Final Project

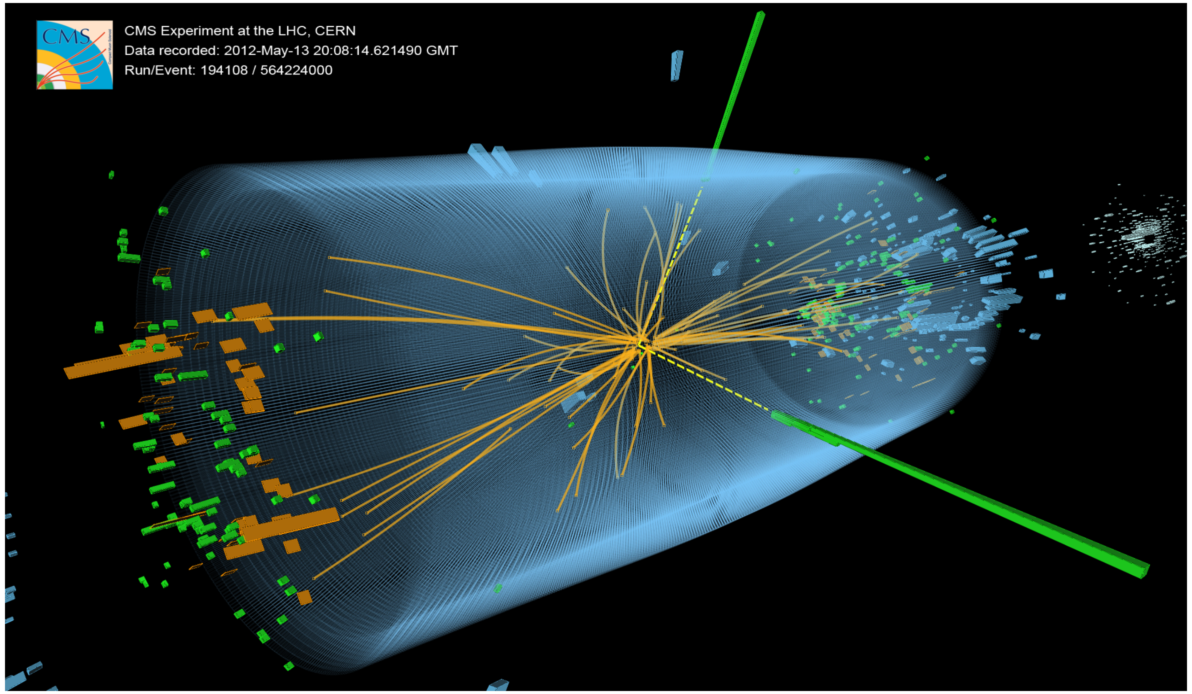
Kyungmin Park
Department of Physics

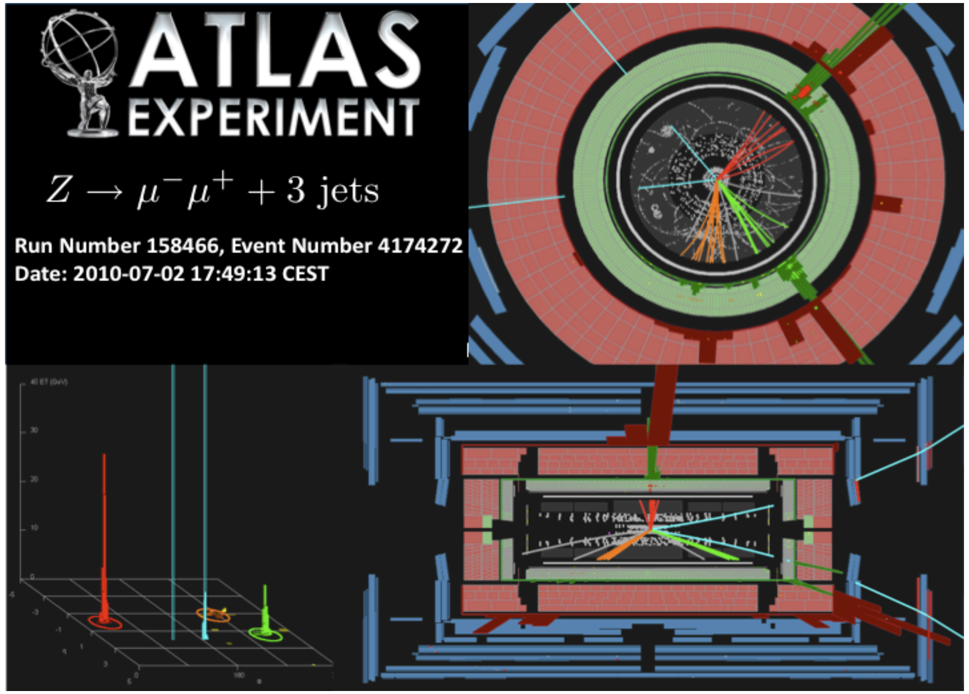
Overview

- Background Theory
- Data
- Computation
- Visualization

Event Display

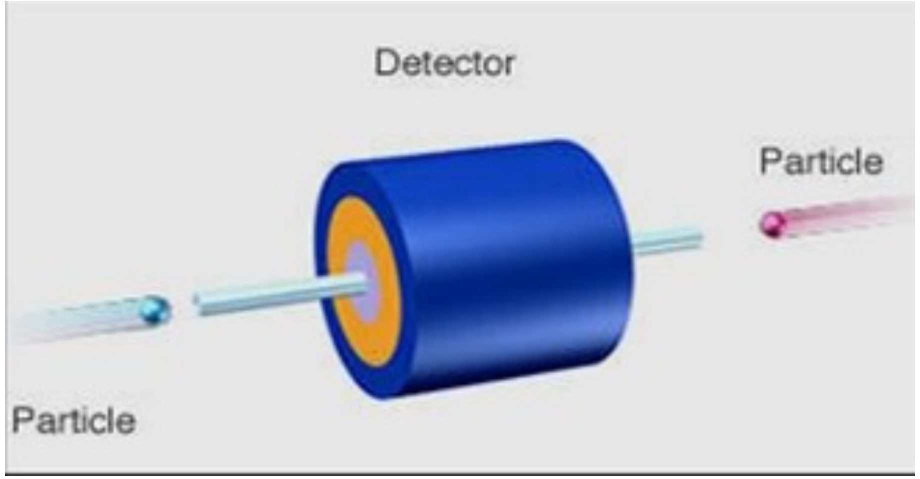
- Event display





Background Theory

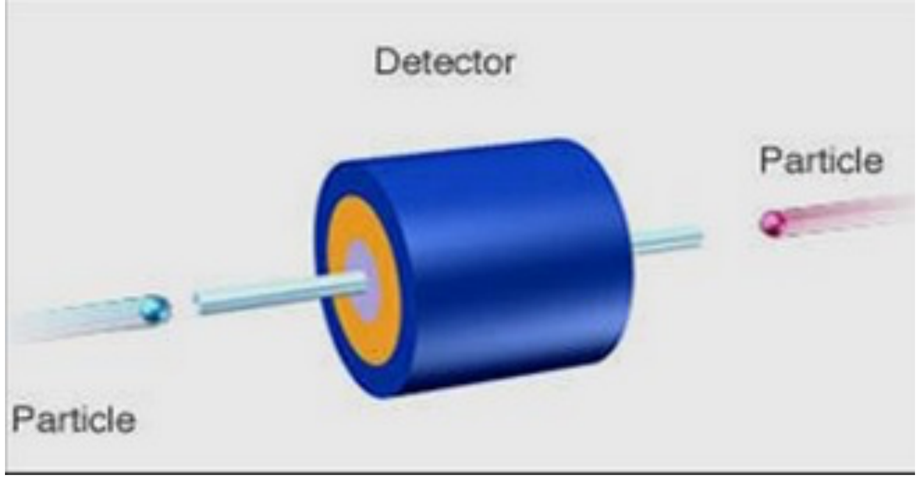
- Proton-Proton (bunch) head on collision



