EUROPEAN SPALLATION SOURCE

Geant4/Garfield interface

https://garfieldpp.web.cern.ch/garfieldpp/examples/geant4-interface/

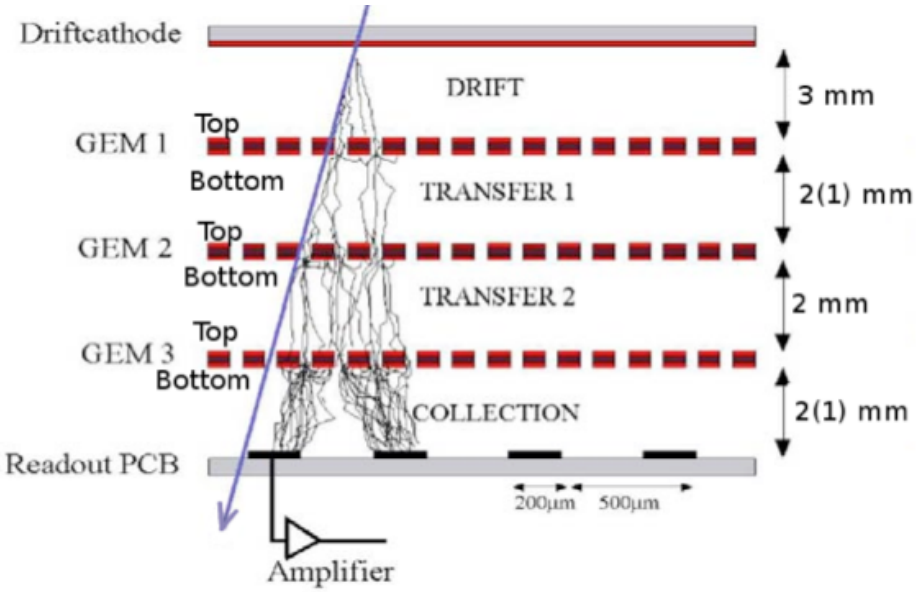
05.09.2016

Dorothea Pfeiffer

Steps in gas detector simulation

To simulate a gaseous detector, the following five processes are relevant:

1. Primary high energy particle ionization
2. Forming of ionization clusters in gas
3. Drift of ionization clusters to amplification stage and detector read out
4. Amplification/creation of additional ionization via avalanch
5. Forming of electronic signal at read out



Why interfacing Geant4/Garfield?

EUROPEAN SPALLATION SOURCE

- Only the first process is provided by the Geant4 kernel
- Depending on the aim of the simulation, the deposited energy and the location where the primary ionization occurs can already be sufficient (1)
- For (2) – (5), the user has to provide either her own code, or can use an existing software package like Garfield (Fortran) or Garfield++
- Garfield is used because of its predictive power of optimizing drift properties of electrons and ions, and calculating avalanches and the gain of the detector
- Possible to exchange data between Geant4 and Garfield via data files, but interfacing the two programs is a more elegant way

How interfacing Geant4/Garfield?

EUROPEAN SPALLATION SOURCE

- **Idea**: Use Geant4 provides physics parameterization capabilities (example Par01)
- The general idea of parameterization in Geant4 is to create a region, where the user can provide her own implementation of the physics and the detector response. In the case of the Geant4/Garfield++ interface, the user implementation of the detector response is based on Garfield++.
- **Problem**: At which point is the control transferred from Geant4 to Garfield++?
- **Best option**: The primary ionization takes place in Geant4, electron are subsequently killed in Geant4 and then handled by Garfield++

