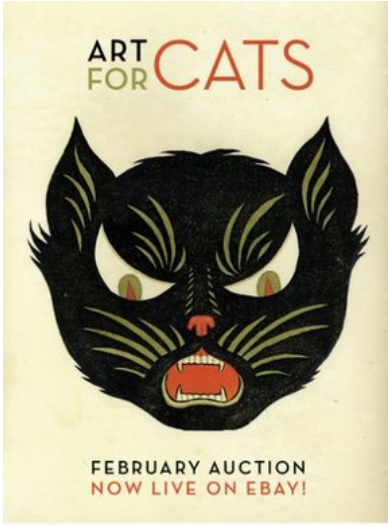
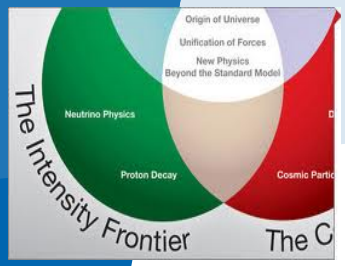
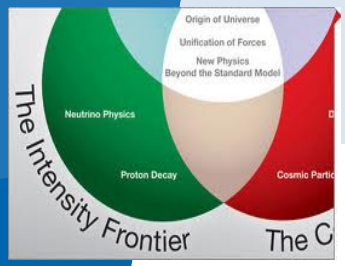


Image © Adam McCauley



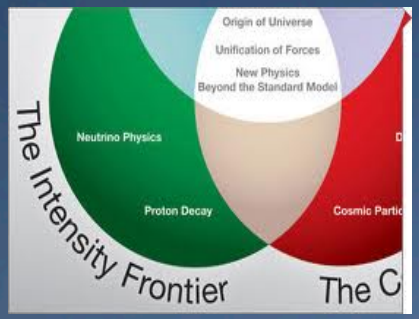
Technical

- Framework: Art → worked with it in the past
- Code Repository: Redmine and Git since this is used by ART and artg4 → requested to create the project (CaTS).
  artg4:
  https://cdcvs.fnal.gov/redmine/projects/artg4
- Detector description: options gdml e.g. used by nova/CaTS extension of Geant4., fhicl: used by artg4, custom: used by mu2e
- release management: relocatable ups???
- build system: cmake (used by Geant 4, CaTS….)
- environment setup: custom shell script
- development machine: (something with art and artg4 installed) gm2gpvm → got an account still waiting for instructions to set up the environment.
- Execution: for now use Geant 4 VO and (limited) grid resources to execute jobs



Technical (cont.)

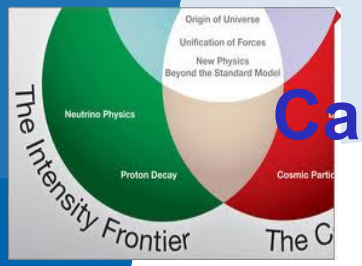
- Display of results: use geant 4 web application and database hosted here at fermilab (just create a new category)
- Configuration of physics lists/ processes: → discuss with Robert, look how it's done in G4



ART based package to monitor physics relevant to intensity frontier physics experiments

Hans Wenzel

November 21st 2013



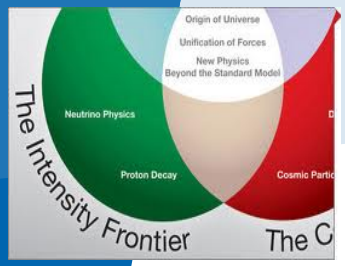
CaTS: Calorimeter and Tracker Simulatio

Calorimeter and Tracker Simulation

CaTS is a flexible and extend-able framework (based on geant4 and ROOT) for the general simulation of calorimeter and tracking detectors.

