SPY + "Initial detector".

HPgTPC Meeting

Eldwan Brianne
DESY
Hamburg, 15th June 2020

DEEP UNDERGROUND NEUTRINO EXPERIMENT

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)

Alan's and Hiro's proposal.

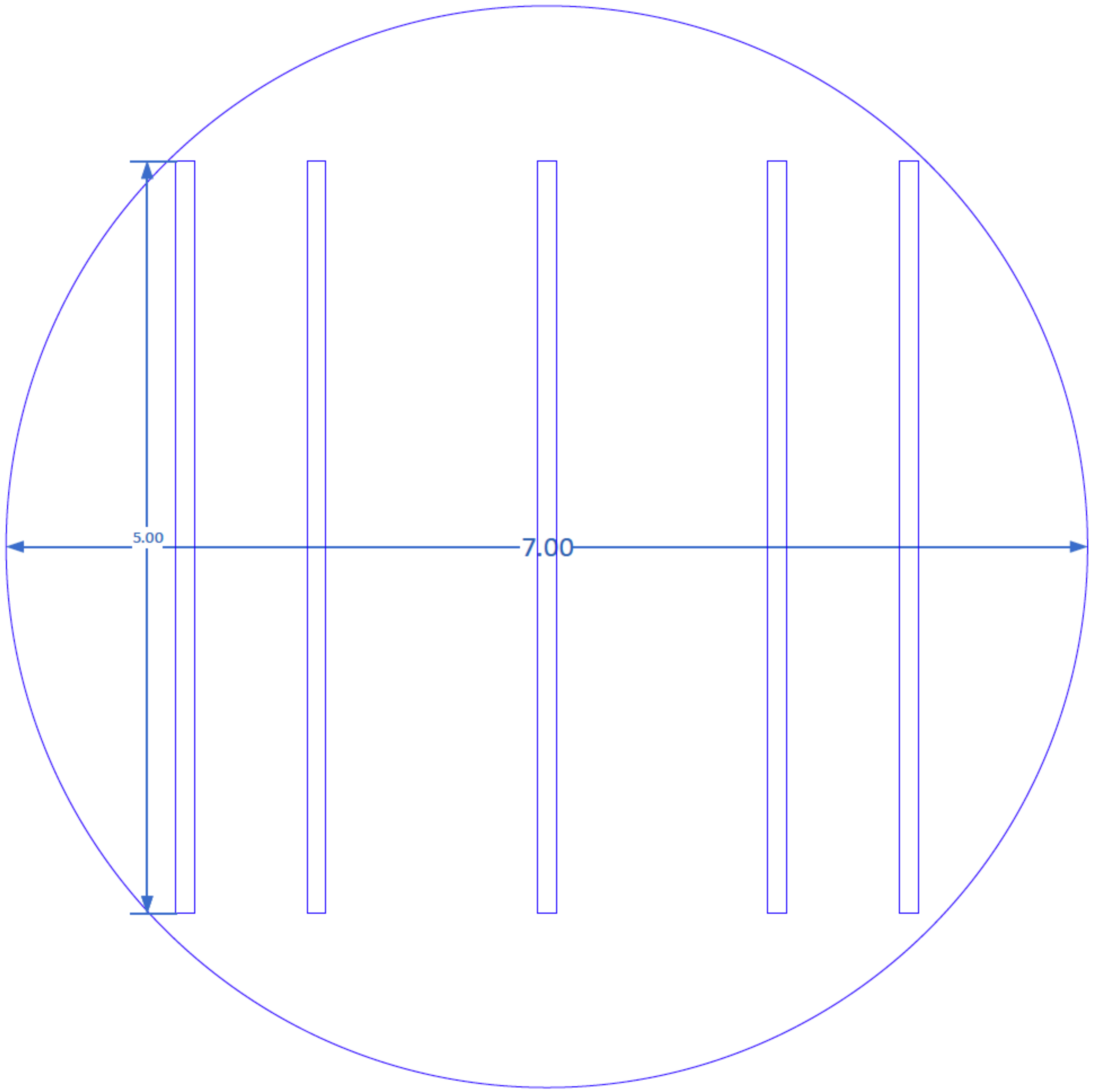
SPY from day one

- Alan and Hiro sent around a couple of weeks ago a nice presentation about the ND requirements
- Slides 21-22 shows that a Full ND is needed to achieve DUNE goals!
    - Only a SSRI would not be enough ➠ need significant running..
- Building a SSRI and then changing it (would have limited use after upgrade) is expensive…
- How to make best use of the money?
    - Secure the magnet first! ➠ the bigger piece and used for the lifetime of the experiment (even after)
