```
#define Analyzer_cxx
#include "Analyzer.h"
#include <TH2.h>
#include <TGraph.h>
#include <TGraphAsymmErrors.h>
#include <TMultiGraph.h>
#include <TStyle.h>
#include <TCanvas.h>
#include <TPaveText.h>
#include <TLegend.h>
#include <iostream>
#include <TRandom.h>
#include <TMath.h>
#include <iostream>


void Analyzer::InitHisto(){
  layerDepthElectron = new TProfile("layerDepthElectron","Calo Layer Depth",250,
0,250,0,24);
  layerDepthProton = new TProfile("layerDepthProton","Calo Layer Depth",250,0,25
0,0,24);

  energyAVGRangeElectron = new TProfile("energyAVGRangeElectron","Energy ratio b
etween LYSO and SCINT(AVG)",300,0,300,0,15);
  energySUMRangeElectron = new TProfile("energySUMRangeElectron","Energy ratio b
etween LYSO and SCINT(SUM)",300,0,300,0,15);

  energyAVGRangeProton = new TProfile("energyAVGRangeProton","Energy ratio betwe
en LYSO and SCINT(AVG)",300,0,300,0,15);
  energySUMRangeProton = new TProfile("energySUMRangeProton","Energy ratio betwe
en LYSO and SCINT(SUM)",300,0,300,0,15);

  Ecinetique_gen = new TH1F("hecgen","Distribution Ek generated",1000,0,200);
  Ecinetique_acc_trk = new TH1F("hecacctrk","Distribution Ek accepteed trk",1000
,0,200);
  Ecinetique_acc_sci = new TH1F("hecaccaco","Distribution Ek accepteed sci",1000
,0,200);
  Ecinetique_acc_veto = new TH1F("hecaccvetp","Distribution Ek accepteed veto",1
000,0,200);

}

void Analyzer::CloseHisto(){

  TCanvas* c1f = new TCanvas("c1f","",200,10,700,500);
  gStyle->SetOptStat(0);
  layerDepthElectron->SetMarkerColor(kBlue);
  layerDepthElectron->SetMarkerStyle(22);
  layerDepthElectron->SetMarkerSize(2);
  layerDepthElectron->GetXaxis()->SetTitle("Energy [MeV]");
  layerDepthElectron->GetXaxis()->CenterTitle(true);
  layerDepthElectron->GetXaxis()->SetTitleSize(0.05);
  layerDepthElectron->GetXaxis()->SetTitleOffset(0.90);
  layerDepthElectron->GetXaxis()->SetLabelSize(0.05);
  layerDepthElectron->GetYaxis()->SetTitle("<#> Layer");
  layerDepthElectron->GetYaxis()->CenterTitle(true);
  layerDepthElectron->GetYaxis()->SetTitleSize(0.05);
  layerDepthElectron->GetYaxis()->SetTitleOffset(0.90);
  layerDepthElectron->GetYaxis()->SetLabelSize(0.05);

  layerDepthProton->SetMarkerColor(kRed);
  layerDepthProton->SetMarkerStyle(23);
  layerDepthProton->SetMarkerSize(2);
  layerDepthProton->GetXaxis()->SetTitle("Energy [MeV]");
  layerDepthProton->GetXaxis()->CenterTitle(true);
  layerDepthProton->GetXaxis()->SetTitleSize(0.05);
  layerDepthProton->GetXaxis()->SetTitleOffset(0.90);
  layerDepthProton->GetXaxis()->SetLabelSize(0.05);
  layerDepthProton->GetYaxis()->SetTitle("<#> Layer");
```

```
  layerDepthProton->GetYaxis()->CenterTitle(true);
  layerDepthProton->GetYaxis()->SetTitleSize(0.05);
  layerDepthProton->GetYaxis()->SetTitleOffset(0.90);
  layerDepthProton->GetYaxis()->SetLabelSize(0.05);

  if(layerDepthElectron->GetBinContent(layerDepthElectron->GetMaximumBin())>laye
rDepthProton->GetBinContent(layerDepthProton->GetMaximumBin())){
    layerDepthElectron->Draw();
    layerDepthProton->Draw("same");
  }else{
    layerDepthProton->Draw();
    layerDepthElectron->Draw("same");
  }

  //  c1f->SaveAs("LayerDepth.eps");

  TCanvas* c2f = new TCanvas("c2f","",200,10,700,500);
  gStyle->SetOptStat(0);

  energyAVGRangeElectron->SetMarkerColor(kBlue);
  energyAVGRangeElectron->SetMarkerStyle(22);
  energyAVGRangeElectron->SetMarkerSize(1);
  energyAVGRangeElectron->SetTitle("");
  energyAVGRangeElectron->GetXaxis()->SetTitle("Energy [MeV]");