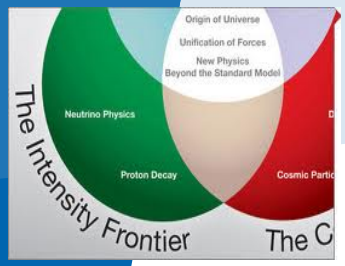
In the following look at CaTS to:

- **identify the features that we want,**
- **features that should be replaced by services of Art,**
- **extensions ??**



Charge

- Extension of geant 4 validation effort but geared specifically towards the needs of the intensity frontier community.
- develop an ART based package for monitoring of all identified physics plots relevant to intensity frontier experiments at the model and physics list level.
- port elements of Julia's (stand alone) tests to ART and in addition integrate both EM and HAD plots into this package.
- explore the possibility to use the G4-ART interface.
- Develop tools to facilitate tests and customization of physics lists.
- The plots to monitor in this package will be associated with individual models (compared with thin target experiments aka first interaction) or physics lists, as well as with quantities to be validated with results from test beam and real experiments.
- For more complex validation, simplified geometry may be used or real configurations from the experiments could be imported to this ART application.



What does art do?

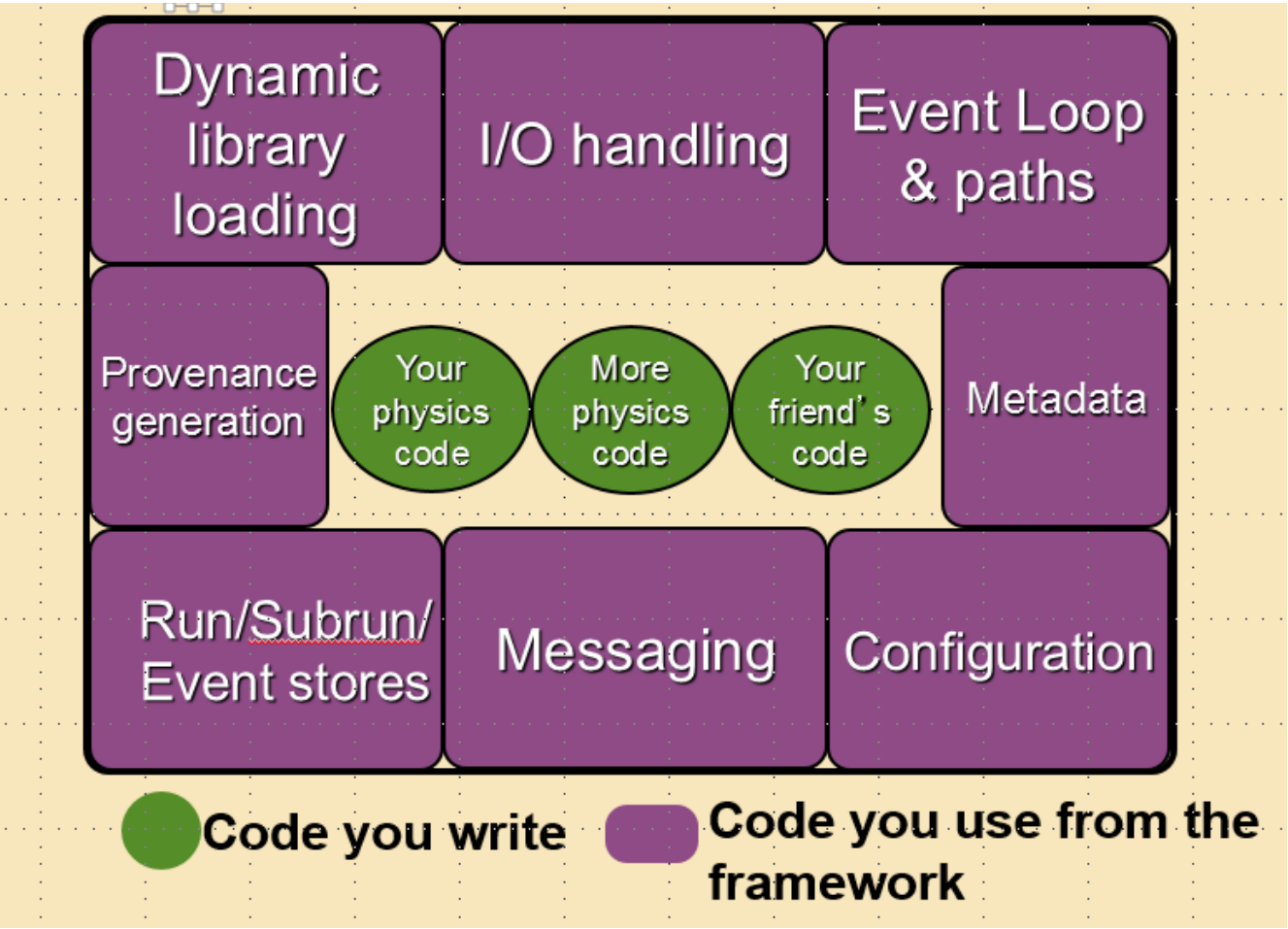
A "lite" forked version of the CMS framework