    - Add a muon tracker inside to track the muons exiting the LAr
- Finally, upgrade to the Full ND after run 1? Or gradually?
- Reuse the muon tracker to tag backgrounds from the Hall (large surface area)

The detector geometry.

SPY + Minerva-like Sc layers

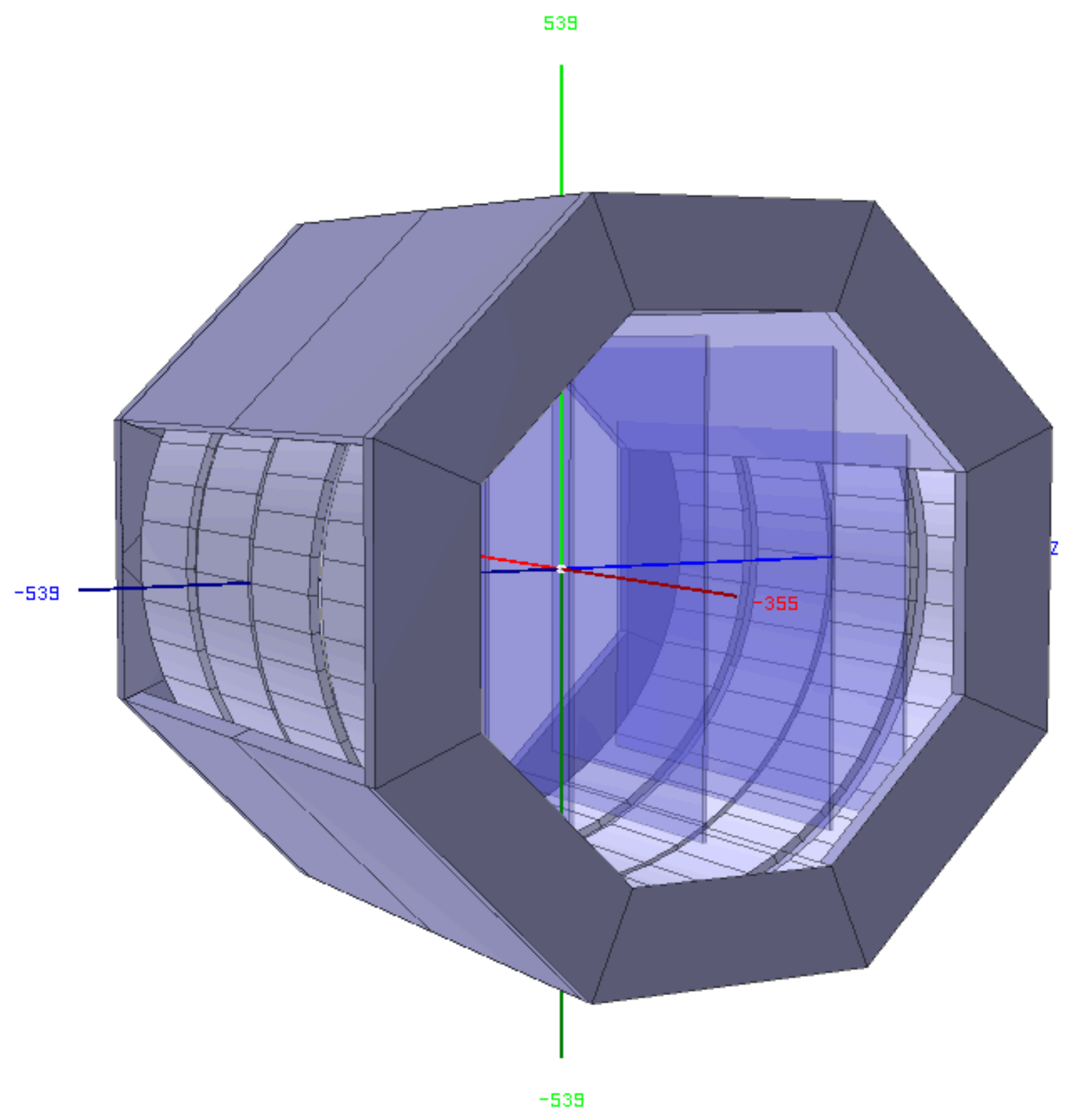
- The temporary MPD (soon new name) is as the following
  - The magnet as the SPY
    - 10 cm Al solenoid
    - an iron return yoke about 30 cm thick, integrating a muon id system ➠ (3 layers 10 cm iron, 1.67 cm Sc)
    - an open window in front of the LAr
    - 7 m in diameter maximum
  - Inside, 5 scintillator layers (6 m x 5 m) of 4 cm thickness segmented Minerva-like (triangles)
    - ➠ distance between layers is to be optimised for better tracking