  energyAVGRangeElectron->GetYaxis()->SetTitle("E_{Rec LYSO}/<E_{Rec SCINT}>");
  energyAVGRangeElectron->SetLineColor(kBlue);

  energyAVGRangeElectron->Draw("BOX");

  //  c2f->SaveAs("Figure/TestE_AVG.eps");

  TCanvas* c3f = new TCanvas("c3f","",200,10,700,500);
  gStyle->SetOptStat(0);
  energySUMRangeElectron->SetMarkerColor(kBlue);
  energySUMRangeElectron->SetMarkerStyle(22);
  energyAVGRangeElectron->SetMarkerSize(1);
  energySUMRangeElectron->SetTitle("");
  energySUMRangeElectron->GetXaxis()->SetTitle("Energy [MeV]");
  energySUMRangeElectron->GetYaxis()->SetTitle("E_{Rec LYSO}/#Sigma E_{Rec SCINT
}");
  energySUMRangeElectron->SetLineColor(kBlue);

  energySUMRangeElectron->Draw("BOX");

  //  c3f->SaveAs("Figure/TestE_SUM.eps");

  TCanvas* c4f = new TCanvas("c4f","",200,10,700,500);
  gStyle->SetOptStat(0);
  energyAVGRangeProton->SetMarkerColor(kRed);
  energyAVGRangeProton->SetMarkerStyle(22);
  energyAVGRangeProton->SetMarkerSize(1);
  energyAVGRangeProton->SetTitle("");
  energyAVGRangeProton->GetXaxis()->SetTitle("Energy [MeV]");
  energyAVGRangeProton->GetYaxis()->SetTitle("E_{Rec LYSO}/<E_{Rec SCINT}>");
  energyAVGRangeProton->SetLineColor(kRed);

  energyAVGRangeProton->Draw("BOX");

  //  c4f->SaveAs("Figure/TestP_AVG.eps");

  TCanvas* c5f = new TCanvas("c5f","",200,10,700,500);
  gStyle->SetOptStat(0);
  energySUMRangeProton->SetMarkerColor(kRed);
  energySUMRangeProton->SetMarkerStyle(22);
  energySUMRangeProton->SetMarkerSize(1);
  energySUMRangeProton->SetTitle("");
  energySUMRangeProton->GetXaxis()->SetTitle("Energy [MeV]");
  energySUMRangeProton->GetYaxis()->SetTitle("E_{Rec LYSO}/#Sigma E_{Rec SCINT}"
);
```

```
  energySUMRangeProton->SetLineColor(kRed);

  energySUMRangeProton->Draw("BOX");

  gStyle->SetOptTitle(0);
  gStyle->SetOptStat(0);
  gStyle->SetCanvasColor(10);
  gStyle->SetPadColor(10);
  gStyle->SetPalette(1,0);
  gStyle->SetFrameFillColor(kWhite);

  char cpad[80];
  TPad* ptpad;

  TCanvas* c1 = new TCanvas("c1","ec gen electron",0,0,500,500);
  c1->SetFillColor(0);
  c1->Divide(2,2,0.001,0.001);
  for (int i=0; i<4; i++) {
    sprintf(cpad,"c1_%d",i+1);
    ptpad = (TPad*) c1->FindObject(cpad);
    ptpad->SetFillColor(10);
    ptpad->SetLogy(1);
    ptpad->SetLogx(0);
    ptpad->SetLeftMargin(0.15);
    ptpad->SetBottomMargin(0.15);
    ptpad->SetRightMargin(0.125);
    ptpad->SetTopMargin(0.125);
    c1->cd(i+1);
    switch(i) {
    case 0: Ecinetique_gen->Draw(); break;
    case 1: Ecinetique_acc_trk->Draw(); break;
    case 2: Ecinetique_acc_sci->Draw(); break;
    case 3: Ecinetique_acc_veto->Draw(); break;
    default: break;
    }
  }


}

void Analyzer::Loop()
{

  //  double Energy = E;
  int bufsize = 80000000;

  printf("get entries\n");
  Long64_t nentries = fTreeChain->GetEntriesFast();
  printf("get entries\n");
  Long64_t nbytes = 0, nb = 0;
  Double_t maxLayer = 0;

  printf("nentries %d\n",nentries);
  //  for (Long64_t jentry=0; jentry<50000;jentry++) {
  for (Long64_t jentry=0; jentry<nentries;jentry++) {

    bool s1Hit=false;
    bool s2Hit=false;
    bool noVetoHit=false;
    bool hitOnTkLayer1=false;
    bool hitOnTkLayer2=false;
    bool goodHit=false;
    maxLayer=0;
    Long64_t ientry = LoadTree(jentry);
    if (ientry < 0) break;
    nb = fTreeChain->GetEntry(jentry);   nbytes += nb;
    std::vector<RootTrack> myTracks = Event->GetTracks();
    std::vector<RootCaloHit> myCaloHit = Event->GetCaloHit();
    std::vector<RootCaloHit> myVetoHit = Event->GetVetoHit();
```

```
