E1039 offline software introduction

Haiwang Yu (NMSU)

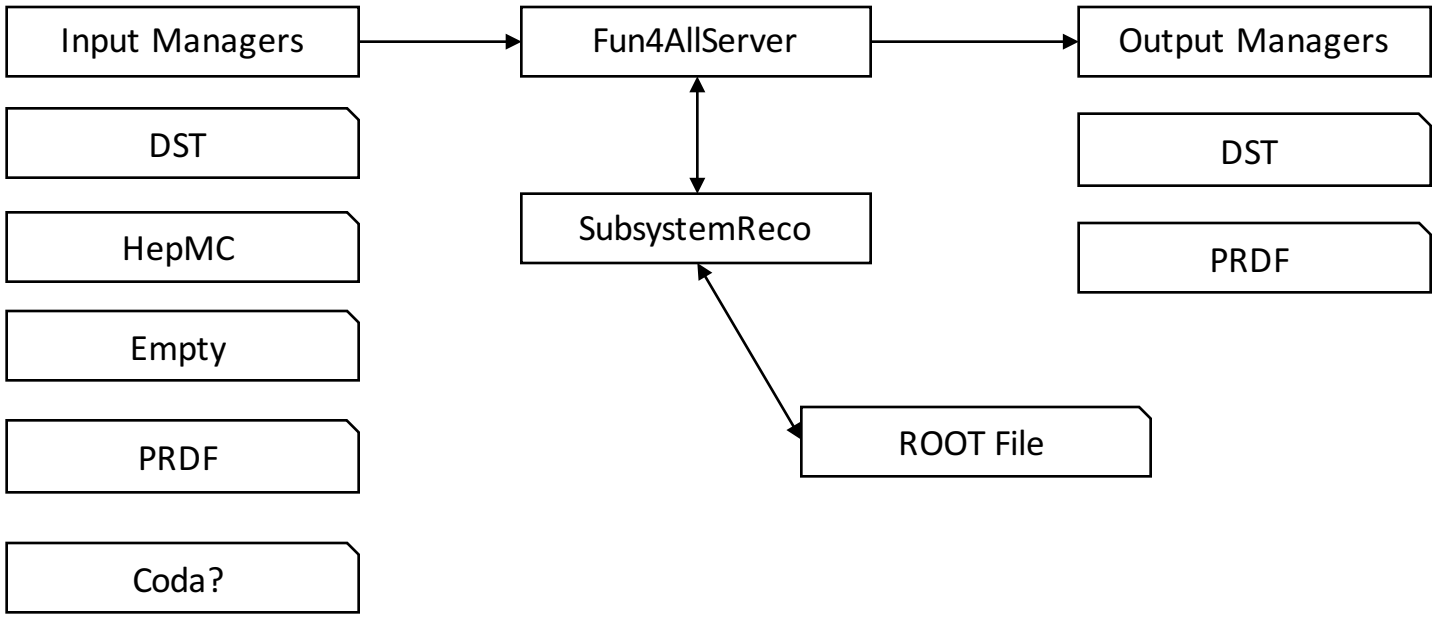
Fun4All framework by C. Pinkenburg

Used in PHENIX/sPHENIX experiment. Developed by C. Pinkenburg from BNL. This tutorial also contents several slides from his talk.

There is ONLY ONE executable: root.exe
ROOT becomes useful by two methods:

- Compile system/user code into *.so
  - rootcint makes dictionaries of user objects so that these can be instantiated and manipulated at the root prompt.
- Execute macro at the root prompt.

User *.so attaches to core *.so via libadd in their package's Makefile.am file.



The Phool Node Tree

- The Node Tree is at the center of the PHENIX software universe (but it's more or less invisible to you). It's the way we organize our data.
- **It is NOT a ROOT TTree**
- We have 3 different Types of Nodes:
  - PHCompositeNode: contains other Nodes
  - PHDataNode: contains any object
  - PHIODataNode: contains objects which can be written out to DST
- PHCompositeNodes and PHIODataNodes can be saved to a DST and read back
- This DST contains root TTrees, the node structure is saved in the branch names. Due to ROOT's limitations not all objects can become PHIODataNodes (e.g. anything containing BOOST).
- We currently save 2 root trees in each output file, one which contains the eventwise information, one which contains the runwise information
- Input Managers put objects as PHIODataNodes on the node tree, output managers save selected PHIODataNodes to a file.
- Fun4All can manage multiple independent node trees

**ALL items in PHIODataNode move seamlessly to Database and File storage.
AUTOMATIC FORMATING

Read and create node

```
#include <g4hit/PHG4HitContainer.h>
#include <fun4all/getClass.h>

Myanalysis::InitRun(PHCompositeNode *topNode) {
   PHNodeIterator iter(topNode);
   PHCompositeNode* eventNode = static_cast<PHCompositeNode*>(iter.findFirst("PHCompositeNode", "DST"));
   if(!eventNode) {/*add some protection*/}

   PHG4HitContainer *g4hits = new PHG4HitContainer();
   PHIODataNode<PHObject>* g4hits_node = new PHIODataNode<PHObject>(g4hits, " G4HIT_HCALIN ", "PHObject");

   eventNode->addNode(g4hits_node);
}

Myanalysis::process_event(PHCompositeNode *topNode){
   PHG4HitContainer *g4hits = findNode::getClass<PHG4HitContainer>(topNode,"G4HIT_HCALIN");
   if(g4hits) {/*analysis code*/}
}
```

