Lorenzett framework

https://github.com/jodafons/lorenzett

Follow the instructions in README.md

You have to obtain your GitHub and Docker Hub accounts:

https://github.com

https://www.docker.com

The Docker Hub account has to be linked to the GitHub account.
In your Docker Hub account look at Linked account, you will see:
Provider (GitHub) and Account (your name account) press the curly arrows to connect (a blue plug connect symbol will appear).

Fork the directory jodafons/lorenzett

Press the fork symbol and “clone or download” to your local area

```
> git nome https://github.com/<your_account>/lorenzett.git
```

Using a specific image (sdumont for example)

```
> docker run -v :/output gabrielmilan/lorenzett:sdumont <args>
```

```
> source setup_envs.sh
```

For **event generation**, this is the command you're looking for (example for Zee):

```
> prun_job.py -c "generator.py --filter Zee -i generator/PythiaGenerator/data/
zee_config.cmnd --outputLevel 6 --seed 0 -evt <n_events> --pileupAvg <average-pileup>
--bc_id_start -8 --bc_id_end 7" -o zee.root -mt <n_threads> -n <n_jobs>
```

Setting values for the parameters <n_events>, <average-pileup>, <n_threads>, <n_jobs>

> prun_job.py -c "generator.py --filter Zee -i generator/PythiaGenerator/data/zee_config.cmnd --outputLevel 6 --seed 0 --evt 30 --pileupAvg 30 --bc_id_start -8 --bc_id_end 7" -o zee.root -mt 3 -n 5

For **reco**,

> reco_trf.py -i zee.root --outputLevel 6 -nt 5 -o reco_zee.root

To inspect the root file, it is necessary to copy from docker to any place in your computer (go to this place to copy the root file)
> sudo docker cp fde23f718bae:/code/lorenzett/reco_zee.root /Users/thalesoliveira/Documents

fde23f718bae is the bash number.

To build a new image

- Go to docker directory of lorenzett and create a new image: mkdir new_image,
- Inside the other images, you find Dockerfile and setup_env.sh,
- Copy them to the folder of new_image,
- Edit Dockerfile and go to line 88 (below)

```
RUN mkdir /code && cd /code && git clone https://github.com/jodafons/lorenzett.git.
```

- Change the path to lorenzett of your GitHub repository

```
RUN mkdir /code && cd /code && git clone https://github.com/your_github/lorenzett.git
```

- Save and quit the file.

> docker build --compress -t your_guthub/lorenzett:latest .

PS: In general it appears errors associated to “did not find a path, etc …”, in this case it is recommended to repeat the above command (I did it 3 times - Yara).

The compilation takes some hours. The final message can be

“ ---> 83be27243ea4
Successfully built 83be27243ea4
Successfully tagged yaamaral/lorenzett:latest

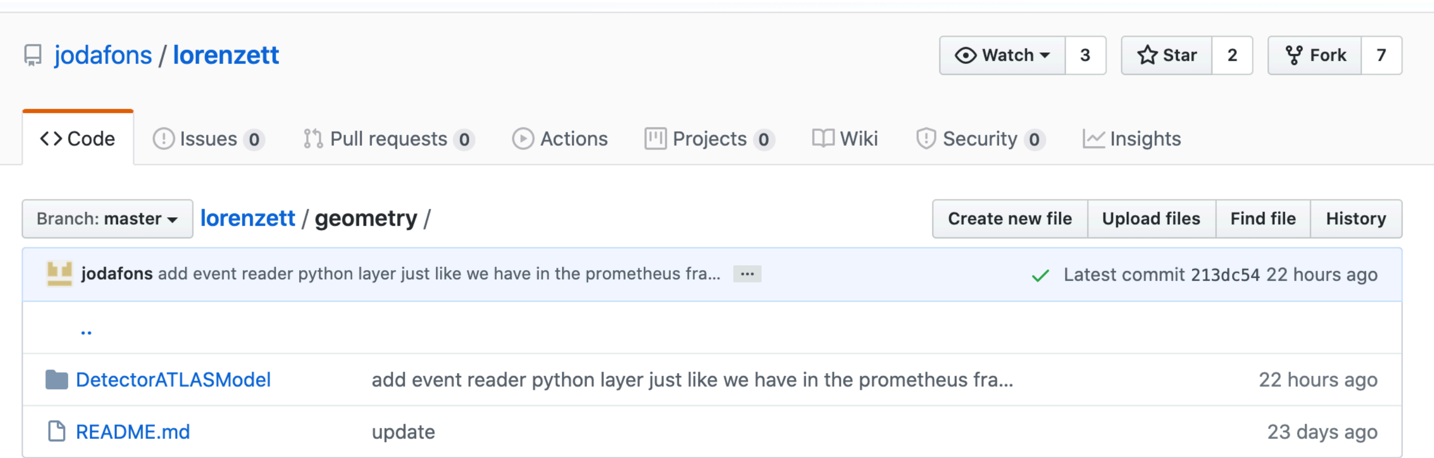
To run the image,
> docker run -it tmenezes/lorenzett /bin/bash