Supplies all expected framework services as well as links between data objects (Ptr's and Assn's)

Used by many Fermilab Intensity Frontier Experiments: (NOvA, g-2, Mu2e, MicroBoone,LBNE) and some others (e.g. DS50)

Written by SCD/CET department

Currently being adapted for multi-processing and DAQ



What do you write?

You write modules that can access data and do things at certain times

Types of MODULES:

(All modules can read data from the event)

o Input source:

A source for data. E.g. a ROOT file or Empty for start of simulated data

o Producers:

Create new event data from scratch or by running algorithms on existing data

o Filters:

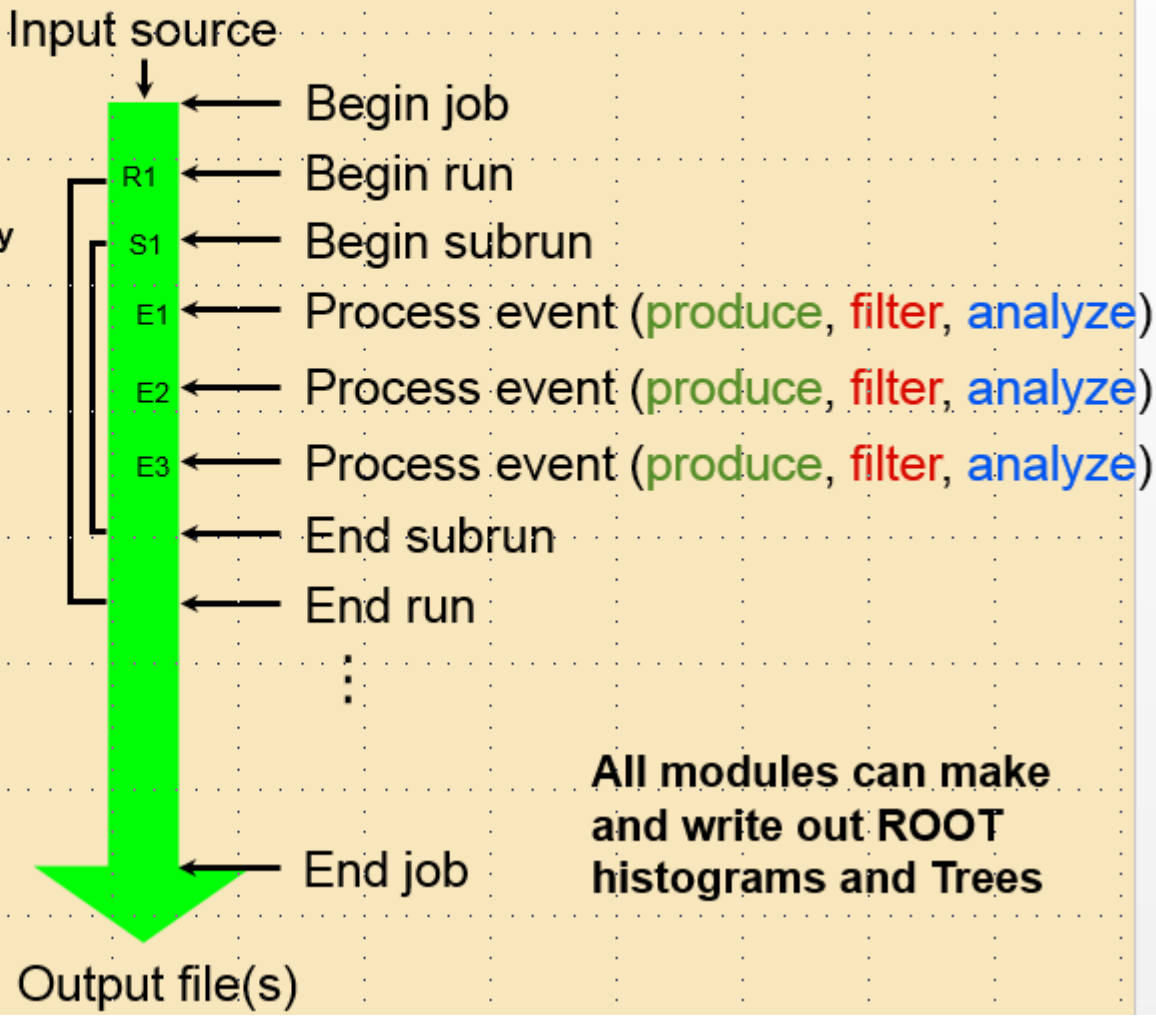
Like producers, but can stop running of downstream modules