- **P_T = 0** → The total energy will be used to create new particles **($E = E_{beam1} + E_{beam2}$)**

Background Theory

- Proton-Proton (bunch) head on collision

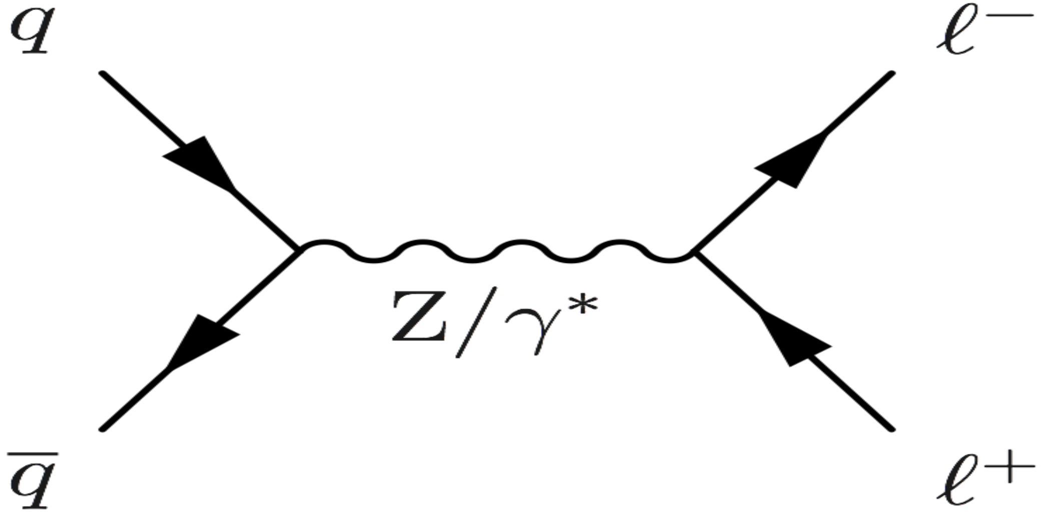


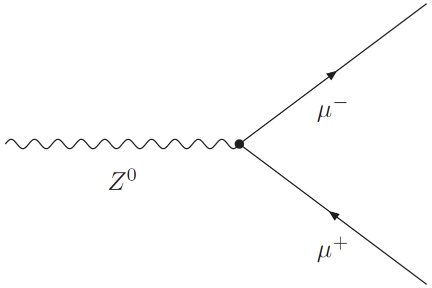
Draw proton – proton bunches moving toward each other

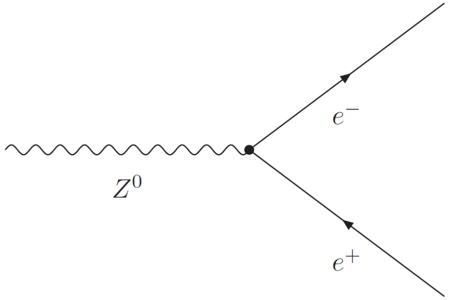
- $P_T = 0$ → The total energy will be used to create new particles ($E = E_{beam1} + E_{beam2}$)

Background Theory

- Z boson decay



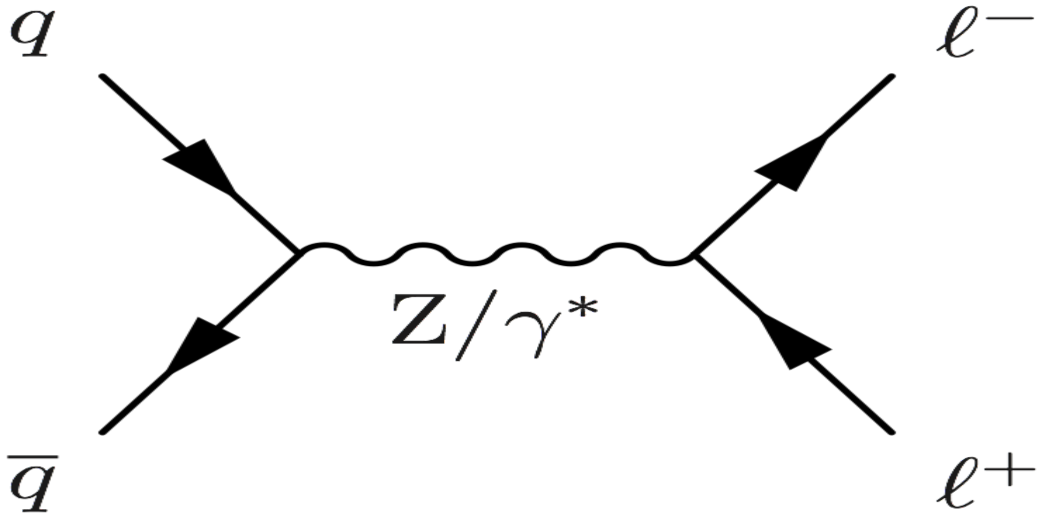


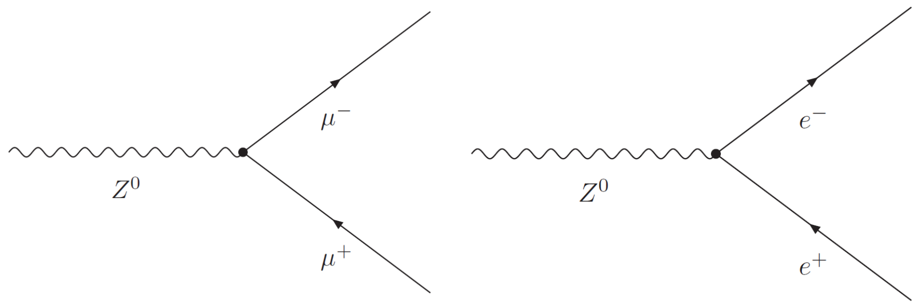


- proton proton > Z boson (mean lifetime: 3 x 10^{-25} sec)
- Z boson decaying into lepton-antilepton pair

Background Theory

- Z boson decay





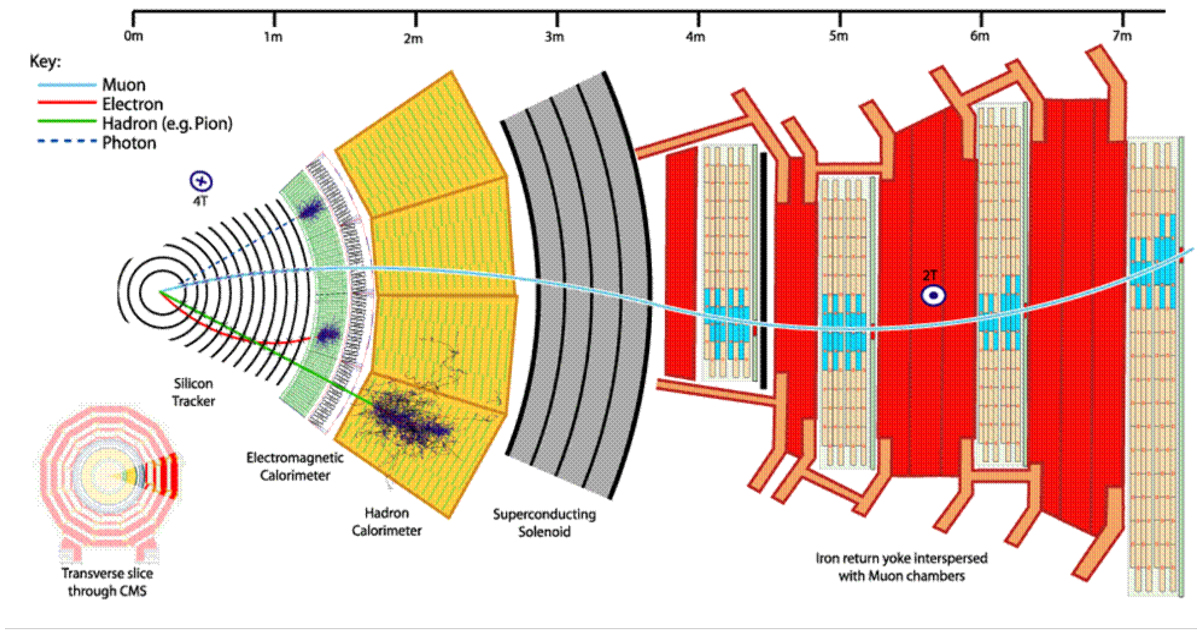
Draw lepton – antilepton pair after the collision

