Global Reconstruction

Foot Global Parameters

Run Information Writing

TOE Global Action

TOE Executable

GenFit Global Action

GenFit Executable

SubFolders

Conclusions

Foot Global Parameters (i)

- File: FootGlobal.par

```
. . .
IncludeKalman:   n
IncludeTOE:      n
EnableLocalReco: n

Kalman Mode:        ON
Tracking Systems Considered:        VT IT MSD
Reverse Tracking:   false

. . .

############   Options for reconstruction   ################

EnableTree:          y
EnableHisto:         y
EnableTracking:      y

EnableSaveHits:      n
EnableRootObject:    n
EnableTofZmc:        n
EnableTofCalBar:     n

. . .
```

Enable Glb Reco from level0 trees

Reconstruction options

➡ Add reconstruction options that were as argument in executables

Foot Global Parameters (ii)

- GlobalPar: (i)

```
// singleton class of global foot parameters
class GlobalPar {
. . .
   string  m_outputntuplename;
   bool m_printoutntuple;

   Bool_t m_enableLocalReco;
   Bool_t m_enableTree;
   Bool_t m_enableHisto;
   Bool_t m_enableSaveHits;
   Bool_t m_enableTracking;
   Bool_t m_enableRootObject;
   Bool_t m_enableTofZmc;
   Bool_t m_enableTofCalBar;

   bool m_includeST;
   bool m_includeBM;
   bool m_includeTG;
. . .
};
```

➡ Add reconstruction options as members of the class (with Getter and Setter)

Foot Global Parameters (iii)

- GlobalPar: (ii)

```
GlobalPar::GlobalPar( const TString expName )
{
    TString absName = Form("./config/%s/%s", expName.Data(), m_defParName.Data());
    m_parFileName = absName.Data();

    m_copyInputFile.clear();
    m_debug = 0;

    m_kalmanMode = -1;

    m_kalReverse = false;
    m_verFLUKA = false;

    ReadParamFile();
}
```

➡ Add experiment name in the constructor

➡ FootGlobal.par file located in each sub-folder experiment now

Foot Global Parameters (iv)

- TAGrunInfo (i):

```
struct GlobalParameter_t : public  TObject {
 . . .
   Bool_t EnableLocalReco;
   Bool_t EnableTree;
   Bool_t EnableHisto;
   Bool_t EnableSaveHits;
   Bool_t EnableTracking;
   Bool_t EnableRootObject;
   Bool_t EnableTofZmc;
   Bool_t EnableTofCalBar;
   Bool_t IncludeKalman;
   Bool_t IncludeTOE;
   Bool_t IncludeDI;
   Bool_t IncludeST;
   Bool_t IncludeBM;
   Bool_t IncludeTG;
   Bool_t IncludeVT;
   Bool_t IncludeIT;
   Bool_t IncludeMSD;
   Bool_t IncludeTW;
   Bool_t IncludeCA;
   ClassDef(GlobalParameter_t,1)
};

class TAGrunInfo : public TAGobject {
private:
   TString            fsCam;        // campaign name
   Short_t            fiRun;        // run number
   GlobalParameter_t  fGlobalParameter; // global parameters
. . .
};
```

➡ Add all reconstruction parameters in run info

Foot Global Parameters (v)

- TAGrunInfo : printout (ii)

```
KEY: TAGrunInfo runinfo;1
root [3] runinfo->Print()
Run info:       cam: 12C_200/  run:    1
Global info:
  EnableLocalReco: 0
  EnableTree: 1
  EnableHisto: 1
  EnableTracking: 1
  EnableSaveHits: 0
  EnableRootObject: 0
  EnableTofZmc: 0
  EnableTofCalBar: 0

  IncludeKalman: 0
  IncludeTOE: 1

  IncludeDI: 1
  IncludeST: 1
  IncludeBM: 1
  IncludeTG: 1
  IncludeVT: 1
  IncludeIT: 1
  IncludeMSD: 1
  IncludeTW: 1
  IncludeCA: 0
```