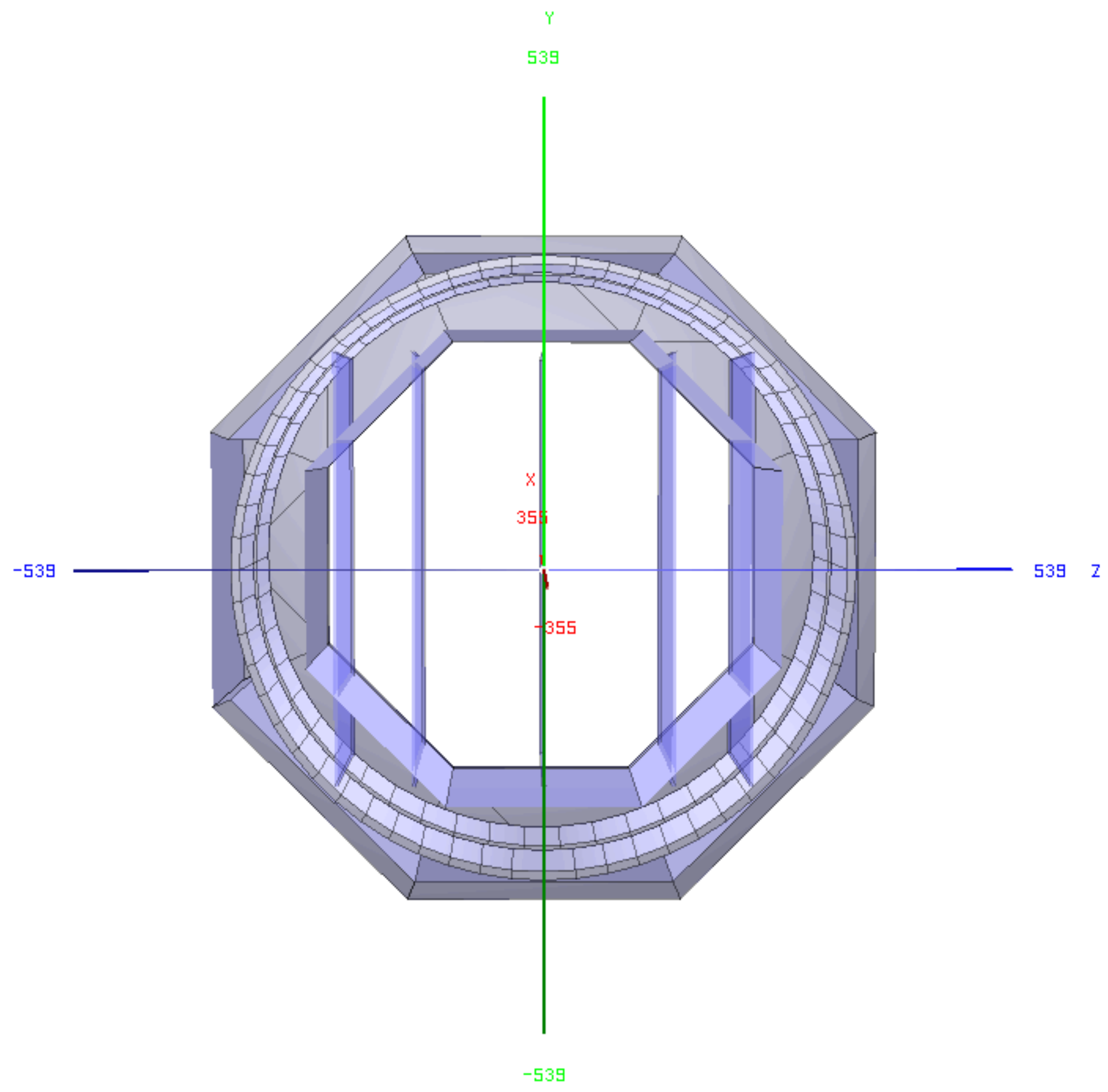


The detector geometry.

Implementation in dunendggd

- I have implemented the geometry in dunendggd
- Created its own class/config file
  - https://github.com/ebrianne/dunendggd/tree/2to3 /!\ changed to py3
- SPY + 5 Sc layers + 3 MuID layers
- Tracking layers position:
  - [Q('-240cm'), Q('-150cm'), Q('0cm'), Q('150cm'), Q('240cm')]
- Origin at (0, 0, 0) (MPD alone)
  - can be done in ND hall (just tell me)
- Already put into GArSoft
  - gdml/MPD_Standalone/Temporary_Det_SPY_wMuID.gdml





Conversion from edep-sim to GArSoft.

Heavy re-implementation...

- The current module handled only
  - the TPC/ECAL (need sensitive volume name on case by case…)
  - Segmentation of cells for the ECAL
  - Provided ghep files (in case generated with GENIE) for MCTruth/GTruth
- New needs
  - Handle the new Tracker and the MuID

Conversion from edep-sim to GArSoft.