- proton proton > Z boson (mean lifetime: 3 x 10^{-25} sec)
- Z boson decaying into lepton-antilepton pair

Background Theory

- CMS Detector layers

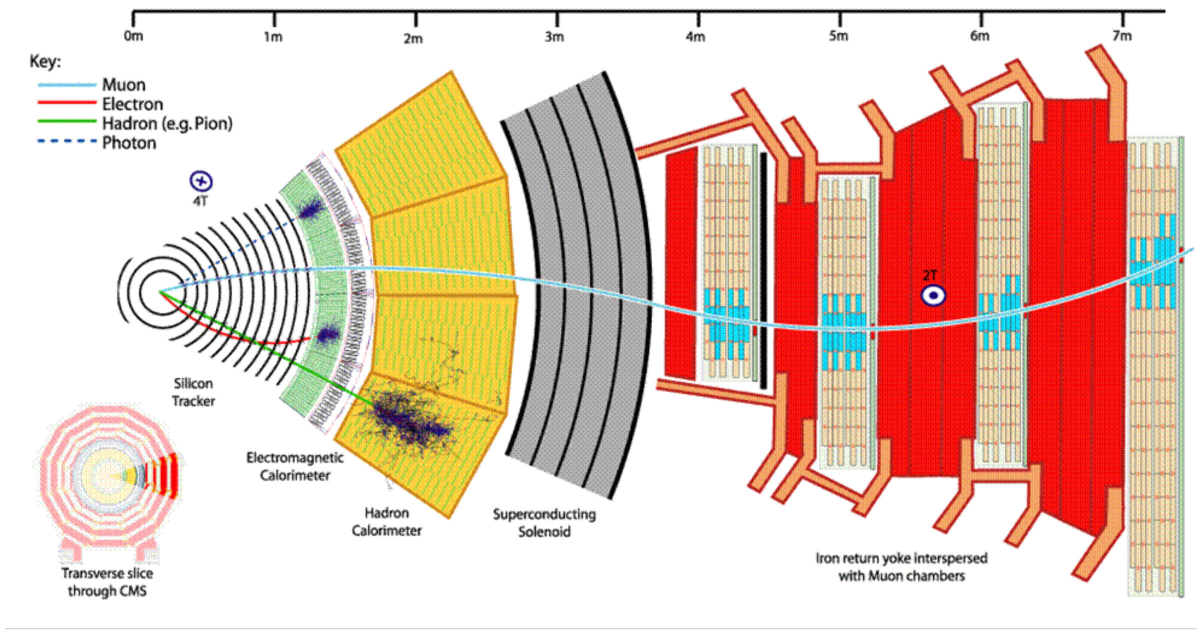
Silicon tracker → Electromagnetic calorimeter → Hadron calorimeter → Superconducting solenoid → muon chamber



Background Theory

- CMS Detector layers

Silicon tracker → Electromagnetic calorimeter → Hadron calorimeter → Superconducting solenoid → muon chamber



Draw four layers (ECAL, HCAL, superconducting solenoid, muon chamber)

Background Theory

- Particle trajectories

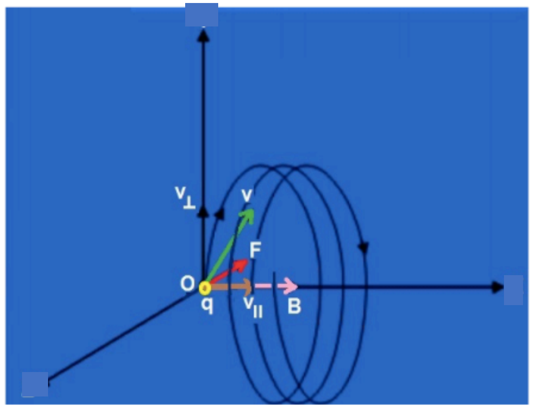
- Momentum of a particle (Px, Py, Pz) barely changes while it is moving inside the detector.

- Assume

- Uncharged particle moves towards the direction parallel to its initial momentum.

Background Theory

- Charged particle traces out a helical path due to the Lorentz force.



Taking $F = m \cdot v_\perp^2 / R$

(being R the radius of curvature in the plane perpendicular to the magnetic field)

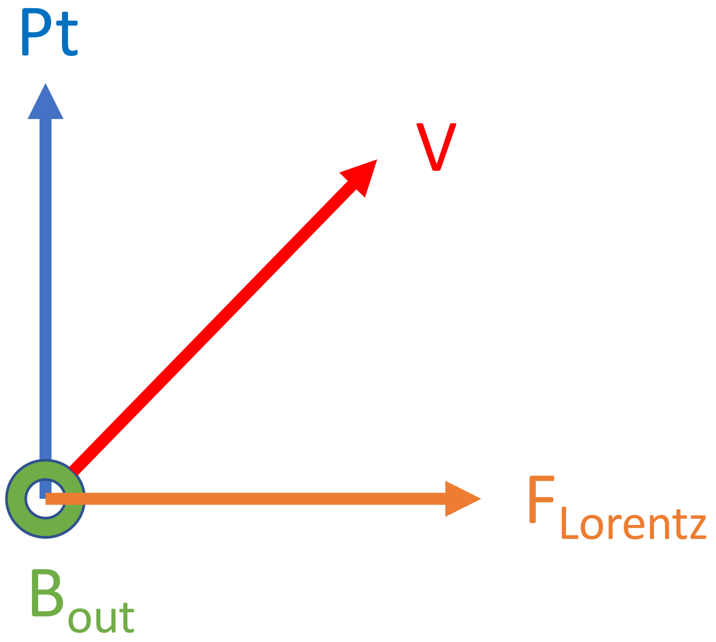
From (1) we have: $m \cdot v_\perp^2 / R = q \cdot v_\perp \cdot B$

So $m \cdot v_\perp = q \cdot R \cdot B$

Finally, the equation for the transverse momentum:

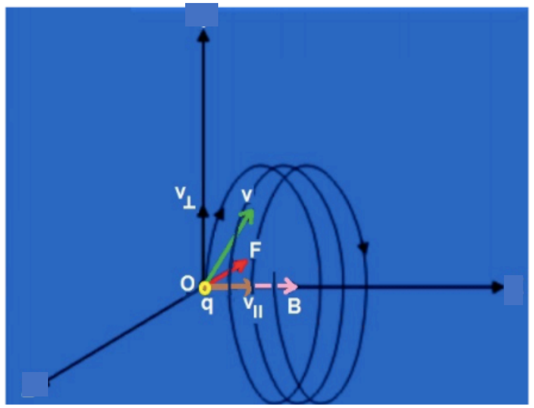
$$\mathbf{P_\perp = q \cdot R \cdot B}$$

The value of $P_\perp$ of a produced particle is conserved, so is a very important parmeter to be considered. Although this calculus has been made using a classical frame, the equation is the same using relativistic conditions.



Background Theory

- Charged particle traces out a helical path due to the Lorentz force.



Taking $F = m \cdot v_\perp^2 / R$

(being R the radius of curvature in the plane perpendicular to the magnetic field)

From (1) we have: $m \cdot v_\perp^2 / R = q \cdot v_\perp \cdot B$

So $m \cdot v_\perp = q \cdot R \cdot B$

Finally, the equation for the transverse momentum:

$$\mathbf{P_\perp = q \cdot R \cdot B}$$

The value of $P_\perp$ of a produced particle is conserved, so is a very important parmeter to be considered. Although this calculus has been made using a classical frame, the equation is the same using relativistic conditions.