o Analyzers:

Cannot save to event. For, e.g. diagnostics plots

o Output module:

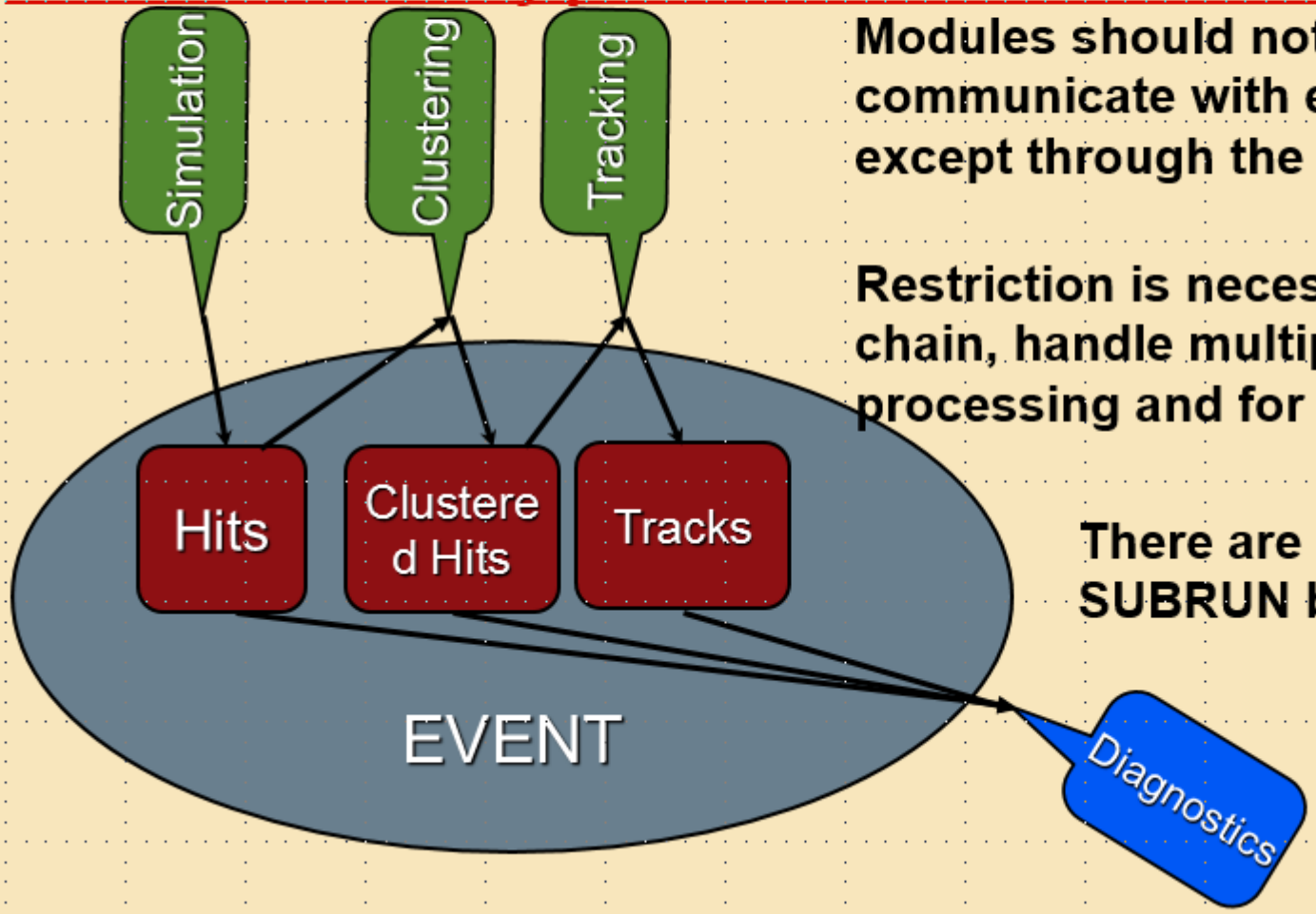
Writes data to output file (ROOT). Can specify conditions and have many files