Heavy re-implementation…

- The current module handled only
  - the TPC/ECAL (need sensitive volume name on case by case…)
  - Segmentation of cells for the ECAL
  - Provided ghep files (in case generated with GENIE) for MCTruth/ GTruth
- New needs
  - Handle the new Tracker and the MuID
    - Pretty easy (just added the cases for these) to collect the hits (data product same as ECAL as it uses Sc) ✅
    - Battled with my poor art experience with same data products in a producer (Thanks Tom!) ✅

```cpp
else if( d->first == "MuID_vol" ) {
    //MuonID detector in the SPY
    for (std::vector<TG4HitSegment>::const_iterator h = d->second.begin(); h != d->second.end(); ++h)
    {
        const TG4HitSegment *hit = &(*h);

        int trackID = hit->GetPrimaryId();
        double edep = VisibleEnergyDeposition(hit, fApplyBirks) * CLHEP::MeV / CLHEP::GeV;
        double time = (hit->GetStart().T() + hit->GetStop().T())/2 / CLHEP::s;
        double x = (hit->GetStart().X() + hit->GetStop().X())/2 /CLHEP::cm;
        double y = (hit->GetStart().Y() + hit->GetStop().Y())/2 /CLHEP::cm;
        double z = (hit->GetStart().Z() + hit->GetStop().Z())/2 /CLHEP::cm;

        if(edep == 0 || edep < fEnergyCut)
        continue;

        //Check if it is in the active material of the ECAL
        TGeoNode *node = fGeo->FindNode(x, y, z);//Node in cm...
        std::string VolumeName  = node->GetVolume()->GetName();
        std::string volmaterial = node->GetMedium()->GetMaterial()->GetName();
        if ( ! std::regex_match(volmaterial, std::regex(fECALMaterial)) ) continue;

        unsigned int layer = GetLayerNumber(VolumeName); //get layer number
        unsigned int slice = GetSliceNumber(VolumeName); // get slice number
        unsigned int det_id = 4;
        unsigned int stave = GetStaveNumber(VolumeName);
        unsigned int module = GetModuleNumber(VolumeName);

        std::array<double, 3> GlobalPosCM = {x, y, z};
        std::array<double, 3> LocalPosCM;
        gar::geo::LocalTransformation<TGeoHMatrix> trans;
        fGeo->WorldToLocal(GlobalPosCM, LocalPosCM, trans);

        LOG_DEBUG("ConvertEdep2Art")
        << "Sensitive volume " << d->first
        << " Hit " << hit
        << " in volume " << VolumeName
        << " in material " << volmaterial
        << " det_id " << det_id
        << " layer " << layer
        << " slice " << slice
        << " stave " << stave
        << " module " << module;

        gar::raw::CellID_t cellID = fGeo->GetCellID(node, det_id, stave, module, layer, slice, LocalPosCM);//encoding the cellID
        on 64 bits

        double G4Pos[3] = {0., 0., 0.}; // in cm
        G4Pos[0] = GlobalPosCM[0];
        G4Pos[1] = GlobalPosCM[1];
        G4Pos[2] = GlobalPosCM[2];

        gar::sdp::CaloDeposit calohit( trackID, time, edep, G4Pos, cellID );
        if(m_MuIDDeposits.find(cellID) != m_MuIDDeposits.end())
        m_MuIDDeposits[cellID].push_back(calohit);
        else {
            std::vector<gar::sdp::CaloDeposit> vechit;
            vechit.push_back(calohit);
```

Conversion from edep-sim to GArSoft.

Heavy re-implementation…

- The current module handled only
  - the TPC/ECAL (need sensitive volume name on case by case…)
  - Segmentation of cells for the ECAL
  - Provided ghep files (in case generated with GENIE) for MCTruth/ GTruth
- New needs
  - Handle the new Tracker and the MuID ✅
  - More segmentations (ECAL/Tracker/MuID … need to be more generic somehow)

Conversion from edep-sim to GArSoft.