Calculate the radius and the center of the circular path on x-y plane

→ Draw the helical path

Data

- Get simulation data using MadGraph/Delphes

1. Run MadGraph simulation

- generate p p > Z > mu+ mu-
- generate p p > Z > e+ e-

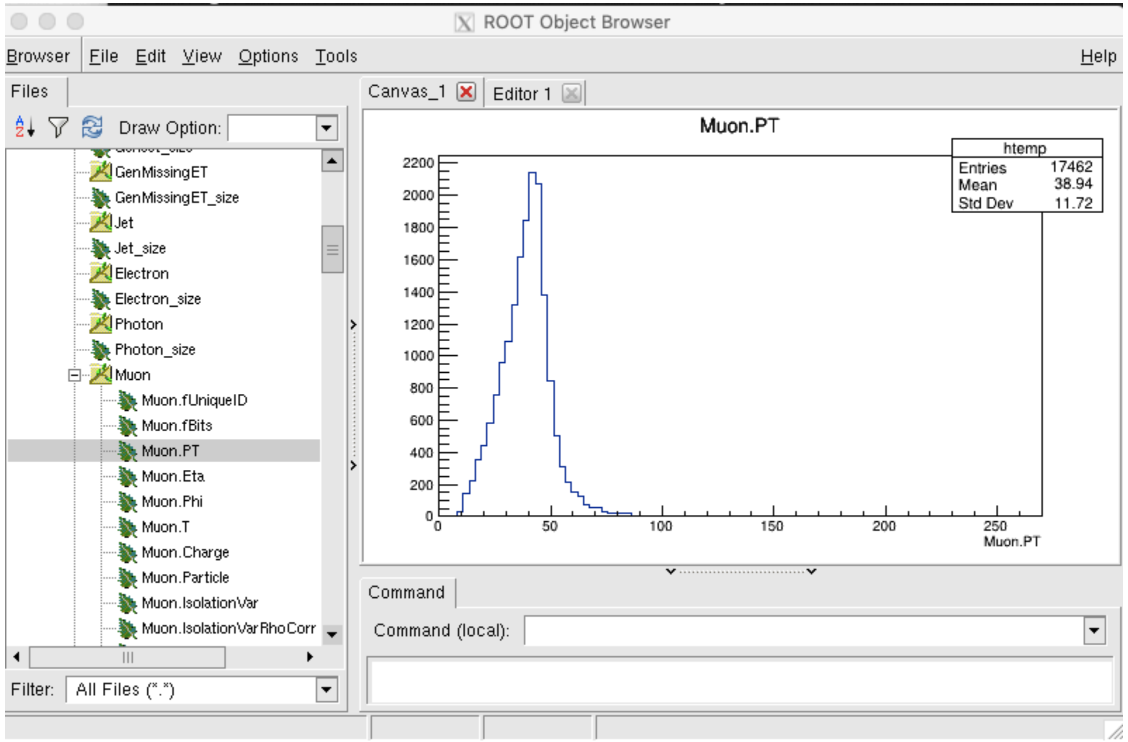
→ 10000 events for each

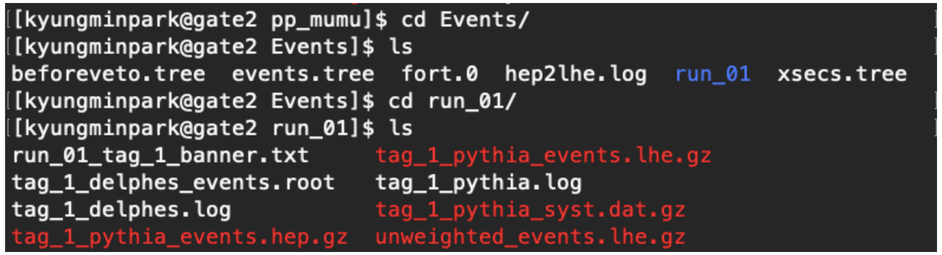
```
Defined multiparticle all = g u c d s u~ c~ d~ s~ a ve vm vt e- mu- ve~
 vm~ vt~ e+ mu+ t b t~ b~ z w+ h w- ta- ta+
MG5_aMC>
MG5_aMC>
MG5_aMC>generate p p > Z > mu+ mu-
INFO: Checking for minimal orders which gives processes.
INFO: Please specify coupling orders to bypass this step.
INFO: Trying process: g g > z > mu+ mu- WEIGHTED<=4 @1
INFO: Trying process: u u~ > z > mu+ mu- WEIGHTED<=4 @1
INFO: Process has 1 diagrams
INFO: Trying process: u c~ > z > mu+ mu- WEIGHTED<=4 @1
INFO: Trying process: c u~ > z > mu+ mu- WEIGHTED<=4 @1
INFO: Trying process: c c~ > z > mu+ mu- WEIGHTED<=4 @1
INFO: Process has 1 diagrams
INFO: Trying process: d d~ > z > mu+ mu- WEIGHTED<=4 @1
INFO: Process has 1 diagrams
INFO: Trying process: d s~ > z > mu+ mu- WEIGHTED<=4 @1
INFO: Trying process: s d~ > z > mu+ mu- WEIGHTED<=4 @1
INFO: Trying process: s s~ > z > mu+ mu- WEIGHTED<=4 @1
INFO: Process has 1 diagrams
INFO: Process u~ u > mu+ mu- added to mirror process u u~ > mu+ mu-
INFO: Process c~ c > mu+ mu- added to mirror process c c~ > mu+ mu-
INFO: Process d~ d > mu+ mu- added to mirror process d d~ > mu+ mu-
INFO: Process s~ s > mu+ mu- added to mirror process s s~ > mu+ mu-
4 processes with 4 diagrams generated in 0.058 s
Total: 4 processes with 4 diagrams
```

Data

- Get simulation data using MadGraph/Delphes

2. Get Delphes output root file





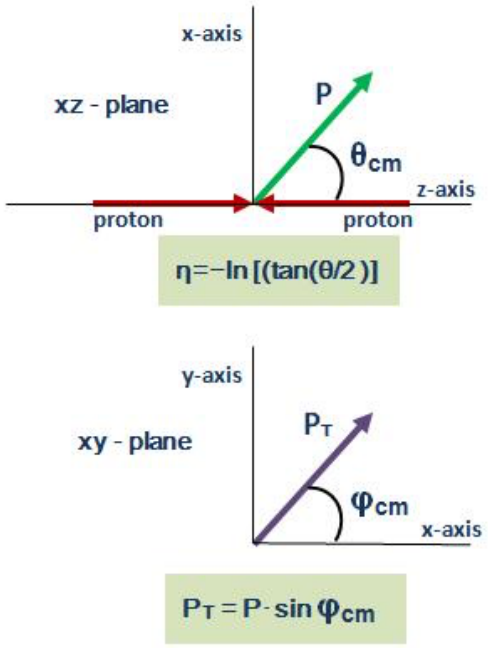
Data

- Get simulation data using MadGraph/delphes

3. Recreate the output file so that it can be read in any directory

```
gSystem->Load("/home/kyungminpark/MadGraph/delphes/libDelphes");

TFile *f = new TFile("ppTOmumu.root");
TTree *t = (TTree*)f->Get("Delphes");
TClonesArray *muons = 0;
TClonesArray *photons = 0;
t->SetBranchAddress("Muon", &muons);
t->SetBranchAddress("Photon", &photons);

TFile *newFile = new TFile("ztomumu.root","RECREATE");
TTree *muonTree = new TTree("data", "data");

Int_t NMuon;
Int_t Muon_Charge[2];
Double_t Muon_PT[2], Muon_Eta[2], Muon_Phi[2];
Double_t Muon_Px[2], Muon_Py[2], Muon_Pz[2];

Int_t NPhoton;
Double_t Photon_PT[2], Photon_Eta[2], Photon_Phi[2];
Double_t Photon_Px[2], Photon_Py[2], Photon_Pz[2], Photon_Energy[2];


muonTree->Branch("NMuon", &NMuon);
muonTree->Branch("Muon_Charge", Muon_Charge, "Muon_Charge[2]/I");
muonTree->Branch("Muon_PT", Muon_PT, "Muon_PT[2]/D");
muonTree->Branch("Muon_Eta", Muon_Eta, "Muon_Eta[2]/D");
muonTree->Branch("Muon_Phi", Muon_Phi, "Muon_Phi[2]/D");
muonTree->Branch("Muon_Px", Muon_Px, "Muon_Px[2]/D");
muonTree->Branch("Muon_Py", Muon_Py, "Muon_Py[2]/D");
```

```
        if ( photons->GetEntries() != 0 ) {
        for (int j=0; j< photons->GetEntries(); j++) {
                auto p = static_cast<const Photon*>(photons->At(j));
                NPhoton = photons->GetEntries();

                Photon_PT[j] = p->PT;
                Photon_Eta[j] = p->Eta;
                Photon_Phi[j] = p->Phi;

                Photon_Px[j] = Photon_PT[j]*cos(Photon_Phi[j]);
                Photon_Py[j] = Photon_PT[j]*sin(Photon_Phi[j]);
                Photon_Pz[j] = Photon_PT[j]*sinh(Photon_Eta[j]);

                Photon_Energy[j] = p->E;

        }
        }
        else { NPhoton = 0; }
    muonTree->Fill();
    }
    muonTree->Write();
    newFile->Close();
```