All modules can make and write out ROOT histograms and Trees

Chain modules - but an important golden gule

Modules must only pass data to each other via the EVENT

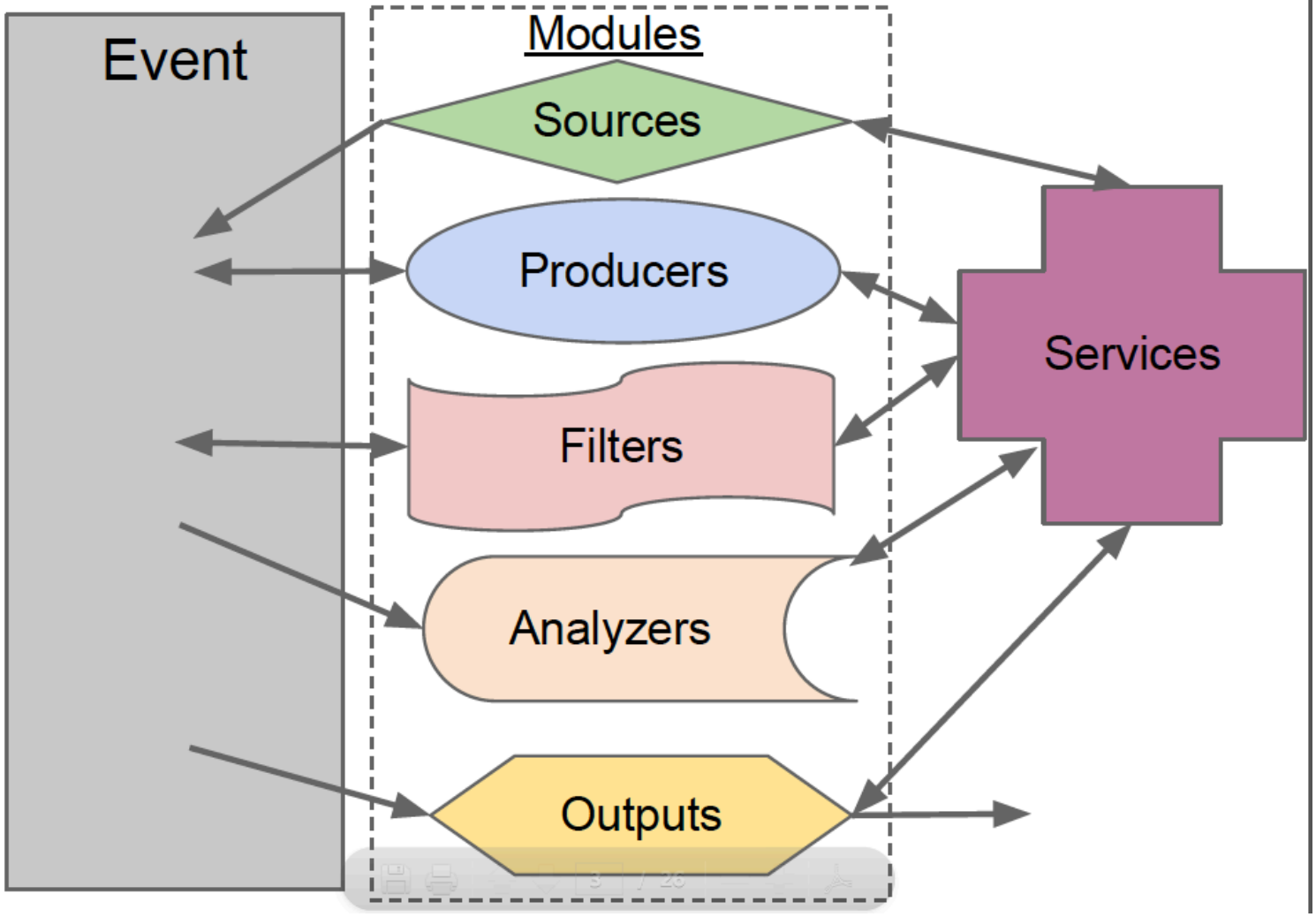


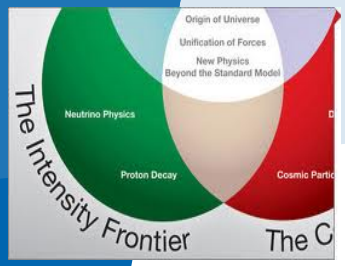
Modules should not communicate with each other, except through the event.

Restriction is necessary to break chain, handle multiprocessor processing and for sanity.

There are RUN and SUBRUN buckets too

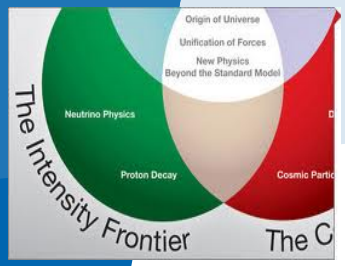
Art Glossary





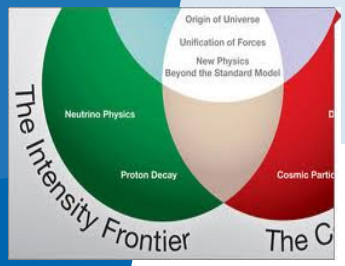
Gdml file (1: schema and definitions)

```
 ?xml version="1.0" encoding="UTF-8" ?>
<gdml xmlns:xsi="http://www.w3.org/2001 XMLSchema-instance"
xsi:noNamespaceSchemaLocation=
"http://service-spi.web.cern.ch/service-spi/app/releases/GDML/schema/gdml.xsd">

  <define>
    <variable name="i" value="0"/>
    <variable name="j" value="0"/>
    <variable name="k" value="0"/>
    <constant name="numlay" value="2"/>
    <constant name="numcol" value="2"/>
    <constant name="numrow" value="2"/>
    <constant name="scalex" value="300"/>
    <constant name="scaley" value="300"/>
    <constant name="scalez" value="5"/>
    <constant name="absoffsetz" value="-0.5"/>
    <constant name="szoffsetz" value="2.0"/>
  </define>
```