To modify the detector parameters and run it in the new image,

Creating a new directory:

The link https://github.com/your_username/lorenzett/tree/master/geometry has the DetectorATLASModel, containing all the configurations of the calorimeter model (materials, volumes, size,…)

The idea is build another calorimeter model to run in the lorenzett framework. We need to create a new DetectorModel directory (next slide).

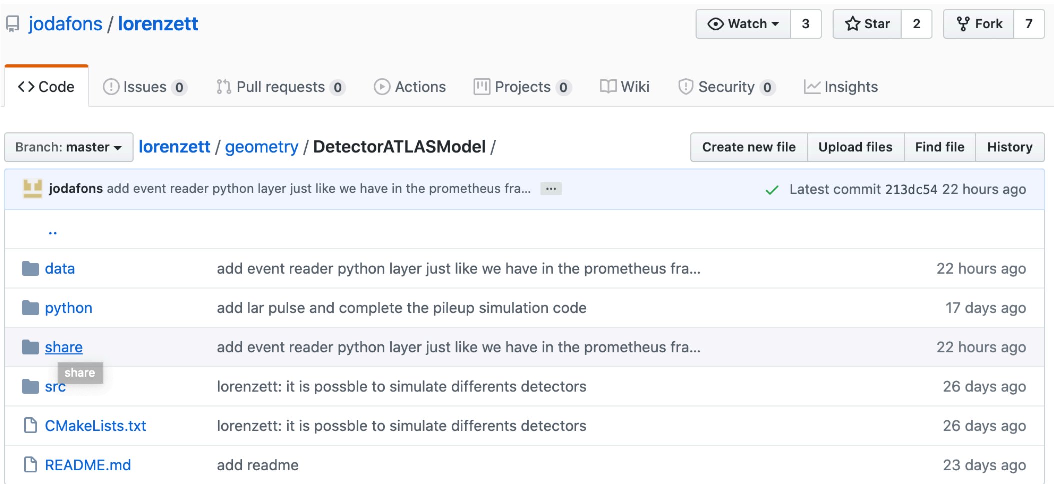


We will create a directory for the new calorimeter in the geometry directory that you made the clone of the lorenzett (your_path/lorenzett/geometry). The idea is copy from the DetectorATLASModel and modify some of its files.

mkdir DetectorGenericModel
cd DetectorGenericModel

To copy the directories:
cp -r your_path/lorenzett/geometry/DetectorATLASModel/* .

To copy the files:
cp your_path/lorenzett/geometry/DetectorATLASModel/* .



- data and python no modifications;
- share contains the calorimeter granularity (granularity_generator.py) if we need to modify in the early future;
- src contains the source information of the code, we will make some modifications in this directory.