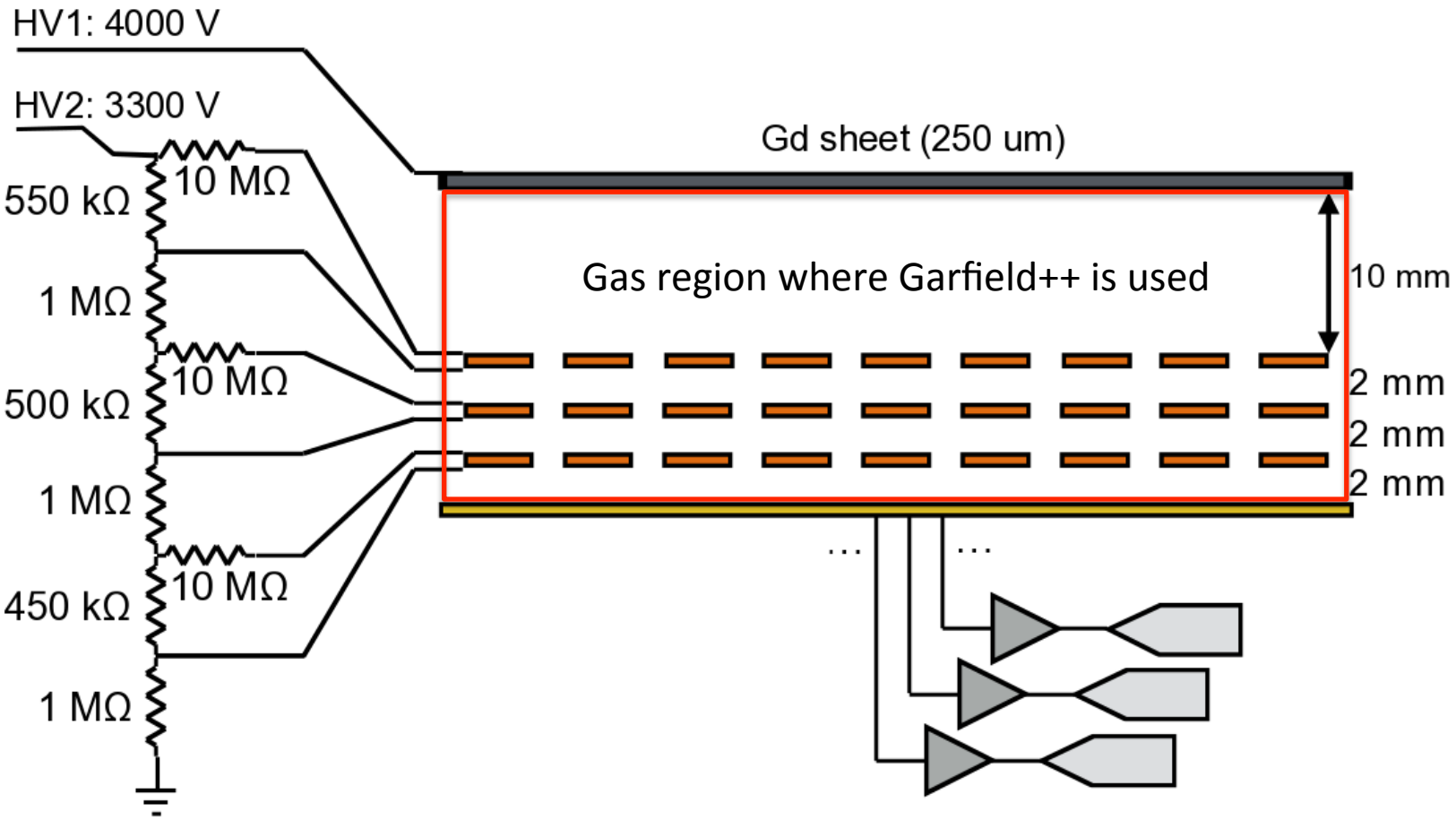
Geant4 EM physics model

- In Geant4 an additional low energy physics model like the G4PAIPhotModel or G4PAIModel has to be active in the region.
- The primary particle that the user wants to detect thus ionizes the gas and creates electrons.
- These electrons are subsequently killed in Geant4, and recreated as delta electrons in Garfield++ using the TransportDeltaElectron() function of Heed.
- Alternative: Use Heed for primary ionization, but approach limited to relativistic particles and lacking Coulomb scattering

G4Region

- A G4Region or envelope consists of one or more G4LogicalVolumes, that often correspond to the volumes of sub detectors.
- To simulate e.g. the detector response of an ionization chamber behind a target, the gas volume inside the chamber would form a G4Region, in which Garfield++ is used as parameterized model instead of Geant4.
- For the remainder of the geometry like the target or the vessel of the chamber, the Geant4 physics is valid.