➡ Example of run information in output root file

Run Information Writing (i)

- TAGactTreeReader(Writer):

```
//! Open root file.
Int_t TAGactTreeReader::Open(const TString& name, Option_t* option, const TString treeName)
{
. . .
  TAGrunInfo* p_ri = (TAGrunInfo*) fpFile->Get(TAGrunInfo::GetObjectName());

  if (p_ri  && p_ri->InheritsFrom("TAGrunInfo")) {
    gTAGroot->SetRunInfo(*p_ri);
  } else {
    Warning("Open()", "No object named 'runinfo' found");
  }
}
```

```
//! Close file.
void TAGactTreeWriter::Close()
{
  TDirectory* p_cwd = gDirectory;

  if (fpFile && fpTree) {
    fpFile->cd();
    fpTree->Write();
    TAGrunInfo runinfo(gTAGroot->CurrentRunInfo());
    runinfo.Write(TAGrunInfo::GetObjectName());
  }
. . .
}
```

➡ Already in framework, run info automatically read and write in root file

Run Information Writing (ii)

- BaseReco:

```
//_____________________________________________________________
BaseReco::BaseReco(TString expName, Int_t runNumber, TString fileNameIn, TString fileNameout)
. . .
{
. . .
   // activate debug level
   GlobalPar::GetPar()->SetDebugLevels();

   // Save run info
   TAGrunInfo info = GlobalPar::GetPar()->GetGlobalInfo();
   info.SetCampaignName(fExpName);
   info.SetRunNumber(fRunNumber);
   gTAGroot->SetRunInfo(info);
. . .
}
```

➡ Add explicitly the run number in constructor

➡ Assigning run info to TAGroot

Run Information Writing (iii)

- G4 Simulation:

```
//....ooo00000ooo........ooo00000ooo........ooo00000ooo........ooo00000ooo......
void TCFOrunAction::SetContainers()
{
   //File for
   fpOutFile = new TFile(GetRootFileName(), "RECREATE");
   if (!fpOutFile) return;
   fpOutFile->cd();

   Int_t run          = gTAGroot->CurrentRunNumber();
   const Char_t* name = gTAGroot->CurrentCampaignName();

   TAGrunInfo info = GlobalPar::GetPar()->GetGlobalInfo();
   info.SetCampaignName(name);
   info.SetRunNumber(run);
   gTAGroot->SetRunInfo(info);
   info.Write(TAGrunInfo::GetObjectName());

   fpTree      = new TTree("EventTree", "FOOT");
    if(fkEvento){
        fpEventoMC = new Evento();
        fpEventoMC->SetBranches(fpTree);
    }
    else{
        fpEventMC = new TAMCevent();
        fpEventMC->SetBranches(fpTree);
    }
}
```

➡ Assigning run info to TAGroot in run action of Geant4 simulation

Run Information Writing (iv)

- Fluka simulation: Txt2(Ntu)Root.cxx

```
int main(int argc, char *argv[])
{
. . .
    if(strcmp(argv[i],"-exp") == 0) { exp = TString(argv[++i]); }   // extention for config/geomap files
    if(strcmp(argv[i],"-run") == 0) { runNb = atoi(argv[++i]);  }   // Run Number

. . .
  GlobalPar::Instance(exp);
  GlobalPar::GetPar()->Print();

  TFile *f_out = new TFile(outname,"RECREATE");
  f_out->cd();

  TAGrunInfo info = GlobalPar::Instance()->GetGlobalInfo();
  info.SetCampaignName(exp);
  info.SetRunNumber(runNb);
  info.Write(TAGrunInfo::GetObjectName());
. . .
}
```

➡ Add the run number and campaign as arguments of executable

➡ Reading of FootGlobal.par and save run info

TOE Executable (i)