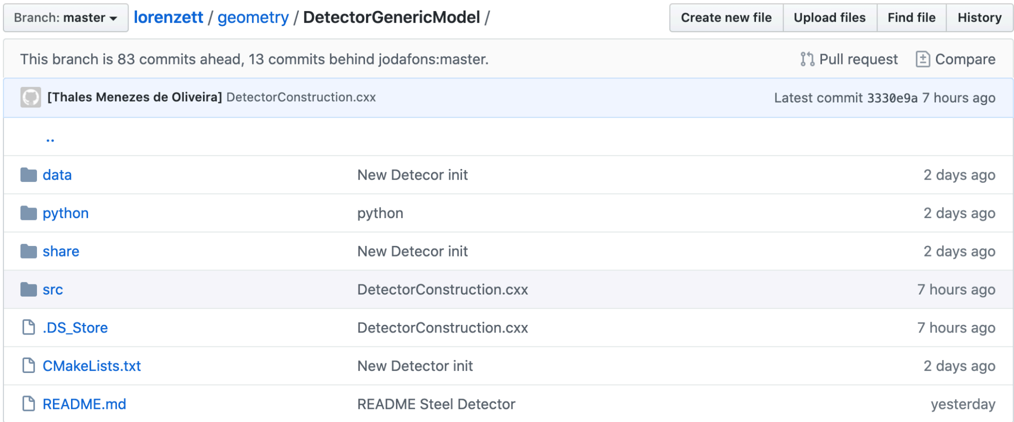
After the creation of the directory, we need to add it in the compilation list, given by CMakeLists.txt, in the main lorenzett directory.

In order to achieve this, we need to add the following lines after the already present structure.

add_subdirectory(geometry/DetectorGenericModel)

$<TARGET_OBJECTS:DetectorGenericModel>

After the copy of the files and directories, you will find the following structure inside your new directory, similar to the original directory.



The first thing that we need to modify is the CMakeLists.txt inside the new calorimeter directory in order to build the modifications of this directory (the new calorimeter).

We will modify the following lines.

```
file(GLOB_RECURSE HEADERS src/C*.h src/
DetectorATLASConstruction.h) >>
file(GLOB_RECURSE HEADERS src/C*.h src/
DetectorGenericConstruction.h )
```

```
ROOT_GENERATE_DICTIONARY(DetectorATLASModelDict ${HEADERS}
LINKDEF ${CMAKE_CURRENT_SOURCE_DIR}/src/LinkDef.h  MODULE
DetectorATLASModel)
```

>>

```
ROOT_GENERATE_DICTIONARY(DetectorGenericModelDict ${HEADERS}
LINKDEF ${CMAKE_CURRENT_SOURCE_DIR}/src/LinkDef.h  MODULE
DetectorGenericModel)
```

```
add_library(DetectorATLASModel  OBJECT ${SOURCES}
DetectorATLASModelDict.cxx) >>
add_library(DetectorGenericModel  OBJECT ${SOURCES}
DetectorGenericModelDict.cxx)

install(FILES ${HEADERS}  DESTINATION DetectorATLASModel) >>
install(FILES ${HEADERS}  DESTINATION DetectorGenericModel)

gaugi_install_python_modules( ${CMAKE_CURRENT_SOURCE_DIR}/python
DetectorATLASModel) >>
gaugi_install_python_modules( ${CMAKE_CURRENT_SOURCE_DIR}/python
DetectorGenericModel)
```

Inside the DetectorGenericModel, located in lorenzett/geometry/DetectorGenericModel/src we will find three codes named.

LinkDef.h, DetectorATLASConstruction.h, DetectorATLASConstruction.cxx

First we need to modify the names.

```
mv DetectorATLASConstruction.h DetectorGenericConstruction.h
mv DetectorATLASConstruction.cxx DetectorGenericConstruction.cxx
```

We will modify those three codes.

FIRST: LinkDef.h

```
#include “src/DetectorATLASConstruction.h" >>
#include “src/DetectorGenericConstruction.h"

#pragma link C++ class DetectorATLASConstruction+; >>
#pragma link C++ class DetectorGenericConstruction+;
```

SECOND: DetectorGenericConstruction.h

This code carries the main function of the calorimeter building. The CreateBarrel function is responsible to define regions of the calorimeter which is composed by a LogicalVolume, active and absorber materials, number of layers, thicknesss of the active and passive regions.

void CreateBarrel(G4LogicalVolume *worldLV,	
	std::string name,
	G4Material *defaltMaterial,
	G4Material *absorberMaterial,
	G4Material *gapMaterial,
	int nofLayers,
	double absoThickness,
	double gapThickness,
	double calorRmin,
	double calorZ,
	G4ThreeVector center_pos,
	G4Region* region);