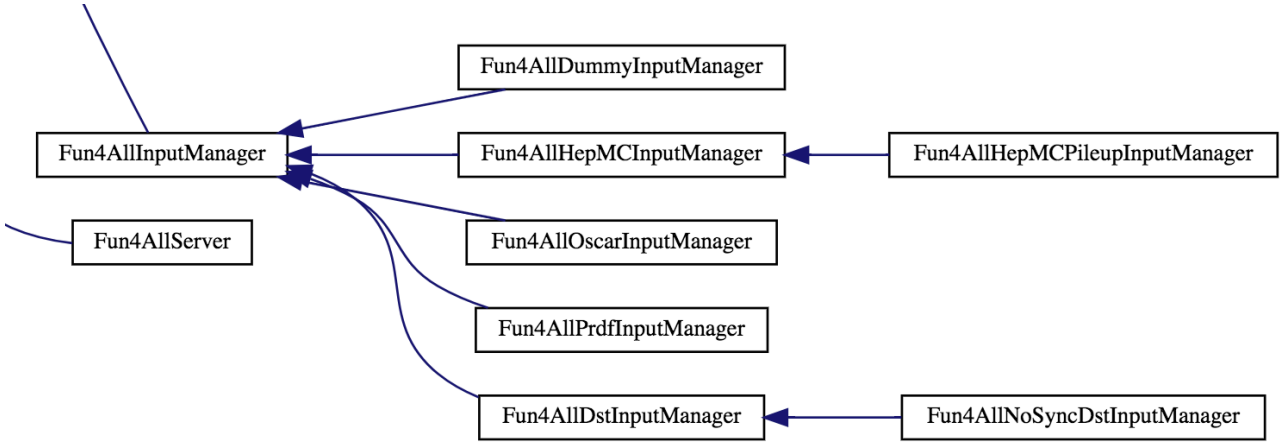
Node Tree use in the Hodoscope study

```
TOP (PHCompositeNode)/
   DST (PHCompositeNode)/
      PHHepMCGenEventMap (IO,PHHepMCGenEventMap)
      PHG4INEVENT (PHDataNode)
      G4HIT_C1X (IO,PHG4HitContainer)
      G4HIT_C1V (IO,PHG4HitContainer)
      G4HIT_C1U (IO,PHG4HitContainer)
      G4HIT_C2U (IO,PHG4HitContainer)
      G4HIT_C2X (IO,PHG4HitContainer)
      G4HIT_C2V (IO,PHG4HitContainer)
      G4HIT_C3T (IO,PHG4HitContainer)
      G4HIT_C3B (IO,PHG4HitContainer)
      G4HIT_osta4 (IO,PHG4HitContainer)
      G4HIT_osta3 (IO,PHG4HitContainer)
      G4HIT_osta2 (IO,PHG4HitContainer)
      G4HIT_osta1 (IO,PHG4HitContainer)
      G4HIT_H1y (IO,PHG4HitContainer)
      G4HIT_H1x (IO,PHG4HitContainer)
      G4HIT_H2y (IO,PHG4HitContainer)
      G4HIT_H2x (IO,PHG4HitContainer)
      G4HIT_H3x (IO,PHG4HitContainer)
      G4HIT_P1V (IO,PHG4HitContainer)
      G4HIT_P2H (IO,PHG4HitContainer)
      G4HIT_P2V (IO,PHG4HitContainer)
      G4HIT_P1H (IO,PHG4HitContainer)
      G4HIT_H4y1L (IO,PHG4HitContainer)
      G4HIT_H4y1R (IO,PHG4HitContainer)
      G4HIT_H4y2L (IO,PHG4HitContainer)
      G4HIT_H4y2R (IO,PHG4HitContainer)
      G4HIT_H4xT (IO,PHG4HitContainer)
      G4HIT_H4xB (IO,PHG4HitContainer)
      G4TruthInfo (IO,PHG4TruthInfoContainer)
      SQHitVector (IO,SQHitVector_v1)
      SRecEvent (IO,SRecEvent)
```

```
RUN (PHCompositeNode)/
   PHGenIntegral (IO,PHGenIntegralv1)
   FIELD_CONFIG (IO,PHFieldConfig_v3)
   G4GEOPARAM_Insens_0 (IO,PdbParameterMapContainer)
   C1X (PHCompositeNode)/
      G4GEOPARAM_C1X (IO,PdbParameterMapContainer)
   BLOCKGEOM_C1X (IO,PHG4BlockGeomContainer)
   C1V (PHCompositeNode)/
      G4GEOPARAM_C1V (IO,PdbParameterMapContainer)
   BLOCKGEOM_C1V (IO,PHG4BlockGeomContainer)
```

```
PAR (PHCompositeNode)/
   FIELD_MAP (PHDataNode)
   G4GEO_Insens_0 (PHDataNode)
   C1X (PHCompositeNode)/
      G4GEO_C1X (PHDataNode)
   C1V (PHCompositeNode)/
      G4GEO_C1V (PHDataNode)
   C1U (PHCompositeNode)/
      G4GEO_C1U (PHDataNode)
   C2U (PHCompositeNode)/
      G4GEO_C2U (PHDataNode)
```

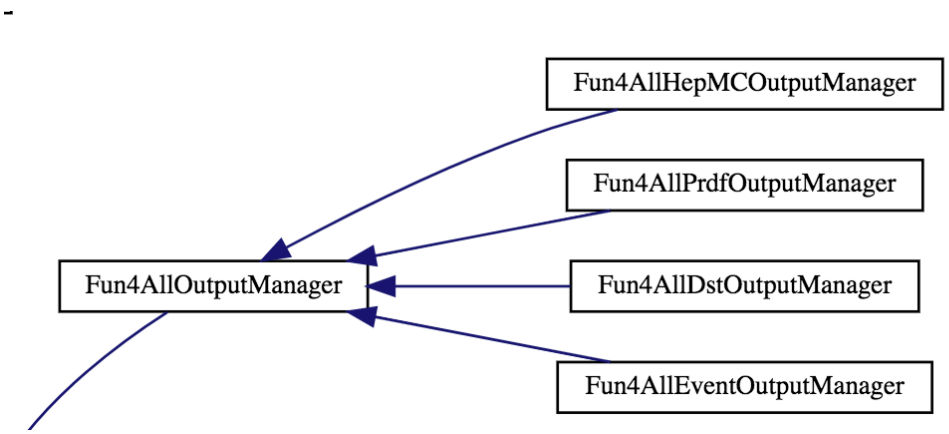
Input manager



- Can handle multiple input streams with optional syncing
- Support embedding

```
19   int fileopen(const std::string &filenam);
20   int fileclose();
21   int run(const int nevents = 0);
22   int isOpen() {return isopen;}
23
24   void Print(const std::string &what = "ALL") const;
25   int ResetEvent();
26   int PushBackEvents(const int i);
27   int GetSyncObject(SyncObject **mastersync);
28   int SyncIt(const SyncObject *mastersync);
```