Computation

- Charged particle path (x,y,z)

→ (x , y): circular path / z: linear path

1. Calculate R

Equation: ***Pt = q*R*B***

Pt = sqrt(Muon_Px[i]*Muon_Px[i] + Muon_Py[i]*Muon_Py[i])

q = muon charge = -e (+e for antimuon)

B = 3.8 T

Computation

- Charged particle path (x,y,z)

2. Scale

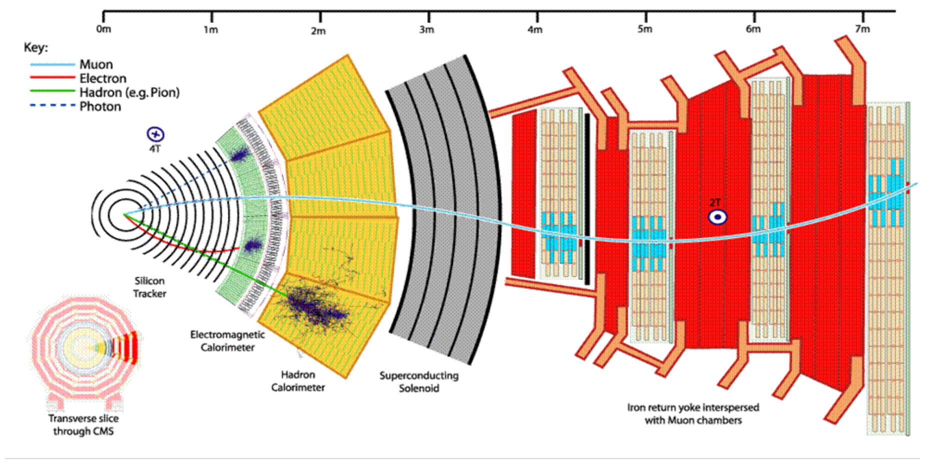
Ex) R = 12m

Superconducting solenoid: ~3.5m

→ in canvas scale: 400

Radius of circular path on x-y plane: ~12m

→ in canvas scale: 12*(400/3.5)

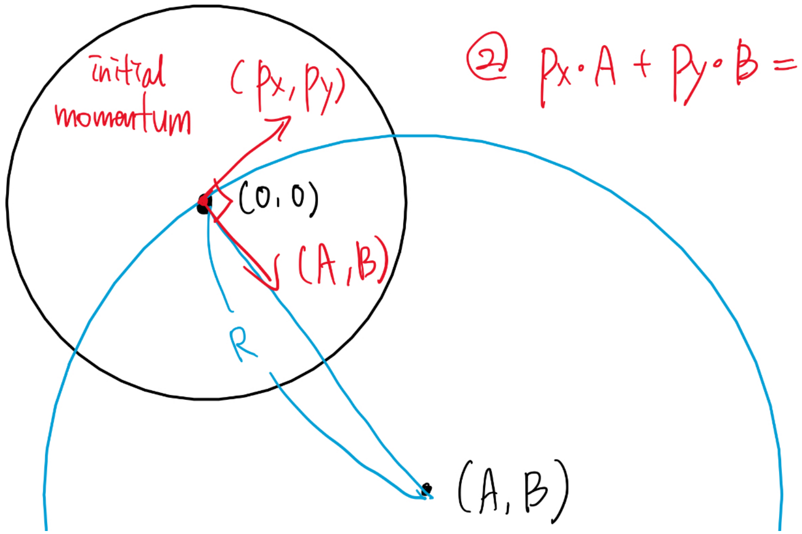


Computation

- Charged particle path (x,y,z)

3. Find the center (A,B) of the circular path

1) Calculate the absolute values of A and B



① $(A-0)^2+(B-0)^2=R^2$

② $P_x\cdot A+P_y\cdot B=0$

① $P_x A+P_y B=0$

$\Rightarrow B=-\frac{P_x}{P_y}A$

② $A^2+B^2=R^2$

$\Rightarrow A^2+\frac{P_x^2}{P_y^2}A^2=R^2$

$\rightarrow \left(1+\frac{P_x^2}{P_y^2}\right)A^2=R^2$

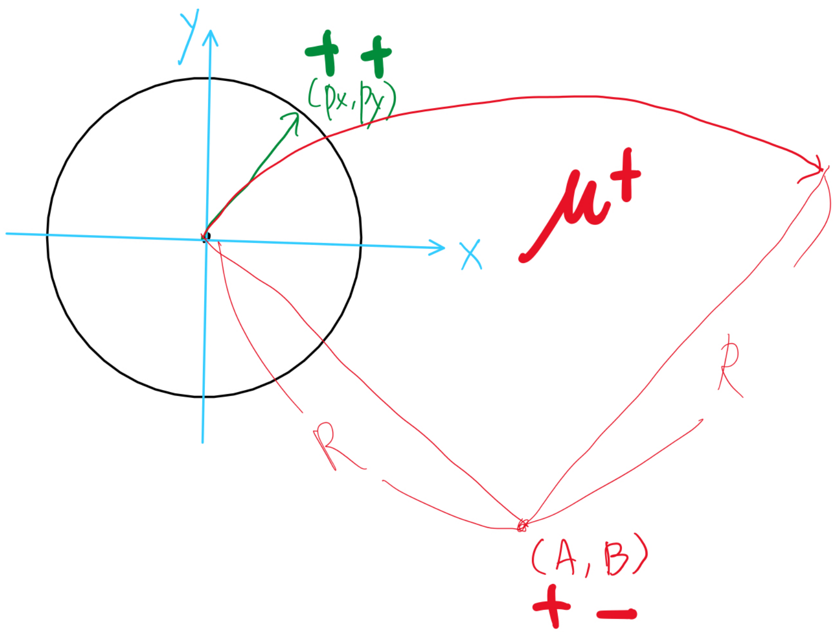
$$\Rightarrow \begin{cases} |A|=\sqrt{\dfrac{R^2}{1+\frac{P_x^2}{P_y^2}}} \\ |B|=\left|\dfrac{P_x}{P_y}\right||A| \end{cases}$$

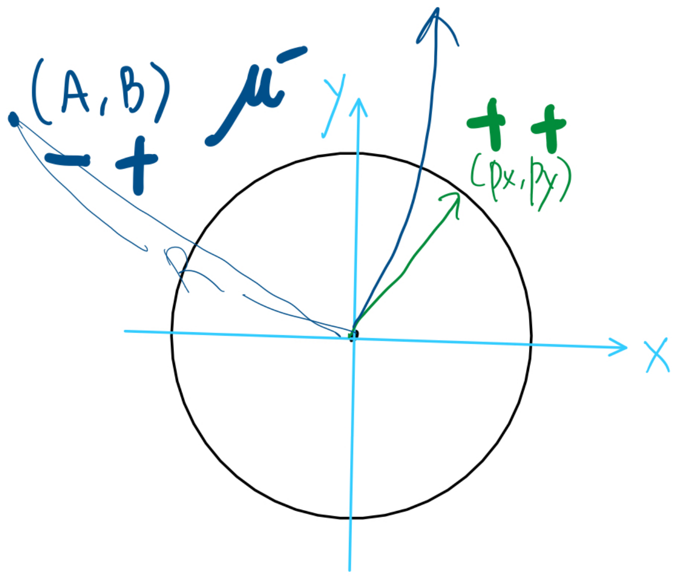
Computation

- Charged particle path (x,y,z)