In this code we will do the following modifications.

```
#ifndef DetectorATLASConstruction_h >>
#ifndef DetectorGenericConstruction_h

#define DetectorATLASConstruction_h >>
#define DetectorGenericConstruction_h

class DetectorATLASConstruction : public
G4VUserDetectorConstruction, public MsgService >>
class DetectorGenericConstruction : public
G4VUserDetectorConstruction, public MsgService
```

In the public declaration.

```
DetectorATLASConstruction(std::string); >>
DetectorGenericConstruction(std::string);

virtual ~DetectorATLASConstruction(); >>
virtual ~DetectorGenericConstruction();
```

THIRD: DetectorGenericConstruction.cxx (line in the left)

```
2 – #include “DetectorATLASConstruction.h" >>
#include “DetectorGenericConstruction.h"

29 – G4GlobalMagFieldMessenger*
DetectorATLASConstruction::m_magFieldMessenger = 0; >>
G4GlobalMagFieldMessenger*
DetectorGenericConstruction::m_magFieldMessenger = 0;
32 –
DetectorATLASConstruction::DetectorATLASConstruction(std::string
name) >>
DetectorGenericConstruction::DetectorGenericConstruction(std::stri
ng name)

42 –  DetectorATLASConstruction::~DetectorATLASConstruction() >>
DetectorGenericConstruction::~DetectorGenericConstruction()]

46 – G4VPhysicalVolume* DetectorATLASConstruction::Construct() >>
G4VPhysicalVolume* DetectorGenericConstruction::Construct()

55 – void DetectorATLASConstruction::DefineMaterials() >> void
DetectorGenericConstruction::DefineMaterials()

98 – G4VPhysicalVolume* DetectorATLASConstruction::DefineVolumes()
>> G4VPhysicalVolume* DetectorGenericConstruction::DefineVolumes()

428 – void DetectorATLASConstruction::ConstructSDandField() >>
void DetectorGenericConstruction::ConstructSDandField()

440 –  void DetectorATLASConstruction::CreateBarrel >> void
DetectorGenericConstruction::CreateBarrel

458 – G4Exception("DetectorATLASConstruction::DefineVolumes()",
"MyCode0001", FatalException, msg);
>>G4Exception("DetectorGenericConstruction::DefineVolumes()",
"MyCode0001", FatalException, msg);
```

Until now we replicated the DetectorATLASModel structure and we have a new calorimeter model to build and modify.

We will change the absorber material for the hadronic part of this new calorimeter, the ATLAS built use Iron and in this Generic Model we will use Steel.

In order to do that we will add new things and modify the DetectorGenericModel.cxx

First:

Add the component elements for the steel. In DetectorGenericConstruction::DefineMaterials()

```
G4Element* elMn = new G4Element("Manganese","Mn", 25., 54.94*g/mole);

G4Element* elSi = new G4Element("Silicon","Si", 14., 28.09*g/mole);

G4Element* elCr = new G4Element("Chromium","Cr", 24., 52.00*g/mole);

G4Element* elNi = new G4Element("Nickel","Ni", 28., 58.70*g/mole);

G4Element* elFe = new G4Element("Iron","Fe", 26., 55.85*g/mole);
```

And to build the steel.

```
G4Material* steel = new G4Material("Stainless Steel", 8.02 * g/cm3, 5); \\

steel->AddElement(elMn, 0.02);

steel->AddElement(elSi, 0.01);

steel->AddElement(elCr, 0.19);

steel->AddElement(elNi, 0.10);

steel->AddElement(elFe, 0.68);
```

The DetectorGenericConstruction.cxx creates, using the CreateBarrel function, all regions of the calorimeter (PS + 3 EM + Boundaries + 3 HAD (including the extended regions)).

Each one of these regions carries three kinds of materials. Default, absorber and active(gap).