Output manager




```
19    int AddNode(const std::string  &nodename);
20    int StripNode(const std::string  &nodename);
21    int outfileopen(const std::string &fname);
22    int RemoveNode(const std::string &nodename);
23
24    void Print(const std::string &what = "ALL") const;
25
26    int Write(PHCompositeNode *startNode);
27    int WriteNode(PHCompositeNode *thisNode);
```

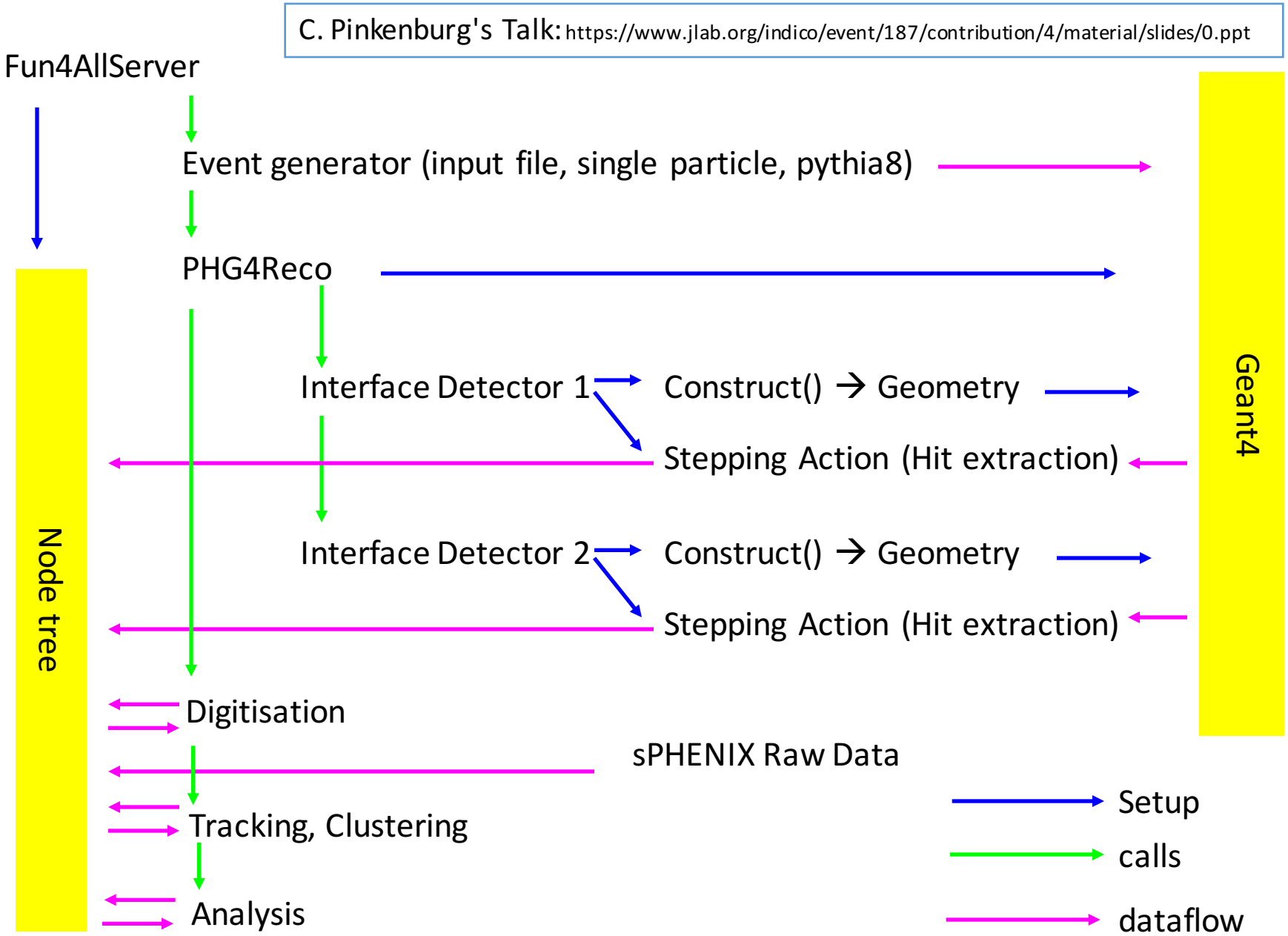
SubsystemReco - the worker module

- Init(PHCompositeNode *topNode): called once when you register the module with the Fun4AllServer
- InitRun(PHCompositeNode *topNode): called whenever data from a new run is encountered
- process_event (PHCompositeNode *topNode): called for every event
- ResetEvent(PHCompositeNode *topNode): called after each event is processed so you can clean up leftovers of this event in your code
- EndRun(const int runnumber): called before the InitRun is called (caveat the Node tree already contains the data from the first event of the new run)
- End(PHCompositeNode *topNode): Last call before we quit