3. Find the center (A,B) of the circular path

2) Calculate the signs of A and B





Computation

- Charged particle path (x,y,z)

3. Find the center (A,B) of the circular path

2) Calculate the signs of A and B

- $(px > 0, py > 0) \rightarrow$ Muon $(A < 0, B > 0)$, Antimuon $(A > 0, B < 0)$
- $(px > 0, py < 0) \rightarrow$ Muon $(A > 0, B > 0)$, Antimuon $(A < 0, B < 0)$
- $(px < 0, py > 0) \rightarrow$ Muon $(A < 0, B < 0)$, Antimuon $(A > 0, B > 0)$
- $(px < 0, py < 0) \rightarrow$ Muon $(A > 0, B < 0)$, Antimuon $(A < 0, B > 0)$

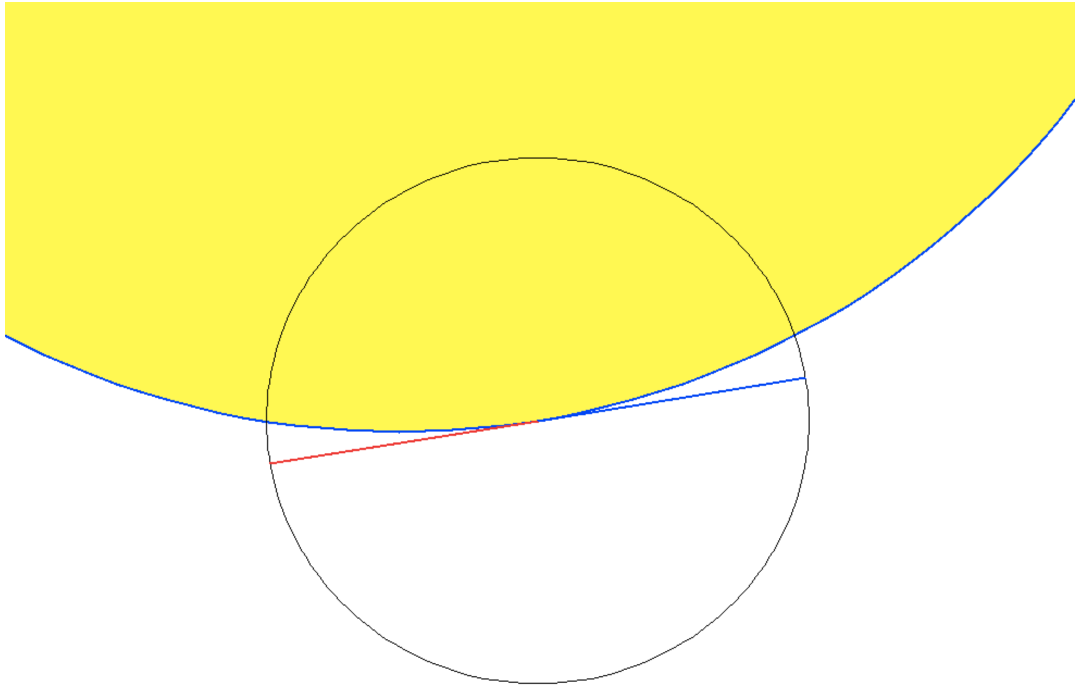
Computation

- Charged particle path (x,y,z)

4. Calculate and draw the 2D path

$\rightarrow (x-A)^2 + (y-B)^2 = R^2$

$$\begin{cases} y = B + sqrt(R^2-(x-A)^2) \\ y = B - sqrt(R^2-(x-A)^2) \end{cases}$$



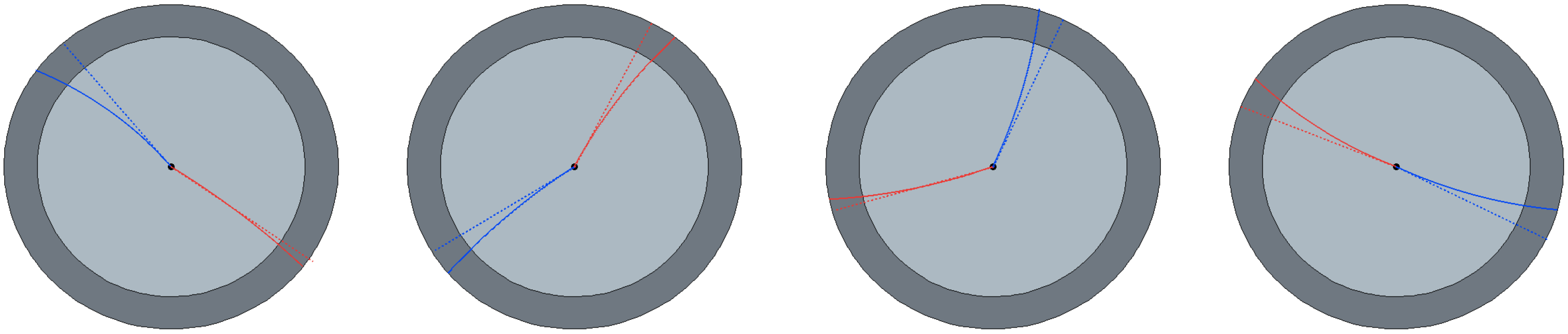
```
x += 1;
x -= 1;
```

Computation

- Charged particle path (x,y,z)

4. Calculate and draw the 2D path

Muon Antimuon ← Muon_Charge[i]



Computation

- Charged particle path (x,y,z)

5. Calculate Z direction linear path

1) Direction: Muon_Pz[i]

2) Scale

→ abs(dx) : abs(Muon_Px[i]) = abs(dz) : abs(Muon_Pz[i])

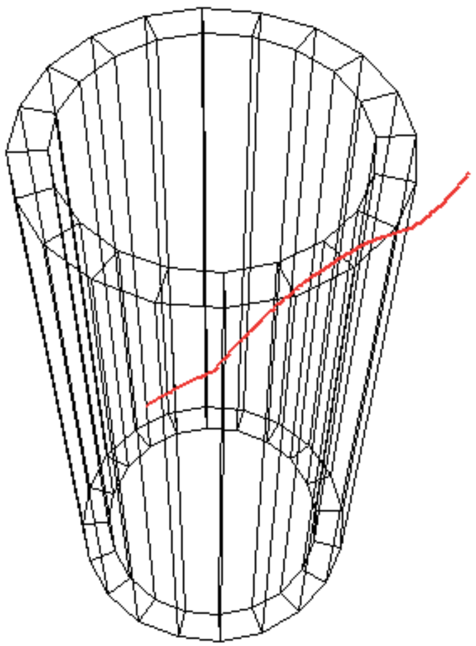
→ x += 1

→ z += Muon_Pz[i]*1/abs(Muon_Px[i])

Computation

- Charged particle path (x,y,z)