The following declarations are already defined in the code, will showed here just for illustration.

```
G4Region* deadMatBeforeCal = new  G4Region("DeadMatBeforeECal");

G4Region* presampler = new G4Region("PS");

G4Region* em1 = new G4Region("EM1");

G4Region* em2 = new G4Region("EM2");

G4Region* em3 = new G4Region("EM3");

G4Region* deadMaterialBeforeHCal = new
G4Region("DeadMaterialBeforeHCal");

G4Region* had1 = new G4Region("HAD1");

G4Region* had2 = new G4Region("HAD2");

G4Region* had3 = new G4Region("HAD3");
```

```
CreateBarrel( worldLV,                              The Logical Volume
    "EM1",                                          Region Name
    G4Material::GetMaterial("Galactic"),            The default material,
gap between the modules
    G4Material::GetMaterial("G4\_Pb"),              The absorber material
    G4Material::GetMaterial("liquidArgon"),         The active material
    16,                                              No of layers, not
the calorimeter layers
    1.51*mm,                                         The thickness of the
absorber region
    4.49*mm,                                         The thickness of the
active region
    150.*cm,                                         The initial radius
of the calorimeter
    6.8*m ,                                          The length along the
beam axis
    G4ThreeVector(0,0,0),                            The position of
center of the region. (origin)
    em1)                                             The G4Region
```

In the ATLASModel we have, for the Electromagnetic and Hadronic Calorimeters, for instance

For EM:

```
G4Material::GetMaterial("Galactic"), // default

G4Material::GetMaterial("G4_Pb"), // absorber

G4Material::GetMaterial("liquidArgon"), // gap
```

For HAD

```
G4Material::GetMaterial("Galactic"),// default

G4Material::GetMaterial("G4_Fe"), // absorber

G4Material::GetMaterial("PLASTIC SCINTILLATOR"), //. active
```

In order to change the absorber material for the hadronic calorimeter. All of the lines which contains the G4_Fe should be replaced by Stainless Steel, the name that we gave to the our material.

```
G4Material::GetMaterial("G4_Fe"), // absorber >>
G4Material::GetMaterial(“Stainless Steel"), // absorber
```

In order to upload the modifications in git:

```
git add .

git commit -m "New commit"

git push origin master
```

To run the bash:

```
docker run -it your_username/lorenzett /bin/bash

source setup_envs.sh
```

Now you have a new environment with your git modifications.

With all the modifications done and compiled, we have to run the framework

The scripts are located in lorenzett/scripts

Branch: master ▾ lorenzett / scripts / — Create new file | Upload files | Find file | History

This branch is 85 commits ahead, 15 commits behind jodafons:master. — Pull request | Compare

[Thales Menezes de Oliveira] scripts — Latest commit 3f6cc9c yesterday

..		
generator.py	update all scripts	16 days ago
prun_job.py	add merge in prun job	2 months ago
reco_cells.py	scripts	yesterday
reco_trf.py	scripts	yesterday

The commands to run the scripts.

The generator step:

```
prun_job.py -c "generator.py --filter Zee -i generator/PythiaGenerator/data/zee_config.cmnd --outputLevel 6 --seed 0 --evt 30 --pileupAvg 30 --bc_id_start -8 --bc_id_end 7" -o zee.root -mt 3 -n 5
```

The reco step:

```
reco_trf.py -i zee.root --outputLevel 6 -nt 5 -o reco_zee.root
```

But the scripts are configured to only one calorimeter. The idea now is to adapt the scripts to account for these new calorimeters.

For the reco_cells.py and reco_trf.py

The original:

```
from DetectorATLASModel import DetectorConstruction as ATLAS
from DetectorATLASModel import CaloCellBuilder

acc = ComponentAccumulator(“ComponentAccumulator",
                            ATLAS(“GenericATLASDetector"),
                            RunVis=args.visualization,
                            NumberOfThreads =
args.numberOfThreads,
                            OutputFile = args.outputFile)
```

Modifications:

Add the following line

```
parser.add_argument('--cal','--calorimeter', action='store',
dest='Calorimeter', required = False, help = "Choose the
calorimeter")
```

Taking care with the indentation. The idea here is to add one more argument to the reco_trf.py which allow us to choose the calorimeter to use.