- Example: seaquest-offline/module_examples/TrkEval

```
51   int Init(PHCompositeNode *topNode);
52   int InitRun(PHCompositeNode *topNode);
53   int process_event(PHCompositeNode *topNode);
54   int End(PHCompositeNode *topNode);
```

sPHENIX simulation framework with Fun4All



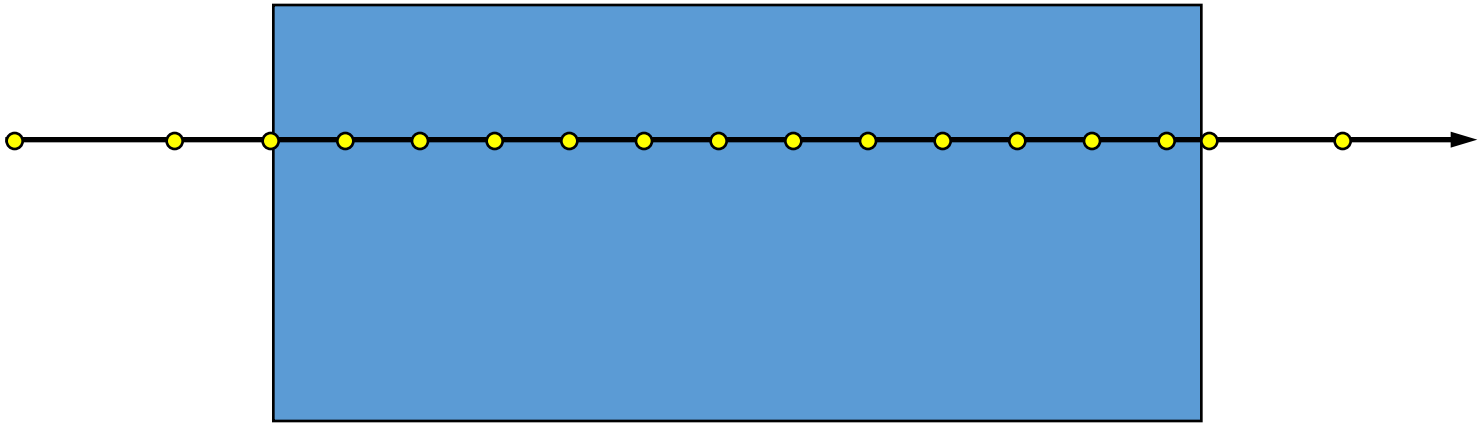
PHG4Reco

- Fun4All Interface to GEANT4
- Sets features of the world (size, shape, material, magnetic field, physics list)
- Provides interface to GEANT command line (especially useful for event display)
- Manages our detectors
- totally configurable on macro level

```
// Fun4All G4 module
PHG4Reco *g4Reco = new PHG4Reco();
//g4Reco->G4Seed(123);
//g4Reco->set_field(5.);
g4Reco->set_field_map(
    jobopt_svc->m_fMagFile+" "+
    jobopt_svc->m_kMagFile,
    4);
// size of the world - every detector has to fit in here
g4Reco->SetWorldSizeX(1000);
g4Reco->SetWorldSizeY(1000);
g4Reco->SetWorldSizeZ(5000);
// shape of our world - it is a tube
g4Reco->SetWorldShape("G4BOX");
// this is what our world is filled with
g4Reco->SetWorldMaterial("G4_AIR"); //G4_Galactic, G4_AIR
// Geant4 Physics list to use
g4Reco->SetPhysicsList("FTFP_BERT");
```

GEANT steps

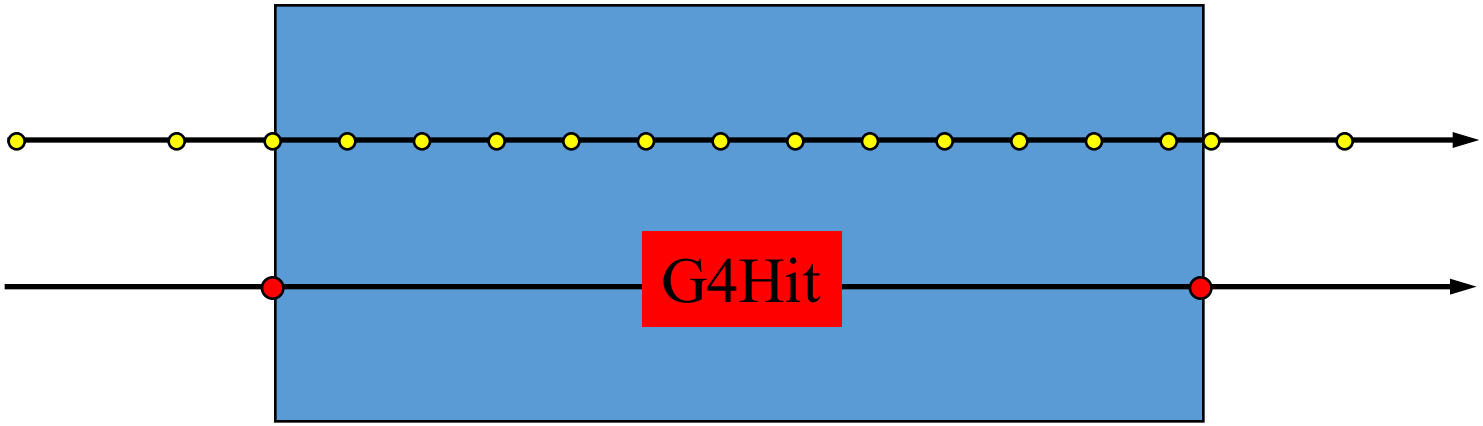
GEANT propagates particles one step at a time. The step size is determined by the physics processes associated with the current particle or when a boundary between volumes is crossed



After each step the user stepping method is called with a pointer to the current volume which has access to the full information (energy loss, particle momentum at beginning and end of step, …)

Integrated G4Hits

In our stepping method we add the energy loss in each volume and store the entry and exit coordinates (and for tracking detectors the momentum at the entry and exit)