    std::vector<RootTrackerHit> myTrackerHit = Event->GetTrackerHit();

    TVector3 electronDir = myTracks[0].GetDirection();
    TVector3 position = myTracks[0].GetPosition();
    float theta = electronDir.Theta()*180/TMath::Pi();
    if(theta>90)
      theta=180-theta;
    float ce = myTracks[0].GetKinEnergy();
    Ecinetique_gen->Fill(ce);
        if(myTrackerHit.size()>0){
          //          printf("trk no hits %d\n",myTrackerHit.size());
          for(size_t th=0;th<myTrackerHit.size();th++){
            int detId = myTrackerHit[th].GetDetectorId();
            int tkid = myTrackerHit[th].GetTrackId();
            //            printf("tk hit %d det id %d\n",th,detId);
            if(detId>2200 && tkid==1) {
              hitOnTkLayer1=true;
              //              printf("hit layer 1\n",th);
            }
            else if(detId>2100 && tkid==1) {
              hitOnTkLayer2=true;
              //              printf("hit layer 2\n",th);
            }
          }
          if(hitOnTkLayer2&&hitOnTkLayer1) {
                   Ecinetique_acc_trk->Fill(ce);
                   printf("no tracker hits 2\n",ce);
          //          printf("\n");
          }
        }
    hitOnTkLayer1=false;
    hitOnTkLayer2=false;
    if(myCaloHit.size()>0){
      if (ce > 10) {
        //         printf("pdg %d x %f y %f z %f\n",myTracks[0].GetPDG(),positio
n.X(),position.Y(),position.Z());
        //         printf("ientry %d energie %10.3e\n",ientry,ce);
      }
      //    printf("Calo hit size %d\n",myCaloHit.size());
      for(size_t sh=0;sh<myCaloHit.size();sh++){
        int scintLayer = myCaloHit[sh].GetVolume();
        //        printf("scintlayer %d\n",scintLayer);
        std::vector<int> plist = myCaloHit[sh].GetParticleList();
        //f or(size_t p=0;p<plist.size();p++)
        //        printf("pid %d PDG %d\n",plist[p],myTracks[plist[p]].GetPDG())
;
        //      if(scintLayer.Contains("S1"))
        if (myCaloHit.size()>=2 && ce <= 15)
        printf("scintLayer %d\n",scintLayer);
        if (scintLayer >= 1300 && plist[0]==1) s1Hit=true;
        if (scintLayer == 1216 && plist[0]==1) s2Hit=true;
      }
      printf("\n");
      if(s1Hit&&s2Hit){
        Ecinetique_acc_sci->Fill(ce);
        if(myVetoHit.size()==0){
          noVetoHit=true;
          Ecinetique_acc_veto->Fill(ce);
        }
        if(myTrackerHit.size()>0&&noVetoHit){
          for(size_t th=0;th<myTrackerHit.size();th++){
            int detId = myTrackerHit[th].GetDetectorId();
            int tkid = myTrackerHit[th].GetTrackId();
            if(detId>2200 && tkid==1) {
              hitOnTkLayer1=true;
              //              printf("hit layer 1\n",th);
            }
            else if(detId>2100 && tkid==1) {
              hitOnTkLayer2=true;
              //              printf("hit layer 2\n",th);
            }
          }
          if(hitOnTkLayer2&&hitOnTkLayer1)
            goodHit=true;
        }
      }
    }

    /*    if(checkPos&&goodHit){
      for(size_t j=0;j<myTracks.size();j++){
        if(myTracks[j].GetTrackID()==1){
          Double_t Xpos =std::fabs(myTracks[j].GetPosition().X());
          Double_t Ypos =std::fabs(myTracks[j].GetPosition().Y());
          if(Xpos>Xlimit&&Ypos>Ylimit)
            goodHit=false;
        }
      }
    }
    if(checkTheta&&goodHit){
      Double_t myThetaWithSmearing = ComputeAngleWithSmearing(myTrackerHit,0.05)
*180/TMath::Pi();
      if(myThetaWithSmearing>Thetalimit)
        goodHit=false;
        } */

    /*    if(goodHit){