6. Draw helical paths of charged particles



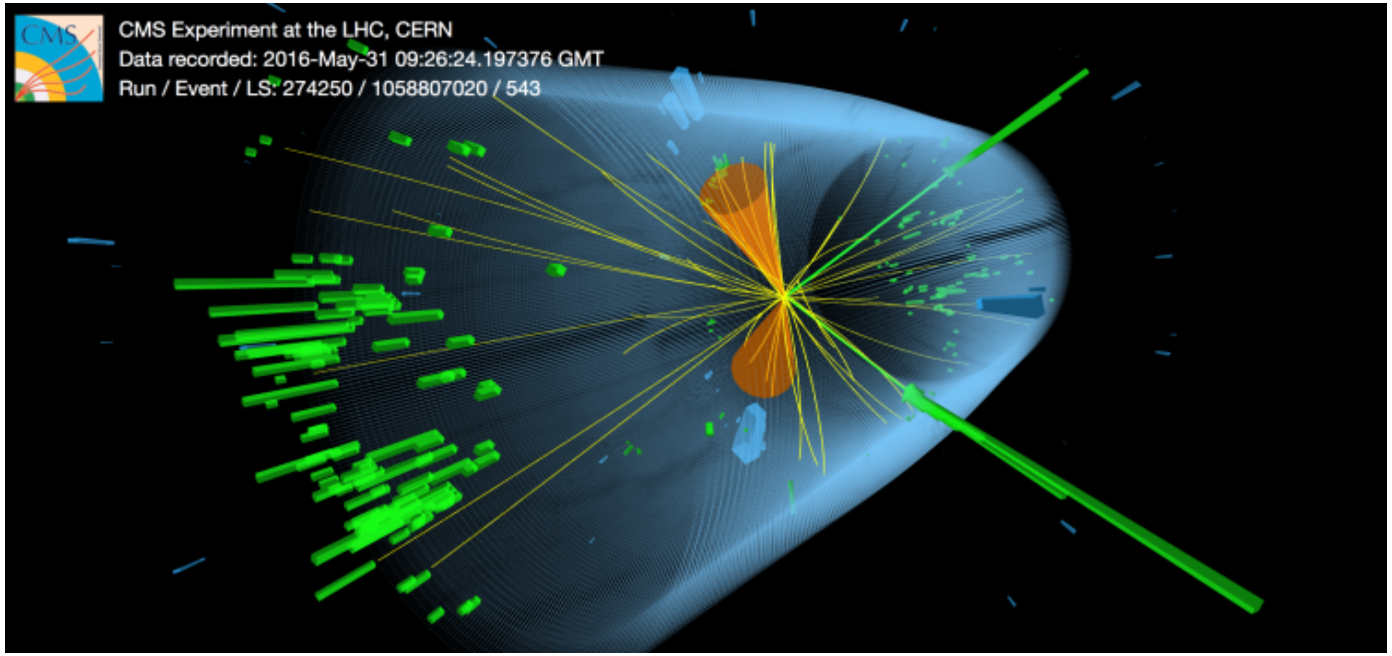
Computation

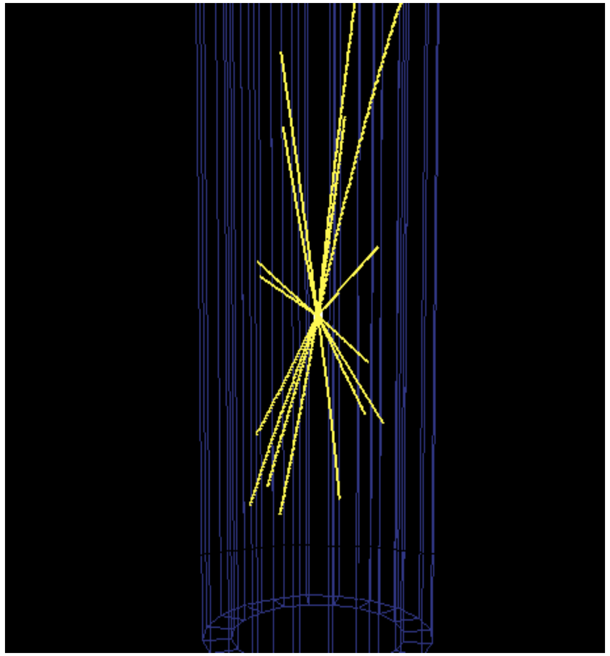
- Muons/Electrons: helical path
- (X,Y): circular path
- Z: linear path

- Photons: linear path
- (X,Y,Z)

Computation

- Protons (yellow lines)



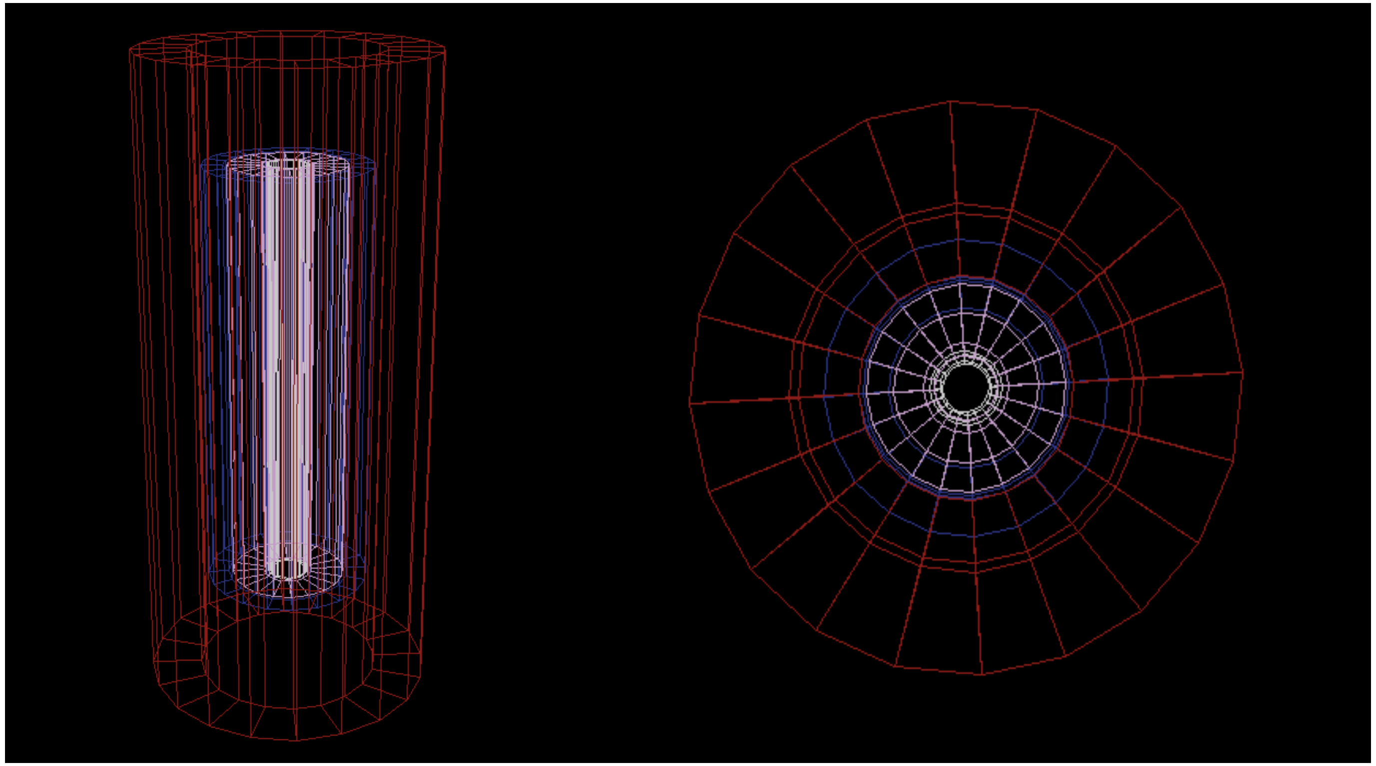


3D Visualization

- Detectors

- TGeometry class

```
geom = new TGeoManager("tube", "poza2");

// material and medium
matVacuum = new TGeoMaterial("Empty Space", 0, 0, 0);
medVacuum = new TGeoMedium("MED",1,matVacuum);

// top container
vol_top = geom->MakeBox("TOP", medVacuum, 1000, 1000, 1000);
geom->SetTopVolume(vol_top);

// superconducting coil container volume
vol_super = geom->MakeBox("SUPER",medVacuum,1000,1000,1000);
//geom->SetTopVolume(vol_super);

// name, material, rmin, rmax, dz
super = geom->MakeTube("superTUBE",medVacuum, 300,400,1000);
super->SetLineWidth(1);
super->SetLineColor(kGray+4);

// electromagnetic calorimeter container volume
vol_ecal = geom->MakeBox("ECAL", medVacuum, 90, 100, 1000);

ecal = geom->MakeTube("ecalTUBE", medVacuum, 90, 100, 1000);
ecal->SetLineWidth(1);
ecal->SetLineColor(kGray);
```