Heavy re-implementation…

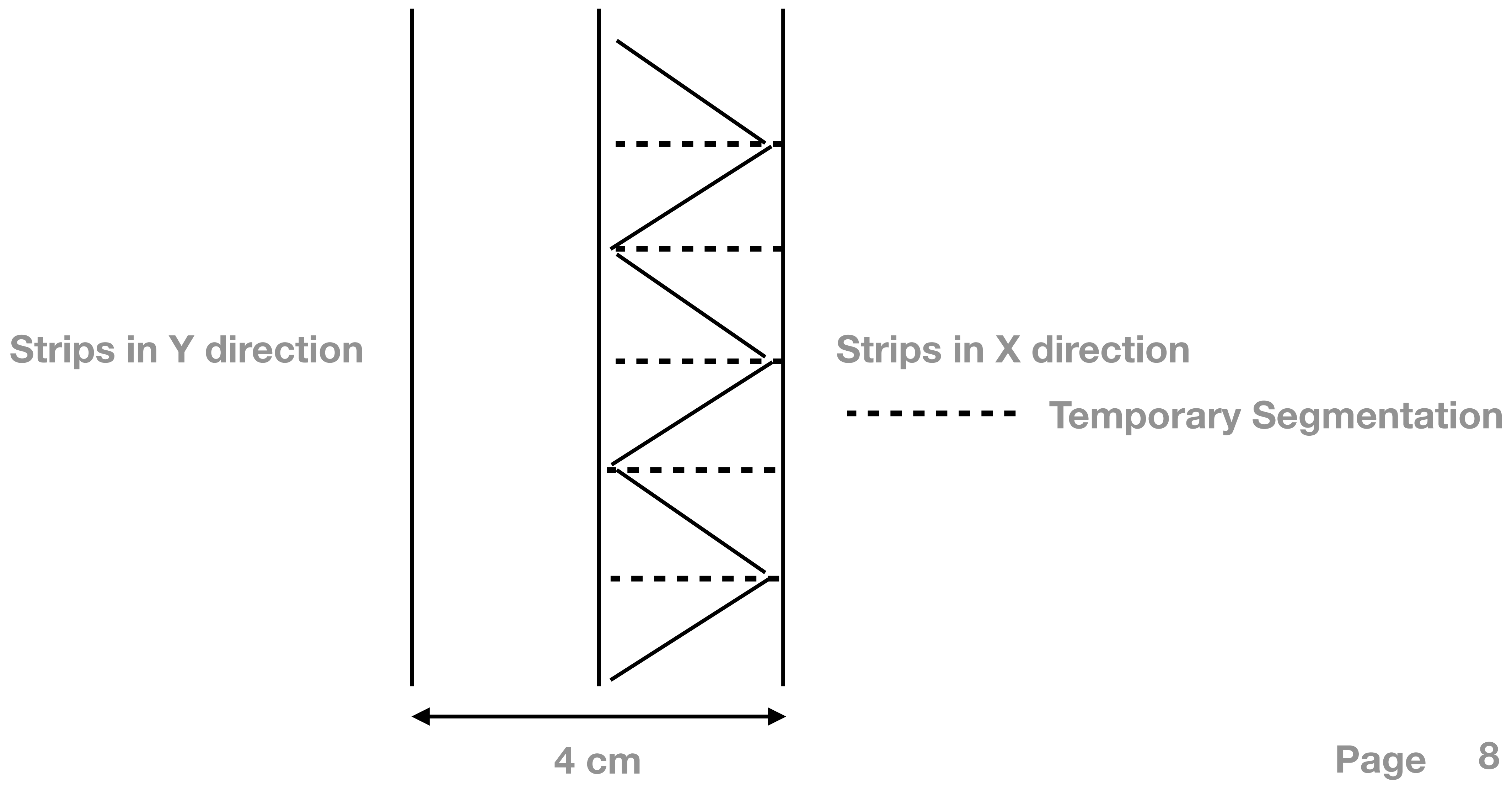
- The current module handled only
  - the TPC/ECAL (need sensitive volume name on case by case…)
  - Segmentation of cells for the ECAL
  - Provided ghep files (in case generated with GENIE) for MCTruth/ GTruth
- New needs
  - Handle the new Tracker and the MuID ✅
  - More segmentations (ECAL/Tracker/MuID … need to be more generic somehow)
    - Was a bit more complicated and required more work
    - Now, segmentation**s** are all initialised via fcl parameters (TPC is still automatic) ✅
    - Added segmentations for the MuID (strips along the X direction, minerva-like segmentation with triangles) ✅
    - Tracking layers for now segmented in cross-strips of 2x2 cm2 (in progress)

```
#include "ECALSegmentationAlg.fcl"
#include "MinervaSegmentationAlg.fcl"
#include "MuIDSegmentationAlg.fcl"

BEGIN_PROLOG

standard_mpd_segals:
{
ECALSegmentationAlgPars: @local::standard_ecalmultigridstripxysegalgpars
}

standard_mpd_spy_segals:
{
ECALSegmentationAlgPars: @local::standard_ecalmultigridstripxysegalgpars
MuIDSegmentationAlgPars: @local::standard_muidsegalgpars
}

standard_mpd_temporary:
{
MinervaSegmentationAlgPars: @local::standard_minervasegalgpars
MuIDSegmentationAlgPars: @local::standard_muidsegalgpars
}

END_PROLOG
```

ECAL only

MuID + ECAL

Minerva-like + MuID



Conversion from edep-sim to GArSoft.