Importing the three models.

```
from DetectorATLASModel import DetectorConstruction as ATLAS
from DetectorGenericModel import DetectorConstruction as Generic
from DetectorScintiModel import DetectorConstruction as Scinti
```

Choosing the model by the new argument, we have to add the lines.

```
if args.Calorimeter == “ATLAS":
  from DetectorATLASModel import CaloCellBuilder
  acc = ComponentAccumulator(“ComponentAccumulator",
  ATLAS(“GenericATLASDetector"),
  RunVis=args.visualization,
  NumberOfThreads = args.numberOfThreads,
  OutputFile = args.outputFile)

if args.Calorimeter == “Generic":
  from DetectorGenericModel import CaloCellBuilder
  acc = ComponentAccumulator(“ComponentAccumulator",
  Generic(“GenericDetector"),
  RunVis=args.visualization,
  NumberOfThreads = args.numberOfThreads,
  OutputFile = args.outputFile)

if args.Calorimeter == “Scintillator":
  from DetectorATLASModel import CaloCellBuilder
  acc = ComponentAccumulator(“ComponentAccumulator",
```

```
Scintillator(“ScintiDetector"),
RunVis=args.visualization,
NumberOfThreads = args.numberOfThreads,
OutputFile = args.outputFile)
```

Now, with an extra argument we have to run the reco_trf.py in the following way

reco_trf.py -i zee.root --outputLevel 6 -nt 2 -o reco_zee.root --cal ATLAS >> ATLAS.log
reco_trf.py -i zee.root --outputLevel 6 -nt 2 -o reco_zee.root --cal Generic >> Generic.log
reco_trf.py -i zee.root --outputLevel 6 -nt 2 -o reco_zee.root --cal Scintillator >> Scintillator.log

The arguments in reco_trf.py

```

parser.add_argument('-i','--inputFile', action='store', dest='inputFile', required = False,
                    help = "The event input file generated by the Pythia event generator.")

parser.add_argument('-o','--outputFile', action='store', dest='outputFile', required = False,
                    help = "The reconstructed event file generated by lzt/geant4 framework.")

parser.add_argument('--outputLevel', action='store', dest='outputLevel', required = False, type=int, default=1,
                    help = "The output level messenger.")

parser.add_argument('-nt','--numberOfThreads', action='store', dest='numberOfThreads', required = False, type=int, default=
                    help = "The number of threads")

parser.add_argument('--evt','--numberOfEvents', action='store', dest='numberOfEvents', required = False, type=int, default=
                    help = "The number of events to apply the reconstruction.")

parser.add_argument('--visualization', action='store_true', dest='visualization', required = False,
                    help = "Run with Qt interface.")

parser.add_argument('--cal','--calorimeter', action='store', dest='Calorimeter', required = False,
                    help = "Choose the calorimeter")

```

The original structure in reco_trf.py

```
from DetectorATLASModel import DetectorConstruction as ATLAS
from DetectorATLASModel import CaloCellBuilder



acc = ComponentAccumulator("ComponentAccumulator",
                           ATLAS("GenericATLASDetector"),
                           RunVis=args.visualization,
                           NumberOfThreads = args.numberOfThreads,
                           OutputFile = args.outputFile)

```

Since the idea is to adapt the script to run it with a calorimeter as a new argument, we have to define a new one and run the reco_trf.py with this new argument. The script have to run some specific lines based on the calorimeter choice.

The modified code should look this way

```
from DetectorATLASModel import DetectorConstruction as ATLAS
from DetectorGenericModel import DetectorConstruction as Generic
from DetectorScintiModel import DetectorConstruction as Scinti


if args.Calorimeter == "ATLAS":

  from DetectorATLASModel import CaloCellBuilder

  acc = ComponentAccumulator("ComponentAccumulator",
                             ATLAS("GenericATLASDetector"),
                             RunVis=args.visualization,
                             NumberOfThreads = args.numberOfThreads,
                             OutputFile = args.outputFile)

if args.Calorimeter == "Generic":

  from DetectorGenericModel import CaloCellBuilder

  acc = ComponentAccumulator("ComponentAccumulator",
                             Generic("GenericATLASDetector"),
                             RunVis=args.visualization,
                             NumberOfThreads = args.numberOfThreads,
                             OutputFile = args.outputFile)

if args.Calorimeter == "Scintillator":

  from DetectorScintiModel import CaloCellBuilder

  acc = ComponentAccumulator("ComponentAccumulator",
                             Scinti("ScintiDetector"),
                             RunVis=args.visualization,
                             NumberOfThreads = args.numberOfThreads,
                             OutputFile = args.outputFile)
```

If you want to upload some log files or pdf files to your git there is a .gitignore file in your directory which can be edited by:

vi .gitignore

This file contains the extensions which will not be uploaded for git, if you want to upload some log and pdf files all you have to do is cut the lines

*.pdf
*.log

We can use the log files to confirm what model we took in the reco step.