3D Visualization

- Particle trajectories

- TGeoTrack
- Add points and draw

```
// create track objects for particles
for (int j=0; j<NPROTON; j++) {
        protons[j] = new TGeoTrack(3,3);
}
for (int i=0; i<NMUON; i++) {
        muonTrack[i] = new TGeoTrack(1,13);
}

for (int i=0; i<NPHOTON; i++) {
        gammaTrack[i] = new TGeoTrack(1,13);
}

// if timer for z to ee event exists, delete it
if (eTimer != NULL) { delete eTimer; eTimer = NULL; }

// set timer for z to mumu event
muTimer = new TTimer(1);
muTimer->SetCommand("AnimateMuon()");
muTimer->TurnOn();
```

```
                // animate muon trajectories
                if ( draw_muons->IsDown() == true ) {
                        if (MuonX[0][t]*MuonX[0][t] + MuonY[0][t]*MuonY[0][t] <= muonChamber*muonChamber ||
MuonX[1][t]*MuonX[1][t] + MuonY[1][t]*MuonY[1][t] <= muonChamber*muonChamber ) {

                                for (int i=0; i<NMUON; i++) {
                                        muonTrack[i]->AddPoint(MuonX[i][t], MuonY[i][t], MuonZ[i][t], t);
                                        muonTrack[i]->SetLineColor(kBlue);
                                        muonTrack[i]->SetLineWidth(3);
                                        muonTrack[i]->Draw("SAME");
                                }
                        }

                        else { muTimer->TurnOff(); cout << "muTIMER OFF" << endl; }
                }
```

3D Visualization

- Moving protons before collision

- TNode

```
// draw moving protons (protons before collision)
parent = new TTUBE("PARENT", "SPHE", "void", 1000, 1000, 1000);
proton1 = new TSPHE("PROTONBUNCH1", "SPHE", "void", 0, 15, 0, 180, 0, 360);
proton2 = new TSPHE("PROTONBUNCH2", "SPHE", "void", 0, 15, 0, 180, 0, 360);

parentNode = new TNode("PARENTNODE", "PARENTNODE", "PARENT");
parentNode->cd();
proton_bunch1 = new TNode("NODE1", "NODE1", "PROTONBUNCH1");
proton_bunch2 = new TNode("NODE2", "NODE2", "PROTONBUNCH2");
proton_bunch1->SetPosition(0,0,2000);
proton_bunch2->SetPosition(0,0,-2000);
proton_bunch1->SetLineColor(kWhite);
proton_bunch2->SetLineColor(kWhite);
parentNode->Draw("SAME");
```

```
// animate proton bunches before collision
if ( proton_bunch1->GetZ() != 0 && proton_bunch2->GetZ() != 0 ) {
        proton_bunch1->SetPosition(0,0,2000-10*proton_step);
        proton_bunch2->SetPosition(0,0,-2000+10*proton_step);
        gPad->Modified();
        gPad->Update();
        proton_step += 1;
        cout << proton_step << endl;
}
```

GUI

- (SHOW/HIDE) Detector Layers
- (SHOW/HIDE) Particles
- (SELECT AND DRAW) Decay Channel
- Reset / Exit

Terminal Output

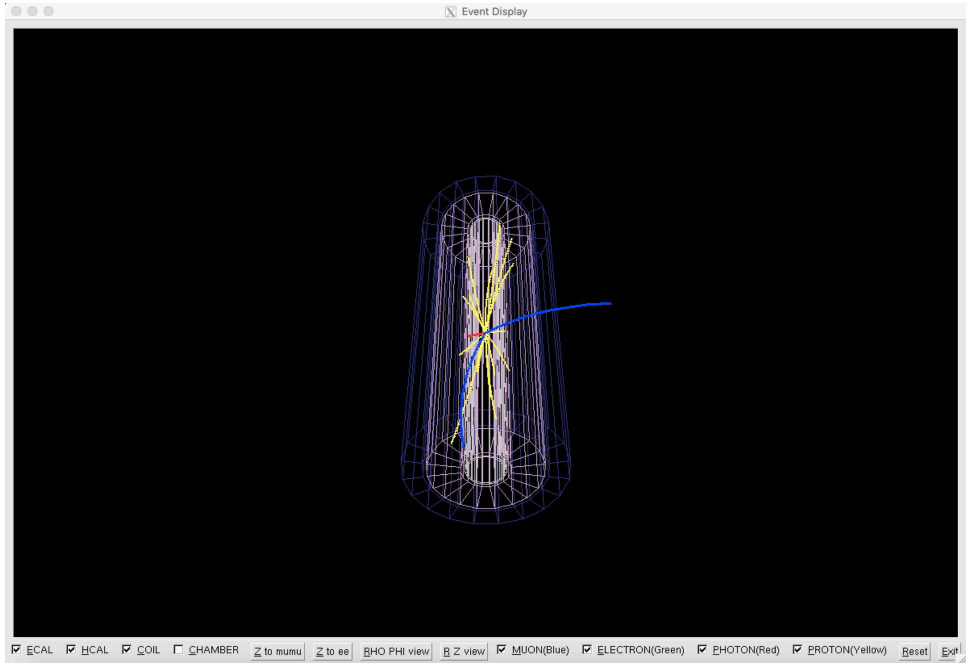
```
EVENT: 962 / 10000

MUON 1 is muon.
MUON 1 pT = 39.4245 GeV

MUON 2 is antimuon.
MUON 2 pT = 28.1268 GeV
```

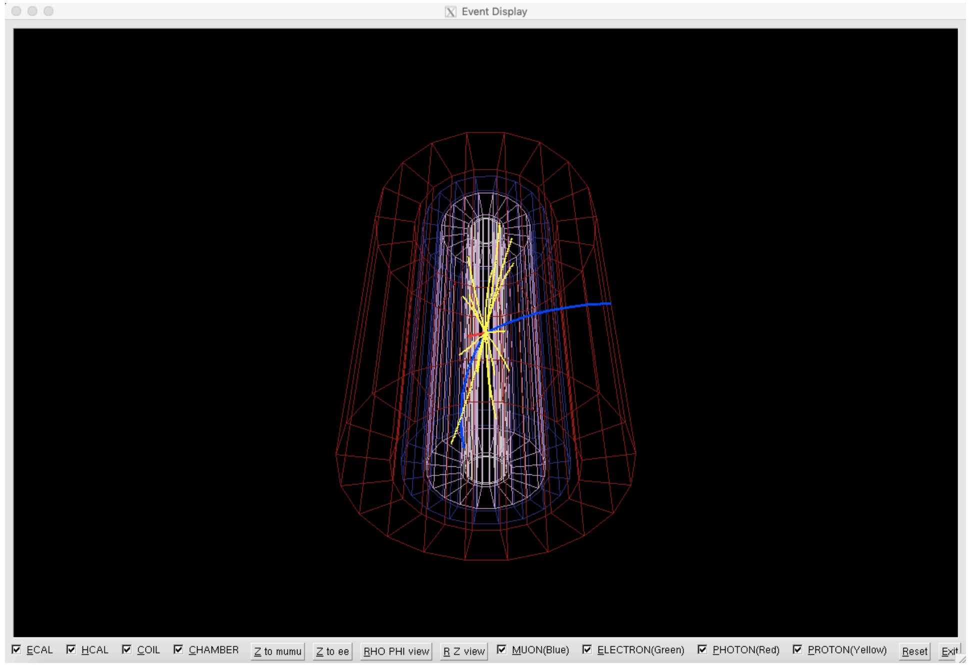
```
EVENT: 9 / 10000

ELECTRON 1 is electron.
ELECTRON 1 pT = 32.8 GeV

ELECTRON2 is positron.
ELECTRON 2 pT = 20.4178 GeV
```

Result

- Z to mumu

Muon / Photon / Proton





Plan

- Correct muon trajectories beyond the superconducting solenoid
- Calculate and draw jets
- Consider secondary+ vertices