Heavy re-implementation…

- The current module handled only
  - the TPC/ECAL (need sensitive volume name on case by case…)
  - Segmentation of cells for the ECAL
  - Provided ghep files (in case generated with GENIE) for MCTruth/GTruth
- New needs
  - Handle the new Tracker and the MuID ✅
  - More segmentations (ECAL/Tracker/MuID … need to be more generic somehow) ~ ✅
  - Particle gun for MCTruth

Conversion from edep-sim to GArSoft.

Heavy re-implementation…

- The current module handled only
  - the TPC/ECAL (need sensitive volume name on case by case…)
  - Segmentation of cells for the ECAL
  - Provided ghep files (in case generated with GENIE) for MCTruth/GTruth
- New needs
  - Handle the new Tracker and the MuID
  - More segmentations (ECAL/Tracker/MuID … need to be more generic somehow)
  - Particle gun for MCTruth
    - Pretty simple also as edep-sim contains it in the rootfile ✅
    - Link between MCTruth and MCParticles are made ✅

```
    //Case where things are made from particle gun! no ghep file provided. Need to create MCTruth object / GTruth object
    simb::MCTruth truth;
    truth.SetOrigin(simb::kSingleParticle);

    for (std::vector<TG4PrimaryVertex>::const_iterator t = fEvent->Primaries.begin(); t != fEvent->Primaries.end(); ++t)
    {
        TLorentzVector pos(t->Position.X() / CLHEP::cm, t->Position.Y() / CLHEP::cm, t->Position.Z() / CLHEP::cm, t->Position.T());

        for (std::vector<TG4PrimaryParticle>::const_iterator p = t->Particles.begin(); p != t->Particles.end(); ++p) {
            int trackid = p->GetTrackId();
            std::string primary("primary");

            TLorentzVector pvec(p->Momentum.Px() * CLHEP::MeV / CLHEP::GeV, p->Momentum.Py() * CLHEP::MeV / CLHEP::GeV, p-
            >Momentum.Pz() * CLHEP::MeV / CLHEP::GeV, p->Momentum.E() * CLHEP::MeV / CLHEP::GeV);

            simb::MCParticle part(trackid, p->GetPDGCode(), primary);
            part.AddTrajectoryPoint(pos, pvec);

            LOG_DEBUG("ConvertEdep2Art") << "Adding primary particle with "
            << " momentum " << part.P()
            << " position " << part.Vx() << " " << part.Vy() << " " << part.Vz();

            truth.Add(part);
        }
    }

    LOG_DEBUG("ConvertEdep2Art") << "Adding mctruth with "
    << " nParticles " << truth.NParticles()
    << " Origin " << truth.Origin();

    mctruthcol->push_back(truth);

    //Make a vector of mctruth art ptr
    art::Ptr<simb::MCTruth> MCTruthPtr = makeMCTruthPtr(mctruthcol->size() - 1);
    mctPtrs.push_back(MCTruthPtr);
}
```

What's left to be done?

To do list

- Geometry is done
- Conversion from edep-sim to GArSoft is done
  - segmentation to be finished in parallel - might need some rework with backgrounds (hit collection and time-stamping)
- To do
  - Hit reconstruction (energy, position, time) based on Minerva data
  - Track fitting and pattern recognition
  - Analysis using muons exiting the LAr (w/o and w backgrounds)

Backup Slides.