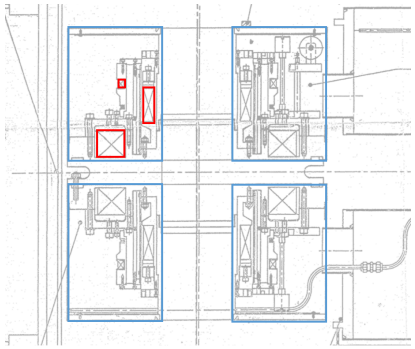
We also keep the ancestry for G4Hits so any hit can be traced back to a primary particle. To reduce size we do not store particles which do not leave G4Hits and are not in the ancestry of a particle which created a G4Hit

Detector classes

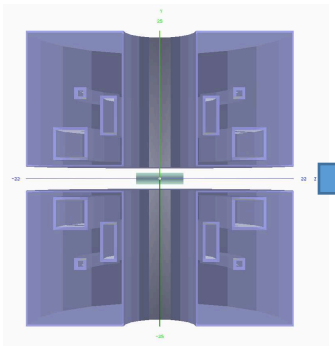
PHG4Subsystem: SubsysReco, interface to Fun4All
PHG4Detector: Detector setup – implements GEANT construct method
PHG4SteppingAction: implements stepping action for each detector
PHG4EventAction: implements actions taken at end of event (e.g. removing G4Hits with zero energy loss)

Example: **PHG4TargetCoil**

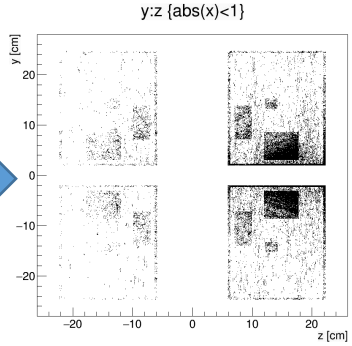
design drawing



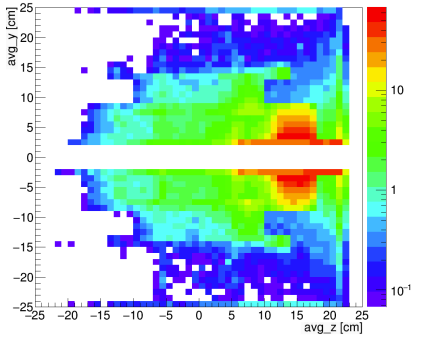
G4 model



G4 simulation
w/ sPHENIX framework



Analysis:
sum up edep of G4Hits

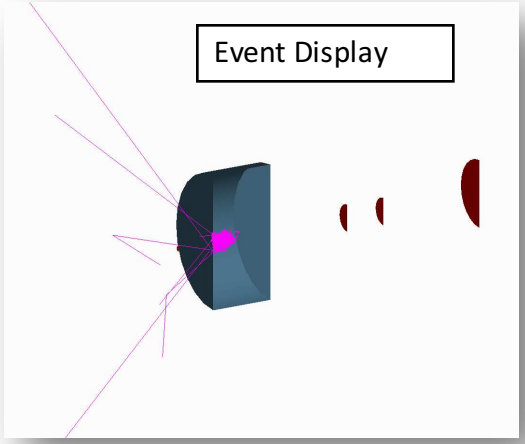


Configure simple geometry in macro

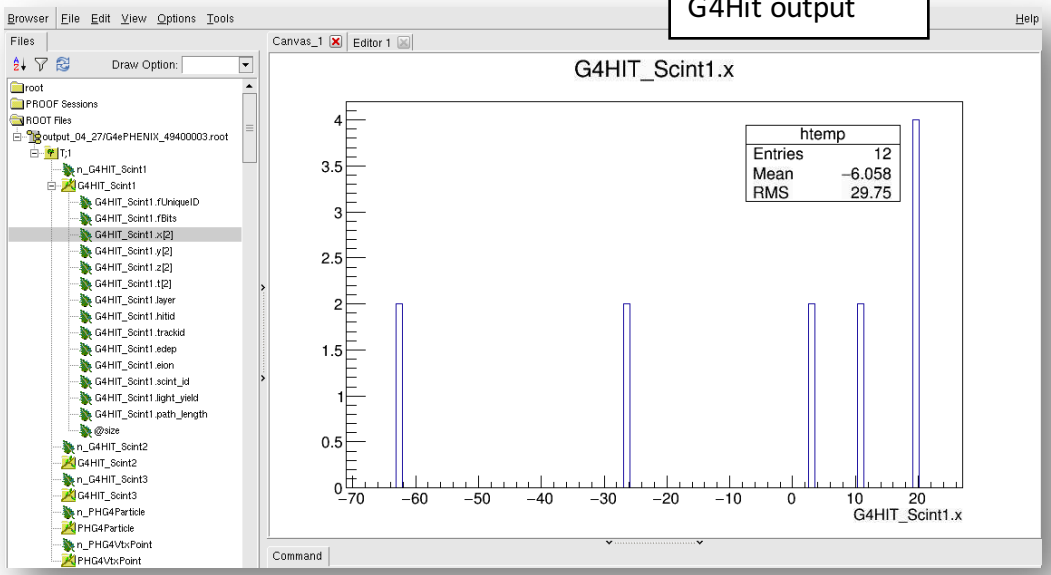
code generating one disk - PHG4DetectorSubsystem

```
cyl = new PHG4CylinderSubsystem("Scint1");
cyl->SuperDetector("Scint1");
cyl->SetRadius(0.0);
cyl->SetThickness(100.1);
cyl->SetLengthViaRapidityCoverage(false);
cyl->SetLength(1);
//cyl->SetMaterial("G4_METHANE");
cyl->SetMaterial("G4_POLYETHYLENE");
cyl->SetPosition(0, 0, target_z + 1400);
cyl->SetActive();
cyl->OverlapCheck(overlapcheck);
g4Reco->registerSubsystem(cyl);
```