For the ATLAS-like.

```
Region <EM1> --  -- appears in <World> world volume
 Root logical volume(s) : EM1
 Materials : Galactic G4_Pb liquidArgon
```

```
Region <EM2> --  -- appears in <World> world volume
 Root logical volume(s) : EM2
 Materials : Galactic G4_Pb liquidArgon
```

```
Region <EM3> --  -- appears in <World> world volume
 Root logical volume(s) : EM3
 Materials : Galactic G4_Pb liquidArgon
```

```
Region <HAD1> --  -- appears in <World> world volume
 Root logical volume(s) : HAD1 HAD1_Extended HAD1_Extended
 Materials : Galactic G4_Fe PLASTIC_SCINTILLATOR
```

```
Region <HAD2> --  -- appears in <World> world volume
 This region is in the mass world.
G4FastSimulationManager[0], G4UserSteppingAction[0]
```

```
Region <HAD3> --  -- appears in <World> world volume
 Root logical volume(s) : HAD3 HAD3_Extended HAD3_Extended
 Materials : Galactic G4_Fe PLASTIC_SCINTILLATOR
```

For the Generic Calorimeter, with steel instead iron in the Hadronic part.

```
Region <EM1> --  -- appears in <World> world volume
 Root logical volume(s) : EM1
 Materials : Galactic G4_Pb liquidArgon
```

```
Region <EM2> --  -- appears in <World> world volume
 Root logical volume(s) : EM2
 Materials : Galactic G4_Pb liquidArgon
```

```
Region <EM3> --  -- appears in <World> world volume
 Root logical volume(s) : EM3
 Materials : Galactic G4_Pb liquidArgon
```

```
Region <HAD1> --  -- appears in <World> world volume
 Root logical volume(s) : HAD1 HAD1_Extended HAD1_Extended
 Materials : Galactic Stainless Steel PLASTIC SCINTILLATOR
```

```
Region <HAD2> --  -- appears in <World> world volume
 Root logical volume(s) : HAD2 HAD2_Extended HAD2_Extended
 Materials : Galactic Stainless Steel PLASTIC SCINTILLATOR
```

```
Region <HAD3> --  -- appears in <World> world volume
 Root logical volume(s) : HAD3 HAD3_Extended HAD3_Extended
 Materials : Galactic Stainless Steel PLASTIC SCINTILLATOR
```

We also can build a new Calorimeter, using scintillator as active material for the EM part, instead the default liquid argon.

```
Region <EM1> --  -- appears in <World> world volume
 Root logical volume(s) : EM1
 Materials : Galactic G4_Pb PLASTIC SCINTILLATOR
```

```
Region <EM2> --  -- appears in <World> world volume
 Root logical volume(s) : EM2
 Materials : Galactic G4_Pb PLASTIC SCINTILLATOR
```

```
Region <EM3> --  -- appears in <World> world volume
 Root logical volume(s) : EM3
 Materials : Galactic G4_Pb PLASTIC SCINTILLATOR
```

```
Region <HAD1> --  -- appears in <World> world volume
 Root logical volume(s) : HAD1 HAD1_Extended HAD1_Extended
 Materials : Galactic G4_Fe PLASTIC SCINTILLATOR
```

```
Region <HAD2> --  -- appears in <World> world volume
 Root logical volume(s) : HAD2 HAD2_Extended HAD2_Extended
 Materials : Galactic G4_Fe PLASTIC SCINTILLATOR
```

```
Region <HAD3> --  -- appears in <World> world volume
 Root logical volume(s) : HAD3 HAD3_Extended HAD3_Extended
 Materials : Galactic G4_Fe PLASTIC SCINTILLATOR
```