- DecodeGlbToe:

```
int main(int argc, char *argv[])
{
. . .
   Bool_t lrc = GlobalPar::GetPar()->IsLocalReco();
   Bool_t ntu = GlobalPar::GetPar()->IsSaveTree();
   Bool_t his = GlobalPar::GetPar()->IsSaveHisto();
   Bool_t hit = GlobalPar::GetPar()->IsSaveHits();
   Bool_t trk = GlobalPar::GetPar()->IsTracking();
   Bool_t obj = GlobalPar::GetPar()->IsReadRootObj();
   Bool_t zmc = GlobalPar::GetPar()->IsTofZmc();
   Bool_t tbc = GlobalPar::GetPar()->IsTofCalBar();

   GlobalPar::GetPar()->IncludeTOE(true);
   GlobalPar::GetPar()->IncludeKalman(false);
   BaseReco* glbRec = 0x0;
   if (lrc)
      glbRec = new GlobalToeReco(exp, runNb, in, out, mc);
   else if (mc) {
      if (!obj)
         glbRec = new LocalRecoMC(exp, runNb, in, out);
      else
         glbRec = new LocalRecoNtuMC(exp, runNb, in, out);
      if(zmc)
         glbRec->EnableZfromMCtrue();
   }  else {
      glbRec = new LocalReco(exp, runNb, in, out);
      if (tbc)
         glbRec->EnableTWcalibPerBar();
   }
. . .
}
```

➡ All arguments retrieved from GlobalPar, Enable TOE (disable Kalman)

➡ Possibility to run global reco from hits (MC or Raw) or from local reco tree

TOE Executable (ii)

- Command line:

```
. . .
IncludeKalman:   n
IncludeTOE:      y
EnableLocalReco: n

Kalman Mode:        ON
Tracking Systems Considered:        VT IT MSD
Reverse Tracking:   false
. . .
```

Will be put off anyhow by executable

Will be put on anyhow by executable

➡ from MC hits: DecodeGlbToe -in **12C_C_200_1.root**
-out 12C_C_200_1_GlbOut.root -nev 10000 -exp 12C_200 -run 1 -mc

```
. . .
IncludeKalman:   n
IncludeTOE:      y
EnableLocalReco: y

Kalman Mode:        ON
Tracking Systems Considered:        VT IT MSD
Reverse Tracking:   false
. . .
```

➡ from L0 reco: DecodeGlbToe -in **12C_C_200_L0Out.root**
-out 12C_C_200_1_GlbOut.root -nev 10000 -exp 12C_200 -run 1 -mc
(not pushed, cos not done in a efficient way, Alessio is working on)

TOE Executable (iii)

- Command line:

```
. . .
IncludeKalman:   y
IncludeTOE:      n
EnableLocalReco: n

Kalman Mode:        ON
Tracking Systems Considered:        VT IT MSD
Reverse Tracking:   false

. . .
```

Will be put on anyhow by executable

Will be put off anyhow by executable

➡ from raw hits: DecodeGlbToe -in data/data_built.2211.physics_foot.daq.VTX.1.dat -out 12C_C_200_1_GlbOut.root -nev 10000-exp GSI -run 2211

Change input file, remove -mc flag

➡Should work, further tests needed

GenFit Global Actions (i)

- Actions:
  - The two classes KFitter and GlobalTrackingStudies do not inherit from TAGaction
  - The histogram handled by “hand”

➡ Put the two classes in the TAGaction framework

➡ Implemented the CreateHistograms() methods

➡ Update the BaseReco accordingly

➡ Using the TADIgenField interface instead of FootField (Remove this latter)

GenFit Global Actions (ii)

- BaseReco:

```
//_____________________________________________________________
void BaseReco::CreateRecActionGlbGF()
{
   if(fFlagTrack) {
      genfit::FieldManager::getInstance()->init( new TADIgenField(fField) );

      // set material and geometry into genfit
      MaterialEffects* materialEffects = MaterialEffects::getInstance();
      materialEffects->init(new TGeoMaterialInterface());

      // include the nucleon into the genfit pdg repository
      if ( GlobalPar::GetPar()->IncludeBM())
         UpdatePDG::Instance();

      // study for kalman Filter
      fActGlbTrackStudies = new GlobalTrackingStudies("glbActTrackStudyGF");
      if (fFlagHisto)
         fActGlbTrackStudies->CreateHistogram();

      // Initialisation of KFfitter
      fActGlbkFitter = new KFitter("glbActTrackGF");
      if (fFlagHisto)
         fActGlbkFitter->CreateHistogram();

   }
}
```