Event Display



G4Hit output



Configure geometry (simple shapes) and sensitive detector at macro level

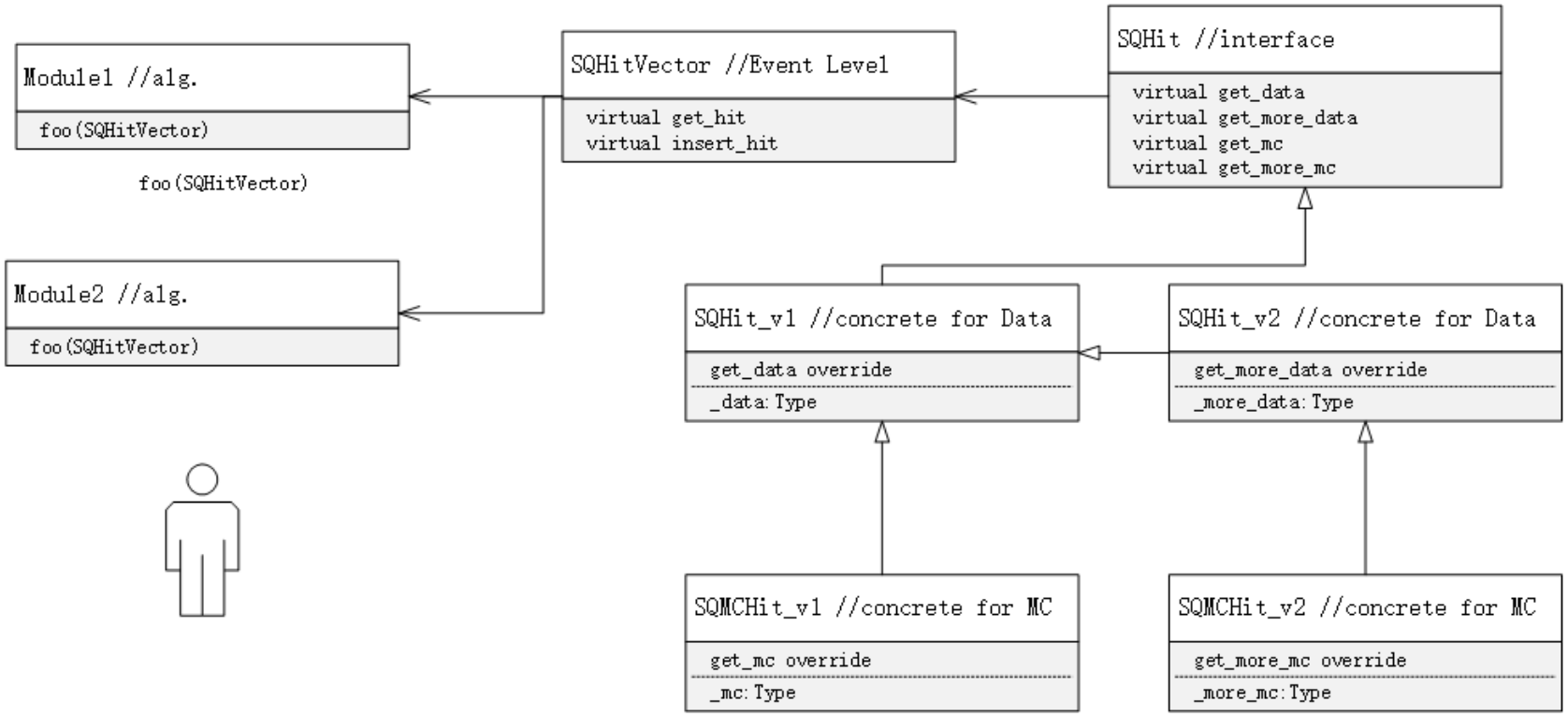
Automatic collect g4hit -> PHG4Hit

Now available: box, tube, cone
Developing: square tube

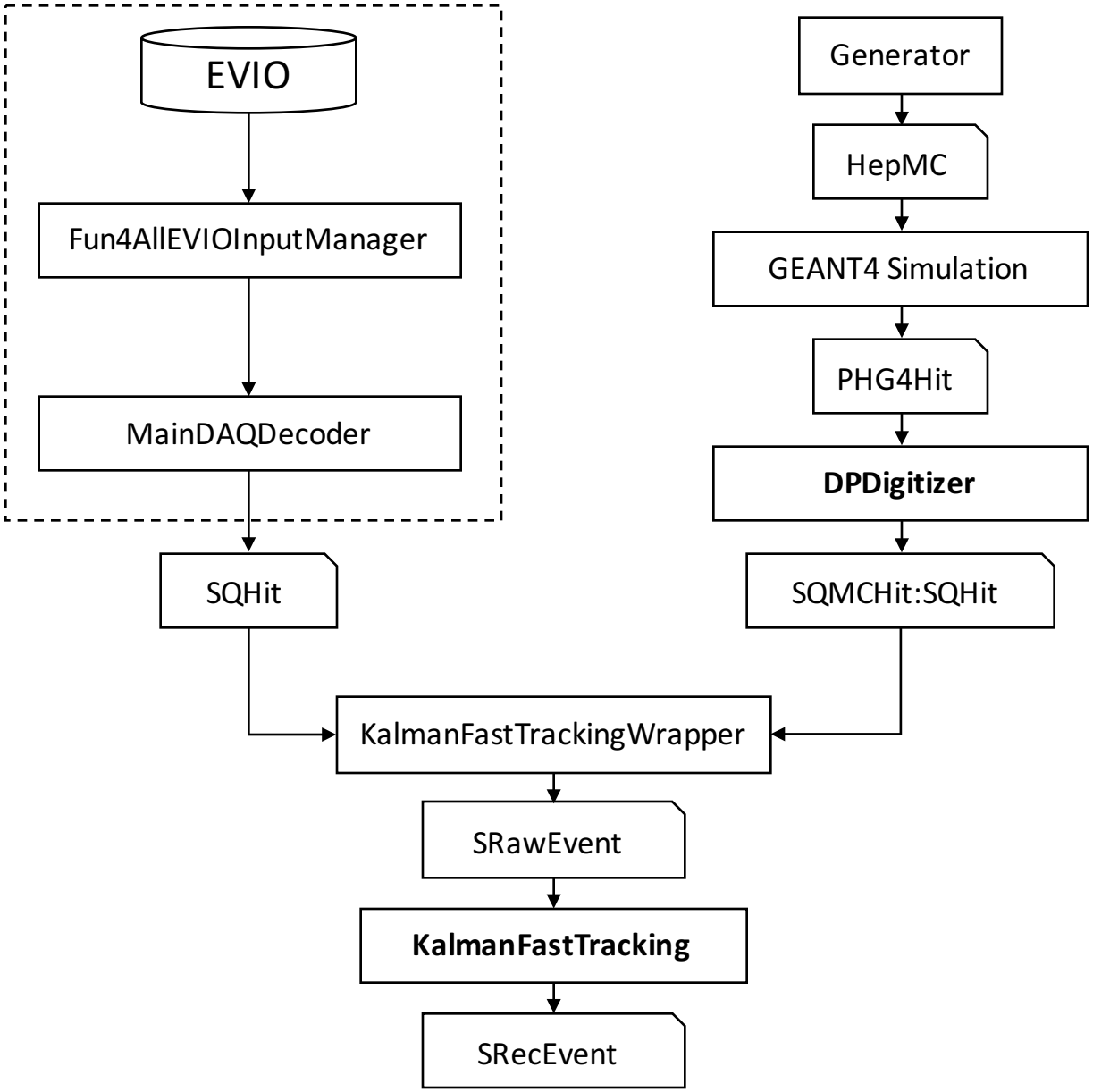
Unified interfaces for both Data and Simulation