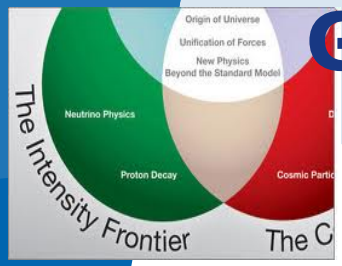
gdml file (2: materials)

```
<materials>
  <element name="Oxygen"   formula="O"  Z="8."> <atom value="16.0"/>  </element>
  <element name="Nitrogen" formula="N"  Z="7."> <atom value="14.01"/> </element>
  <element name="Lead"     formula="Pb" Z="82."> <atom value="207.20"/> </element>
  <element name="Carbon"   formula="C"  Z="6."> <atom value="12.01" unit="g/mole"/> </element>
  <element name="Hydrogen" formula="H"  Z="1."> <atom value="1.01" unit="g/mole"/> </element>

  <material name="Air">
    <D value="1.290" unit="mg/cm3"/>
    <fraction n="0.7" ref="Nitrogen"/>
    <fraction n="0.3" ref="Oxygen"/>
  </material>

  <material name="Scintillator">
    <D value="1.032" unit="g/cm3"/>
    <composite n="9" ref="Carbon"/>
    <composite n="10" ref="Hydrogen"/>
  </material>

  <material name="metalPb">
    <D value="11.340" unit="g/cm3"/>
    <composite n="1" ref="Lead"/>
  </material>
</materials>
```



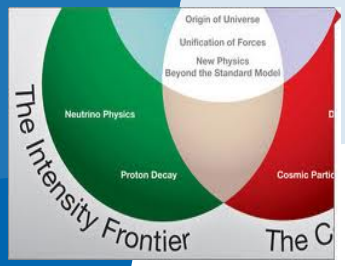
Gdml(3: solids, logical volumes)

```
<solids>
    <box name="WorldBox" lunit="mm" x="5000" y="5000" z="5000"/>
    <box name="CalorimeterCell"  lunit="mm" x="300" y="300" z="4"/>
    <box name="ScintillatorCell"  lunit="mm" x="300" y="300" z="1"/>
  </solids>
```



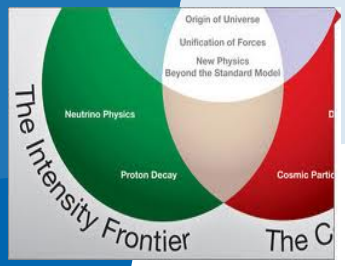
Gdml (4: placing the physical Volumes)

```
<structure>
  <volume name="CaloVol">
    <materialref ref="metalPb"/>
    <solidref ref="CalorimeterCell"/>
    <auxiliary auxtype="SensDet" auxvalue="Calorimeter"/>

  </volume>
  <volume name="ScintVol">
    <materialref ref="Scintillator"/>
    <solidref ref="ScintillatorCell"/>
    <auxiliary auxtype="SensDet" auxvalue="Calorimeter"/>
  </volume>
  <volume name="TOP">
    <materialref ref="Air"/>
    <solidref ref="WorldBox"/>
    <loop for="i" from="0" to="numrow" step="1">
    <loop for="j" from="0" to="numcol" step="1">
    <loop for="k" from="0" to="numlay" step="1">
      <physvol>
        <volumeref ref="CaloVol"/>
        <position name="posijk" x="scalex*(i-numrow/2)" y="scaley*(j-numcol/2)" z="absoffsetz+scalez*(k-numlay/2)"/>
      </physvol>
      <physvol>
        <volumeref ref="ScintVol"/>
        <position name="posijk2" x="scalex*(i-numrow/2)" y="scaley*(j-numcol/2)" z="szoffsetz+scalez*(k-numlay/2)"/>
      </physvol>
    </loop>
    </loop>
    </loop>
  </volume>
</structure>
```

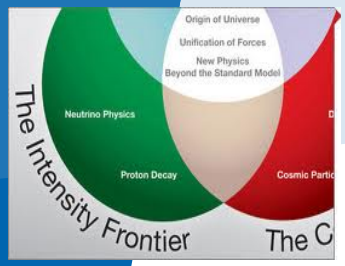


Gdml (5: define the world)

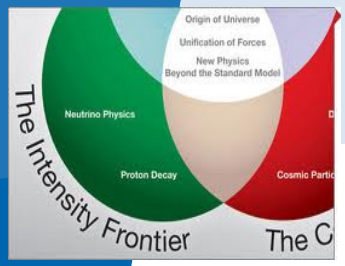
```
  <setup version="1.0" name="Default">
    <world ref="TOP"/>
  </setup>

</gdml>
```