      Double_t myTheta = ComputeAngle(myTrackerHit)*180/TMath::Pi();
      Double_t myThetaWithSmearing = ComputeAngleWithSmearing(myTrackerHit,0.05)
*180/TMath::Pi();

      Double_t trackTheta=0;
      for(size_t j=0;j<myTracks.size();j++){
        if(myTracks[j].GetTrackID()==1)
          trackTheta=myTracks[j].GetDirection().Theta()*180/TMath::Pi();
      }
      if(trackTheta>90)
        trackTheta=180-trackTheta;

      float totalEdep   = 0;
      float totalEnoS1  = 0;
      float scintS1Edep = 0;
      float scintS2Edep = 0;
      float scintEdep   = 0;
      float caloEdep    = 0;
      float siliconEdep = 0;
      for(size_t i=0;i<myTrackerHit.size();i++){
        siliconEdep+=myTrackerHit[i].GetELoss();
        totalEdep+=myTrackerHit[i].GetELoss();
        totalEnoS1+=myTrackerHit[i].GetELoss();
        int detId = myTrackerHit[i].GetDetectorId();
        /*      if(detId>200)
          layerDepElectron->Fill(1,myTrackerHit[i].GetELoss());
        if(detId<200)
        layerDepElectron->Fill(2,myTrackerHit[i].GetELoss());  */
    //      }
      //      siliconEdepHisto->Fill(siliconEdep);
  /*      for(size_t i=0;i<myCaloHit.size();i++){
        scintEdep+=myCaloHit[i].GetTotalEdep();
        totalEdep+=myCaloHit[i].GetTotalEdep();
        TString volume = myCaloHit[i].GetVolume();
        if(volume.Contains("S1")){
          scintS1Edep+=myCaloHit[i].GetTotalEdep();
          //      layerDepElectron->Fill(3,myCaloHit[i].GetTotalEdep());
        }
        else if(volume.Contains("S2")){
          scintS2Edep+=myCaloHit[i].GetTotalEdep();
          totalEnoS1+=myCaloHit[i].GetTotalEdep();
          //      layerDepElectron->Fill(4,myCaloHit[i].GetTotalEdep());
```

```
        }
      }
      //      scintS1EdepHisto->Fill(scintS1Edep);
      //      scintS2EdepHisto->Fill(scintS2Edep);
      //      scintEdepHisto->Fill(scintEdep);
      if(myCaloHit.size()){
        float E_Rec_Scint=0;
        float E_Rec_LYSO=0;
        int hittedBricks=0;
        float eDep = 0;
        float bricksEDep[9];
        for(int i =0;i<9;i++)
          bricksEDep[i] = 0;
        for(size_t i=0;i<myCaloHit.size();i++){
          eDep = myCaloHit[i].GetTotalEdep();;
          caloEdep+=eDep;
          totalEdep+=eDep;
          totalEnoS1+=eDep;
          TString layer = myCaloHit[i].GetVolume();
          int layernumb;
          if(numbLayerCrystal==-1&&layer.Contains("ActiveBlockCrystal")){
            layernumb = numbLayerScint+1;
            hittedBricks++;
            layer.Remove(0,18);
            bricksEDep[layer.Atoi()]+=eDep;
          }else if(layer.Contains("ActiveLayerCrystal")){
            layer.Remove(0,18);
            layernumb = numbLayerScint+numbLayerCrystal-layer.Atoi();
          }else if(layer.Contains("ActiveLayerScint")){
            layer.Remove(0,16);
            layernumb = numbLayerScint-layer.Atoi();
          }else{
            layer.Remove(0,11);
            layernumb = numbCaloLayer-layer.Atoi();
          }
          if(maxLayer<layernumb)
            maxLayer=layernumb;
          //      layerDepElectron->Fill(layernumb+4,eDep);
          if(layernumb== numbLayerScint+1)
            E_Rec_LYSO+=eDep;
          else
            E_Rec_Scint+=eDep;
        }
        float lysoDep=0;
        for(int i = 0;i<9;i++){
          if(bricksEDep[i]!=0){
            lysoDep+=bricksEDep[i];
            //      singlebrickDep->Fill(bricksEDep[i]);
          }
          //      if(lysoDep!=0&&i==8)
          //  allbricksDep->Fill(lysoDep);