➡ Add reconstruction method when Kalman (GenFit) flag is on

➡ The two classes are in the FOOT framework

GenFit Executable (i)

- DecodeGlb:

```
int main(int argc, char *argv[])
{
. . .
   Bool_t lrc = GlobalPar::GetPar()->IsLocalReco();
   Bool_t ntu = GlobalPar::GetPar()->IsSaveTree();
   Bool_t his = GlobalPar::GetPar()->IsSaveHisto();
   Bool_t hit = GlobalPar::GetPar()->IsSaveHits();
   Bool_t trk = GlobalPar::GetPar()->IsTracking();
   Bool_t obj = GlobalPar::GetPar()->IsReadRootObj();
   Bool_t zmc = GlobalPar::GetPar()->IsTofZmc();
   Bool_t tbc = GlobalPar::GetPar()->IsTofCalBar();

   GlobalPar::GetPar()->IncludeTOE(false);
   GlobalPar::GetPar()->IncludeKalman(true);
   BaseReco* glbRec = 0x0;

   if (lrc)
      //glbRec = new GlobalReco(exp, runNb, in, out, mc);
      Error("DecodeGlb()", "No global reco from local reco tree available with GenFit");
   else if (mc) {
      if (!obj)
         glbRec = new LocalRecoMC(exp, runNb, in, out);
      else
         glbRec = new LocalRecoNtuMC(exp, runNb, in, out);
      if(zmc)
         glbRec->EnableZfromMCtrue();
   } else {
      glbRec = new LocalReco(exp, runNb, in, out);
      if (tbc)
         glbRec->EnableTWcalibPerBar();
   }. . .
}
```

➡ All arguments retrieved from GlobalPar, Enable Kalman (disable TOE)

➡ Same as for TOE, reconstruction from L0 output or raw data not yet available

GenFit Executable (ii)

- Command line:

```
. . .
IncludeKalman:   y
IncludeTOE:      n
EnableLocalReco: n

Kalman Mode:        ON
Tracking Systems Considered:        VT IT MSD
Reverse Tracking:   false

. . .
```

Will be put on anyhow by executable

Will be put off anyhow by executable

Rename DecodeGlob to be consistent with DecodeGlbToe

➡ from MC hits: **DecodeGlb** -in **12C_C_200_1.root**
-out 12C_C_200_1_GlbOut.root -nev 10000 -exp 12C_200 -run 1 -mc

```
. . .
IncludeKalman:   y
IncludeTOE:      n
EnableLocalReco: y

Kalman Mode:        ON
Tracking Systems Considered:        VT IT MSD
Reverse Tracking:   false

. . .
```

➡ from L0 reco: DecodeGlb -in **12C_C_200_L0Out.root**
-out 12C_C_200_1_GlbOut.root -nev 10000 -exp 12C_200 -run 1 -mc

GenFit Executable (iii)

- Command line:

```
. . .
IncludeKalman:   y
IncludeTOE:      n
EnableLocalReco: n

Kalman Mode:        ON
Tracking Systems Considered:        VT IT MSD
Reverse Tracking:   false

. . .
```

Will be put on anyhow by executable

Will be put off anyhow by executable

➡ from raw hits: **DecodeGlb** -in data/data_built.2211.physics_foot.daq.VTX.1.dat -out 12C_C_200_1_GlbOut.root -nev 10000-exp GSI -run 2211

Change input file remove -mc flag

➡ Not implemented yet

SubFolder (i)