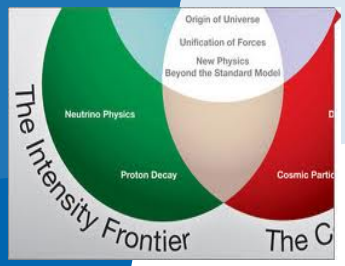

```
process_name:processA
source: {
  module_type: EmptyEvent
  maxEvents:  100
}
services: {
  message : {
     debugModules : ["*"]
     suppressInfo : []
     destinations : {
       LogToConsole : {
         type : "cout"
         threshold : "DEBUG"
         categories : {
           default : { limit : 50 }
         }
       }
     }
  }
```



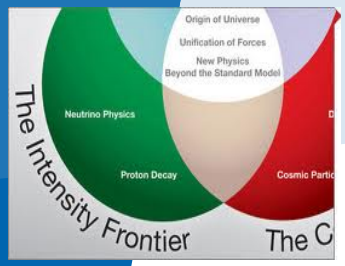

```
TFileService :
 {
   fileName      : "CheckHits.root"
 }
 user: {
   DetectorHolder: {}
   ActionHolder: {}
   RandomNumberGenerator: {}
   PhysicsListHolder: {}
   PhysicsList: {
       PhysicsListName: "FTFP_BERT"
       DumpList: false
       enableCerenkov: false
       enableScintillation: false
       enableAbsorption: false
       enableRayleigh: false
       enableMieHG: false
       enableBoundary: false
       enableWLS: false
}
```



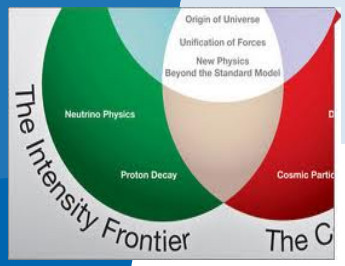

```
    // Detector(s) for the simulation
    GDMLDetector :
    {
    category: "world"
    gdmlFileName_ : "tiledsamplingcal4mmpb1mmSz.gdml"
    }

// Action(s) for the simulation
    ClockAction: {}

    ExampleGeneralAction: {
      name: "exampleGeneral"
    }
    ExampleParticleGunAction: {
      name: "exampleParticleGun"
      use_HEP_event: true
    }
  }
}// end of services!!!
```



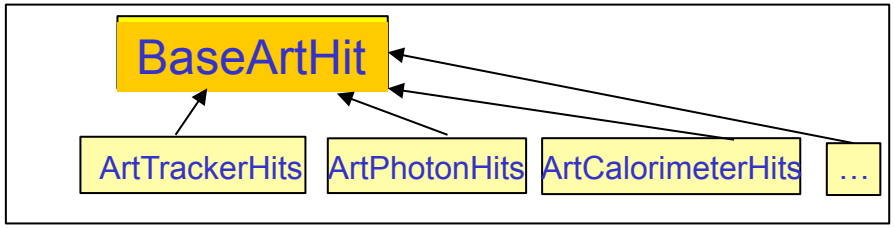

```
outputs: {
out1: {
  module_type: RootOutput
  fileName: "exampleTestingout.root"
  }
}

physics: {
  producers: {
    artg4Main: {
      module_type: artg4Main
      enableVisualization: false
      macroPath: ".:./macros"
      visMacro: "vis.mac"
      //afterEvent: pause
    }

  }
  analyzers: {
   CheckHits: {   module_type: CheckHits
                  hist_dir: "HistoDir"}
}
```




```
  path1: [ artg4Main ]
  stream1:  [ out1, CheckHits ]

  trigger_paths: [ path1 ]
  end_paths: [ stream1 ]
}
```



How to store retrieve the data



Collect all Hit collection in a map:

```
std::map< std::string, vector<BaseArtHit*> >
```

String encodes:
- Logical Volume that the SD is attached so
- Class name of specific Hit class

```
ArtCalorimeterHit* DRHit = dynamic_cast<ArtCalorimeterHit*> (hits[ii]);
```