        }
        //      bricksHitted->Fill(hittedBricks);
        //              caloEdepHisto->Fill(caloEdep);
        //      layerDepthElectron->Fill(theERange,maxLayer);
        //      energyAVGRangeElectron->Fill(theERange,E_Rec_LYSO/(E_Rec_Scint/n
umbLayerScint));
        //      energySUMRangeElectron->Fill(theERange,E_Rec_LYSO/E_Rec_Scint);
      }

      //      totalEdepHistoNoS1NoCut->Fill(totalEnoS1);
      //      totalEdepHistoNoCut->Fill(totalEdep);

      /*      if(totalEnoS1>2.5)
        totalEdepHistoNoS1->Fill(totalEnoS1);
      if(totalEnoS1>1.5)
        totalEdepHistoNoS1Cut2->Fill(totalEnoS1);
      if(totalEdep>=2.5){
        totalEdepHisto->Fill(totalEdep);
        energyVSangle->Fill(myThetaWithSmearing,totalEdep/theERange);
        }
    } */

  }
}

void Analyzer::SetAcceptanceWindows(Double_t X,Double_t Y){
  checkPos=true;
  Xlimit=X/2.;
  Ylimit=Y/2.;
}

void Analyzer::SetThetaAcceptance(Double_t aTheta){
  checkTheta=true;
  Thetalimit=aTheta;
}

Double_t Analyzer::ComputeAngle(std::vector<RootTrackerHit>& myTrackerHit){

  std::vector<RootTrackerHit> layer2Hit;
  std::vector<RootTrackerHit> layer1Hit;

  for(size_t th=0;th<myTrackerHit.size();th++){
    int detId = myTrackerHit[th].GetDetectorId();
    if(detId>200)
      layer2Hit.push_back(myTrackerHit[th]);
    if(detId<200)
      layer1Hit.push_back(myTrackerHit[th]);
  }

  TVector3 posL2;
  TVector3 posL1;

  if(layer2Hit.size()==1){
    posL2 = (layer2Hit[0].GetEntryPoint()+layer2Hit[0].GetExitPoint())*0.5;
  }else{
    for(size_t j= 0;j<layer2Hit.size();j++){
      if(layer2Hit[j].GetTrackId()==1)
        posL2 = (layer2Hit[j].GetEntryPoint()+layer2Hit[j].GetExitPoint())*0.5;
    }
  }
  if(layer1Hit.size()==1){
    posL1 = (layer1Hit[0].GetEntryPoint()+layer1Hit[0].GetExitPoint())*0.5;
  }else{
    for(size_t j= 0;j<layer2Hit.size();j++){
      if(layer1Hit[j].GetTrackId()==1)
        posL1 = (layer1Hit[j].GetEntryPoint()+layer1Hit[j].GetExitPoint())*0.5;
    }
  }

  TVector3 segment = posL2-posL1;

  return segment.Theta();
}

Double_t Analyzer::ComputeAngleWithSmearing(std::vector<RootTrackerHit>& myTrack
erHit,Double_t delta){

  std::vector<RootTrackerHit> layer2Hit;
  std::vector<RootTrackerHit> layer1Hit;

  for(size_t th=0;th<myTrackerHit.size();th++){
    int detId = myTrackerHit[th].GetDetectorId();
    if(detId>200)
      layer2Hit.push_back(myTrackerHit[th]);
    if(detId<200)
      layer1Hit.push_back(myTrackerHit[th]);
  }
```

```

  TVector3 posL2;
  TVector3 posL1;

  if(layer2Hit.size()==1){
    posL2 = (layer2Hit[0].GetEntryPoint()+layer2Hit[0].GetExitPoint())*0.5;
  }else{
    for(size_t j= 0;j<layer2Hit.size();j++){
      if(layer2Hit[j].GetTrackId()==1)
        posL2 = (layer2Hit[j].GetEntryPoint()+layer2Hit[j].GetExitPoint())*0.5;
    }
  }
  if(layer1Hit.size()==1){
    posL1 = (layer1Hit[0].GetEntryPoint()+layer1Hit[0].GetExitPoint())*0.5;
  }else{
    for(size_t j= 0;j<layer2Hit.size();j++){
      if(layer1Hit[j].GetTrackId()==1)
        posL1 = (layer1Hit[j].GetEntryPoint()+layer1Hit[j].GetExitPoint())*0.5;
    }
  }

  TRandom myRand;

  Double_t xL1 = posL1.X()+myRand.Uniform(-delta,delta);
  Double_t yL1 = posL1.Y()+myRand.Uniform(-delta,delta);

  posL1.SetX(xL1);
  posL1.SetY(yL1);

  Double_t xL2 = posL2.X()+myRand.Uniform(-delta,delta);
  Double_t yL2 = posL2.Y()+myRand.Uniform(-delta,delta);

  posL2.SetX(xL2);
  posL2.SetY(yL2);

  TVector3 segment = posL2-posL1;

  return segment.Theta();
}
```