- Histograms:
  - We have now around 400 control histograms in the output root file

```
TFile*        12C_C_200_info_Out.root
KEY: TH1I bmMcHitCell;1   cell index
KEY: TH1I bmMcHitView;1   view index
KEY: TH1I bmMcHitPlane;1  plane index
KEY: TH1F bmMcHitDischargedRdrift;1  Discharged hits according to BM efficiency
KEY: TH1F bmMcHitRdrift;1 Rdrift
KEY: TH1I bmMcHiDistribution;1   Number of ACCEPTED hits x event
KEY: TH1I bmMCHitFake;1   Charged hits fake index
KEY: TH2F bmTrackResidual;1   Residual vs Rdrift
KEY: TH2F bmTrackTargetMap;1  BM - Position of a single track event at target center
. . .
KEY: TH2D twdE_vs_Tof_Z2;1    dE_vs_Tof_2
KEY: TH1F twDist_Z2;1 dist_Z2
KEY: TH2D twdE_vs_Tof_Z3;1    dE_vs_Tof_3
KEY: TH1F twDist_Z3;1 dist_Z3
KEY: TH2D twdE_vs_Tof_Z4;1    dE_vs_Tof_4
KEY: TH1F twDist_Z4;1 dist_Z4
KEY: TH2D twdE_vs_Tof_Z5;1    dE_vs_Tof_5
KEY: TH1F twDist_Z5;1 dist_Z5
KEY: TH2D twdE_vs_Tof_Z6;1    dE_vs_Tof_6
KEY: TH1F twDist_Z6;1 dist_Z6
KEY: TH1F twDist;1    ToF Wall - Minimal distance between planes
KEY: TH1F twCharge1;1 ToF Wall - Charge layer 1
KEY: TH1F twCharge2;1 ToF Wall - Charge layer 2
KEY: TH1F twChargeTot;1   ToF Wall - Total charge
```

➡ Need to organize better histograms

➡Put in subfolder for each detector

SubFolder (ii)

- Reco clases:

```
void LocalReco::SetRawHistogramDir()
{
   // ST
   if (GlobalPar::GetPar()->IncludeST() || GlobalPar::GetPar()->IncludeTW()) {
      TDirectory* subfolder = fActEvtWriter->File()->mkdir(TASTparGeo::GetBaseName());
      fActWdRaw->SetHistogramDir(subfolder);
   }
. . .
}
```

```
void LocalRecoMC::SetRawHistogramDir()
{
   // ST
   if (GlobalPar::GetPar()->IncludeST()) {
      TDirectory* subfolder = fActEvtWriter->File()->mkdir(TASTparGeo::GetBaseName());
      fActNtuRawSt->SetHistogramDir(subfolder);
   }
. . .
}
```

```
//_____________________________________________________
void BaseReco::SetRecHistogramDir()
{
    //BMN
   if (GlobalPar::GetPar()->IncludeBM()) {
      if (fFlagTrack) {
         TDirectory* subfolder = (TDirectory*)(fActEvtWriter->File())->Get(TABMparGeo::GetBaseName());
         fActTrackBm->SetHistogramDir(subfolder);
      }
   }
. . .
}
```

SubFolder (iii)

- Histograms:
  - We have now 8 subfolders control histograms in the output root file

```
TFile*     test2.root
  KEY: TDirectoryFile   ST;1    ST
  KEY: TDirectoryFile   BM;1    BM
  KEY: TDirectoryFile   VT;1    VT
  KEY: TDirectoryFile   IT;1    IT
  KEY: TDirectoryFile   MSD;1   MSD
  KEY: TDirectoryFile   TW;1    TW
  KEY: TDirectoryFile   CA;1    CA
  KEY: TDirectoryFile   FOOT;1  FOOT
```

➡ Easier to find a given histogram

➡ FOOT folder stands for global reconstruction histograms

Conclusions

- Reconstruction options in global parameters
- Run informations automatically save/read in root file
- Put GenFit classes into framework
- The README files in fullrec and level0 contain command line instructions
- Create histogram subfolders for root output file
- Revive some macros to be compliant with campaign manager

➡ More simplified interface for users

➡ Should separate reconstruction from analysis (done offline of the reconstruction)

➡ We are starting to have a nice coherent and safe code !