G4Region


HV1: 4000 V
HV2: 3300 V
550 kΩ
10 MΩ
1 MΩ
500 kΩ
10 MΩ
1 MΩ
450 kΩ
10 MΩ
1 MΩ
Gd sheet (250 um)
Gas region where Garfield++ is used
10 mm
2 mm
2 mm
2 mm
…
…

G4Region

- The G4Region is created during the detector construction:

```
class IonisationChamberDetectorConstruction: public G4VUserDetectorConstruction
{
        G4VPhysicalVolume* Construct()
        {
        ...
                G4Region* regionGarfield = new G4Region("RegionGarfield");
                regionGarfield->AddRootLogicalVolume(fLogicalVolumeGas);
        ...
        }
};
```

G4VFastSimulationModel

- To create the parameterized physics model, the user has to create a new class derived from G4VFastSimulationModel (Responsible in the Geant4 team: Alberto Ribon) .
- Name G4VFastSimulationModel implies that the parameterized model is usually simpler and thus faster than the full Geant4 tracking.
- Garfield++ model a slow model, so name should not be taken in its literal sense.
- G4VFastSimulationModel has three pure virtual methods, which must be overridden in the Garfield++ model:
  - G4bool IsApplicable(const G4ParticleDefinition& particleType)
  - G4bool ModelTrigger(const G4FastTrack& fastTrack)
  - void DoIt(const G4FastTrack& fastTrack, G4FastStep& fastStep)

G4VFastSimulationModel

```
#include "G4Electron.hh"
#include "G4SystemOfUnits.hh"

class GarfieldModel: public G4VFastSimulationModel
{
        G4bool IsApplicable(const G4ParticleDefinition& particleType)
        {
                //GarfieldModel is applicable to electrons
                return &partDef == G4Electron::ElectronDefinition();
        }

        G4bool ModelTrigger(const G4FastTrack& fastTrack)
        {
                //GarfieldModel should be triggered if kinetic energy is larger than 1 MeV
                double ekin = fastTrack.GetPrimaryTrack()->GetKineticEnergy() / MeV;
                if(ekin >= 1. * MeV)
                {
                        return true;
                }
                else
                {
                        return false;
                }

        }

        void DoIt(const G4FastTrack& fastTrack, G4FastStep& fastStep)
        {
                //The details of the Garfield model are implemented here

        }
};
```

G4VFastSimulationModel

- During the detector construction, an instance of the GarfieldModel is created. The G4Region, that was created before, is given as a parameter to the constructor of the GarfieldModel.

```
class IonisationChamberDetectorConstruction
{
        G4VPhysicalVolume* Construct()
        {
        ...
                GarfieldModel* garfieldModel = new GarfieldModel("GarfieldModelIonisationChamber", regionGarfield);
        ...
        }
};
```

G4VUserPhysicsList

- G4FastSimulationManagerProcess is the physics process of the parameterization.
- It serves as interface between the Geant4 tracking and the user parameterization.
- At tracking time, the G4FastSimulationManagerProcess collaborates with the G4VFastSimulationManager of the current volume, and allows the user model to be triggered.
- In the user physics list, the G4FastSimulationManagerProcess must now be added as a discrete process to the process list of the particles for which the model shall apply. A discrete process is a process that uses only the PostStepDoIt method.

G4VUserPhysicsList

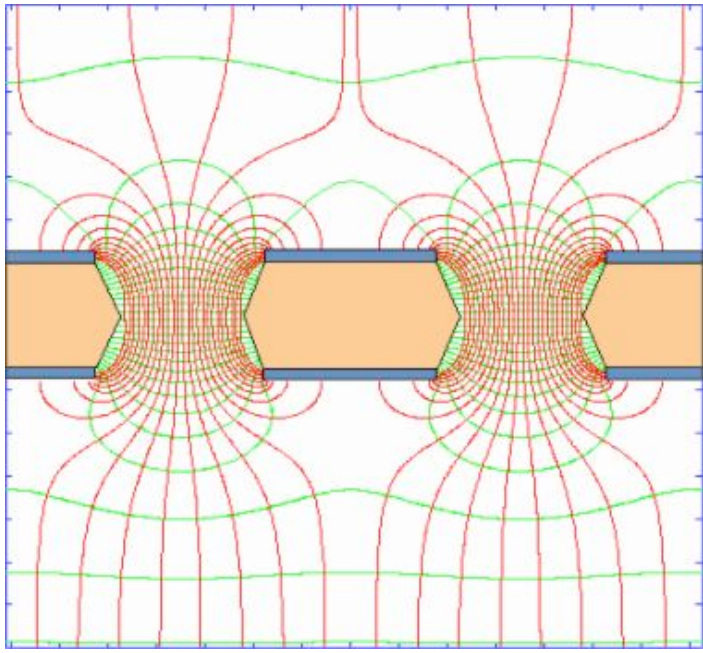
```
ist: public G4VModularPhysicsList


erisation()

 an instance of the G4FastSimulationManagerProcess
mulationManagerProcess* fastSimProcess_garfield = new G4FastSimulationManagerProc

cleIterator->reset();
*theParticleIterator)())

G4ParticleDefinition* particle = theParticleIterator->value();
G4ProcessManager* pmanager = particle->GetProcessManager();

//Select the particle for which the parametrisation applies
if (particle->GetParticleName() == "e-")
{
        //Add the G4FastSimulationManagerProcess as discrete process
        pmanager->AddDiscreteProcess(fastSimProcess_garfield);
}
```