The facade design pattern:

- unified interface
- save disk space
- backward compatibility

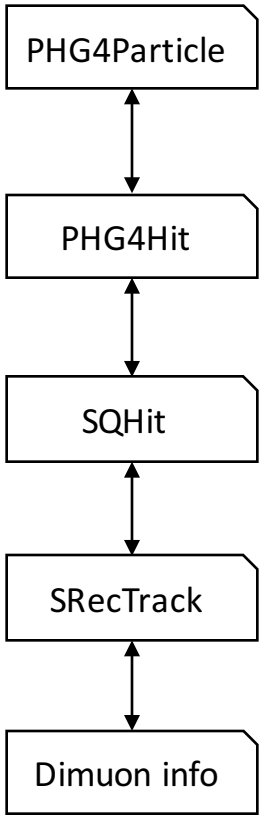


Flowchart of data and simulation reco.



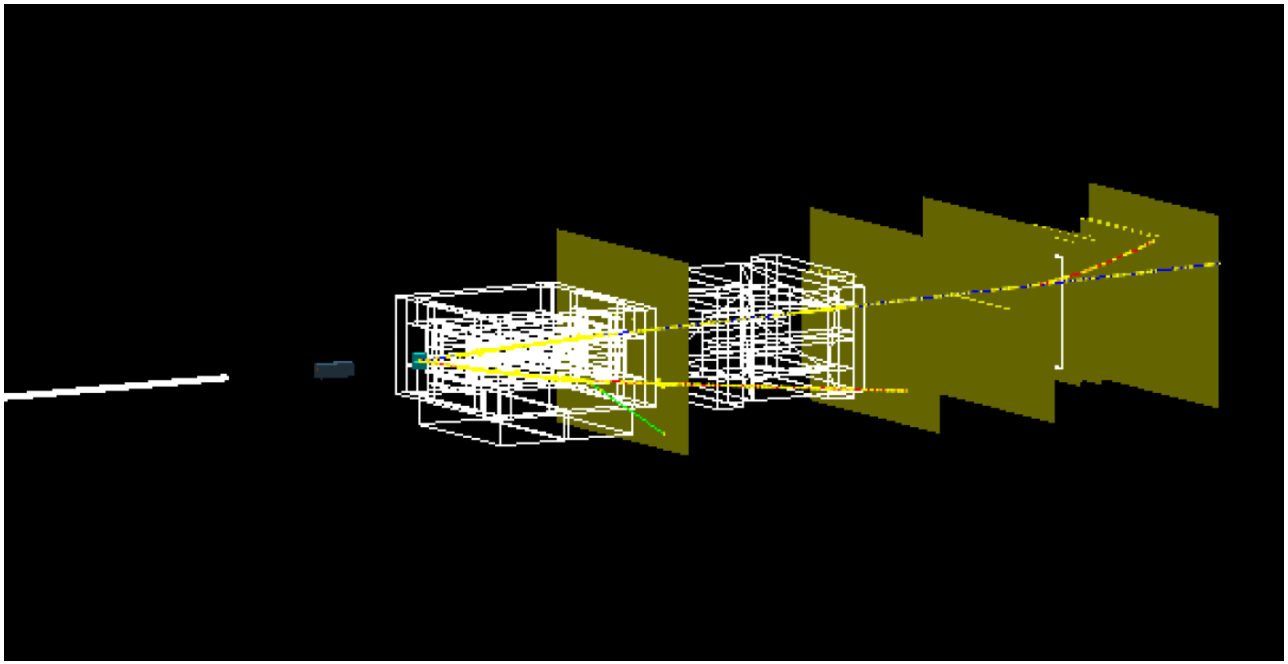
General purposed MC analysis module TrkEval

- General purposed evaluator - read in DST ouput ntuple like TTrees
- seaquest-offline/module_examples/TrkEval
- Still in developing, implemented parts:
  - Hit info - G4Hit matched
  - generated particle info
    - hit info
    - best matched (max hit) reco. track
  - generated dimu info
    - best matched reco. dimu