Garfield++ Physics

- The method void DoIt(const G4FastTrack& fastTrack, G4FastStep& fastStep) contains the Garfield++ simulation, that has to be created by the user depending on the individual needs
- The only material that is used in Garfield for the electron transport is the gas, the other materials do not have any influence on the physics behaviour. The geometry is only important for the E- and B-field definition, and to define the limit of the gas volume. The electrons are not transported beyond the volume.

Geometry and Fields

- Only part of the geometry that is needed is the shape of the gaseous region, and that can be created from the Geant4 geometry via GDML export
- Elements like the wires of a wire chamber or GEM foils can be implemented in Geant4, if important for the simulation (interaction probability). In Garfield++ they will only be present in the form of a field map (GEM foil) or the analytical field definition (wires)
- E- and B-fields can be also defined in Geant4, if they influence the primary particle



Field of two GEM holes

Compilation and linking

EUROPEAN SPALLATION SOURCE

```
#----------------------------------------------------------------------------
# Setup Geant4 include directories and compile definitions
# Setup include directory for this project
#
include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/include)
include_directories($ENV{GARFIELD_HOME}/Include)

link_directories($ENV{GARFIELD_HOME}/Library)

#----------------------------------------------------------------------------
# Locate sources and headers for this project
# NB: headers are included so they will show up in IDEs
#
file(GLOB sources ${PROJECT_SOURCE_DIR}/src/*.cc)
file(GLOB headers ${PROJECT_SOURCE_DIR}/include/*.hh)
file(GLOB headers $ENV{GARFIELD_HOME}/Include/*.hh)

#----------------------------------------------------------------------------
# Add the executable, and link it to the Geant4 libraries
#
add_executable(IonisationChamber IonisationChamber.cc ${sources} ${headers})

target_link_libraries(IonisationChamber ${Geant4_LIBRARIES} -lGarfield -lgfortran )
```

Result is one C++ Geant4 program that is linked against Garfield++

Limitations of Geant4/Garfield interface

Comments by Vladimir Ivanchenko and Alberto Ribon:

- It is fine the way you use the **PAIPhotModel** via the method **G4EmConfigurator::SetExtraEmModel** .
- However, it is safer here to set **0.0** (instead of 0.01*keV) the minimum energy in which the model is applicable.
- The **PAIPhotModel**, although it is recommended by its author (Vladimir Grichine) has been less tested and validated than **PAIModel**, so it could be worth considering also the latter as a possible alternative model.
- The approach of using the fast-sim machinery of Geant4 for interfacing it to Garfield could not work optimally in the case of dense particle flows, for instance very close to the collision point, or in an electromagnetic or hadronic showers.
- In fact, in these cases, Garfield should consider the combined energy deposits produced by all "primary" particles that pass through the gas detector in the same time window.

Users of Geant4/Garfield interface

- **Geant4/Garfield++ interface example is now part of the Garfield++ distribution (Heinrich Schindler)**
  - **https://garfieldpp.web.cern.ch/garfieldpp/**
- **Interface code included in the latest Geant4 VMC version (Ivana Hrivnacova)**
  - **https://root.cern.ch/geant4-vmc**
- **Simulation projects:**
  - ESS: Simulation of Gd-GEM for NMX (DP)
  - NASA Goddard Space Flight Center, Astroparticle Physics Laboratory: Simulation of a large (2m x 2m x 2m) micro-well detector (essentially a GEM capped at one end). Primary interest is in tagging the pair products (e^-/e^+) from 5-200 MeV gamma-rays (Andrei Hanu)
  - GSI R3B collaboration: Simulation of active target detector, where the He4 gas constitutes both the target and the detection medium (Oleg Kiselev)
  - IHEP China: Simulation of B4C neutron detector (Liu Guanglei)
  - Mohammed V University Rabat: Fission Chamber (Arahmane Hanane)
  - Raman Research Institute, Bangalore: (Varun Bahal)

Has to be put into the framework!