Example: Hodo-Acc-Gap Study

https://github.com/E1039-Collaboration/e1039-wiki/wiki/Start-up-guide-with-an-example



Backups

Phool Node Tree for sPHENIX

Print it from the cmd line with
Fun4AllServer *se = Fun4AllServer::instance();
se->Print("NODETREE");

```
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
  DST (PHCompositeNode)/
    G4HIT_HCALIN (PHIODataNode)
    G4HIT_ABSORBER_HCALIN (PHIODataNode
    G4HIT_HCALOUT (PHIODataNode)
    G4HIT_ABSORBER_HCALOUT (PHIODataNo
    SVTX (PHCompositeNode)/
      SvtxHitMap (PHIODataNode)
      SvtxClusterMap (PHIODataNode)
    SVTX_EVAL (PHCompositeNode)/
      SvtxClusterMap_G4HIT_SVTX_Links (PHIODataNode)
RUN (PHCompositeNode)/
    CYLINDERGEOM_SVTX (PHIODataNode)
    CYLINDERGEOM_SVTXSUPPORT (PHIODataNode)
    CYLINDERGEOM_EMCELECTRONICS_0 (PHIODataNode)
    CYLINDERGEOM_HCALIN_SPT (PHIODataNode)
PAR (PHCompositeNode)/
    SVTX (PHCompositeNode)/
      SvtxBeamSpot (PHIODataNode)
```

TOP: Top of Default Node Tree Creation and populating of other node trees is possible (used for embedding)

Phool Node Tree for sPHENIX

Print it from the cmd line with
Fun4AllServer *se = Fun4AllServer::instance();
se->Print("NODETREE");

```
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
  DST (PHCompositeNode)/
    G4HIT_HCALIN (PHIODataNode)
    G4HIT_ABSORBER_HCALIN (PHIODataNode)
    G4HIT_HCALOUT (PHIODataNode)
    G4HIT_ABSORBER_HCALOUT (PHIODataNode)
    SVTX (PHCompositeNode)/
      SvtxHitMap (PHIODataNode)
      SvtxClusterMap (PHIODataNode)
    SVTX_EVAL (PHCompositeNode)/
      SvtxClusterMap_G4HIT_SVTX_Links (PHIODataNo
  RUN (PHCompositeNode)/
    CYLINDERGEOM_SVTX (PHIODataNode)
    CYLINDERGEOM_SVTXSUPPORT (PHIODataNode)
    CYLINDERGEOM_EMCELECTRONICS_0 (PHIODataNode)
    CYLINDERGEOM_HCALIN_SPT (PHIODataNode)
PAR (PHCompositeNode)/
    SVTX (PHCompositeNode)/
      SvtxBeamSpot (PHIODataNode)
```

DST and RUN Node: default for I/O

- DST – eventwise
- RUN - runwise

Objects under the DST node are reset after every event to prevent event mixing. You can select the objects to be saved in the output file. Subnodes like SVTX are saved and restored as well. DST/RUN nodes can be restored from file under other TopNodes
ROOT restrictions apply:
Objects cannot be added while running to avoid event mixing

Phool Node Tree for sPHENIX

Print it from the cmd line with
Fun4AllServer *se = Fun4AllServer::instance();
se->Print("NODETREE");

```
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
  DST (PHCompositeNode)/
    G4HIT_HCALIN (PHIODataNode)
    G4HIT_ABSORBER_HCALIN (PHIODataN
    G4HIT_HCALOUT (PHIODataNode)
    G4HIT_ABSORBER_HCALOUT (PHIODat
    SVTX (PHCompositeNode)/
      SvtxHitMap (PHIODataNode)
      SvtxClusterMap (PHIODataNode)
    SVTX_EVAL (PHCompositeNode)/
      SvtxClusterMap_G4HIT_SVTX_Links (PH
RUN (PHCompositeNode)/
    CYLINDERGEOM_SVTX (PHIODataNode)
    CYLINDERGEOM_SVTXSUPPORT (PHIODataNode)
    CYLINDERGEOM_EMCELECTRONICS_0 (PHIODataNode)
    CYLINDERGEOM_HCALIN_SPT (PHIODataNode)
PAR (PHCompositeNode)/
    SVTX (PHCompositeNode)/
      SvtxBeamSpot (PHIODataNode)
```

Users can add their own branches. Resetting the objects is their responsibility.

The PAR node will probably turn into the node for calibrations which we want to keep on the DST

Phool Node Tree for sPHENIX

Print it from the cmd line with
Fun4AllServer *se = Fun4AllServer::instance();
se->Print("NODETREE");

```
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
  DST (PHCompositeNode)/
    G4HIT_HCALIN (PHIODataNode)
    G4HIT_ABSORBER_HCALIN (PHIODataNo
    G4HIT_HCALOUT (PHIODataNode)
    G4HIT_ABSORBER_HCALOUT (PHIODataN
    SVTX (PHCompositeNode)/
      SvtxHitMap (PHIODataNode)
      SvtxClusterMap (PHIODataNode)
    SVTX_EVAL (PHCompositeNode)/
      SvtxClusterMap_G4HIT_SVTX_Links (PHI
RUN (PHCompositeNode)/
    CYLINDERGEOM_SVTX (PHIODataNode)
    CYLINDERGEOM_SVTXSUPPORT (PHIODataNode)
    CYLINDERGEOM_EMCELECTRONICS_0 (PHIODataNode)
    CYLINDERGEOM_HCALIN_SPT (PHIODataNode)
PAR (PHCompositeNode)/
    SVTX (PHCompositeNode)/
      SvtxBeamSpot (PHIODataNode)
```

Caveat: You loose ownership once an object is put on the node tree. Fun4All deletes the node tree when cleaning up. Deleting nodes is not supported (yet – but ROOT prevents this for the DST node)