MC Root Object

Introduction

Txt2NtuRoot

MC actions

Local Reco

MC Executable

G4 executable

Fluka Reader

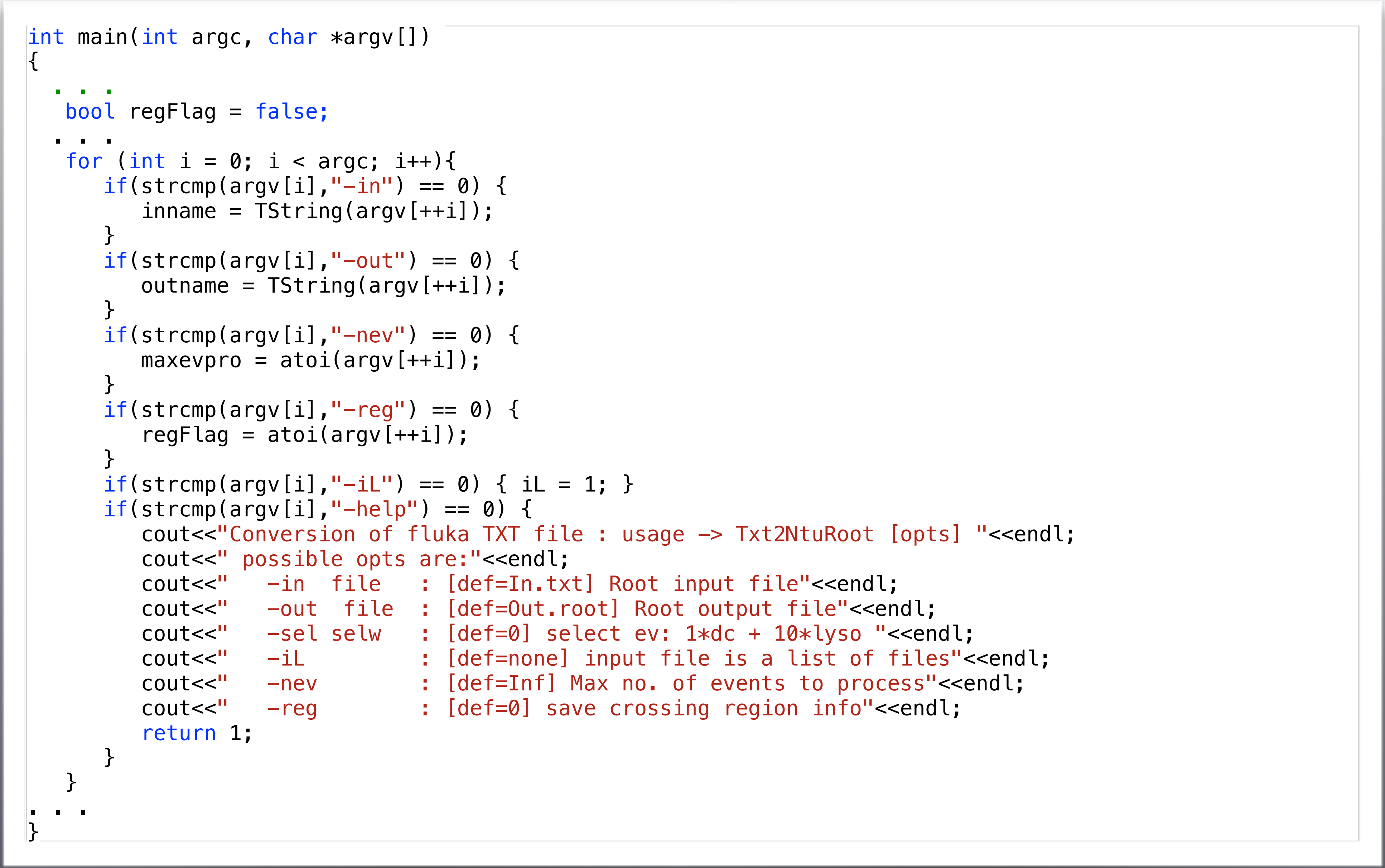
Conclusion

Introduction

- Presently converting the Fluka output in the root tree with the EVENT_STRUCT structure (so with ~170 branches).
- Arrays hard coded in EVENT_STRUCT (const int MAXTR = 6000), could lead to underflow, this is the case, especially, when adding electrons
- Need to convert branches into root object with action TAMCactNtu* to TAMCeveTrack and TAMChit objects
- Comparing reconstructed data with MC data, do it on fly or need to copy MC data in the reconstructed data root output file.

- Save disk space and CPU time, if directly converting Fluka output in root objects.
- Moreover could use the framework (TAGactTreeReader) to read back objects.

Txt2NtuRoot

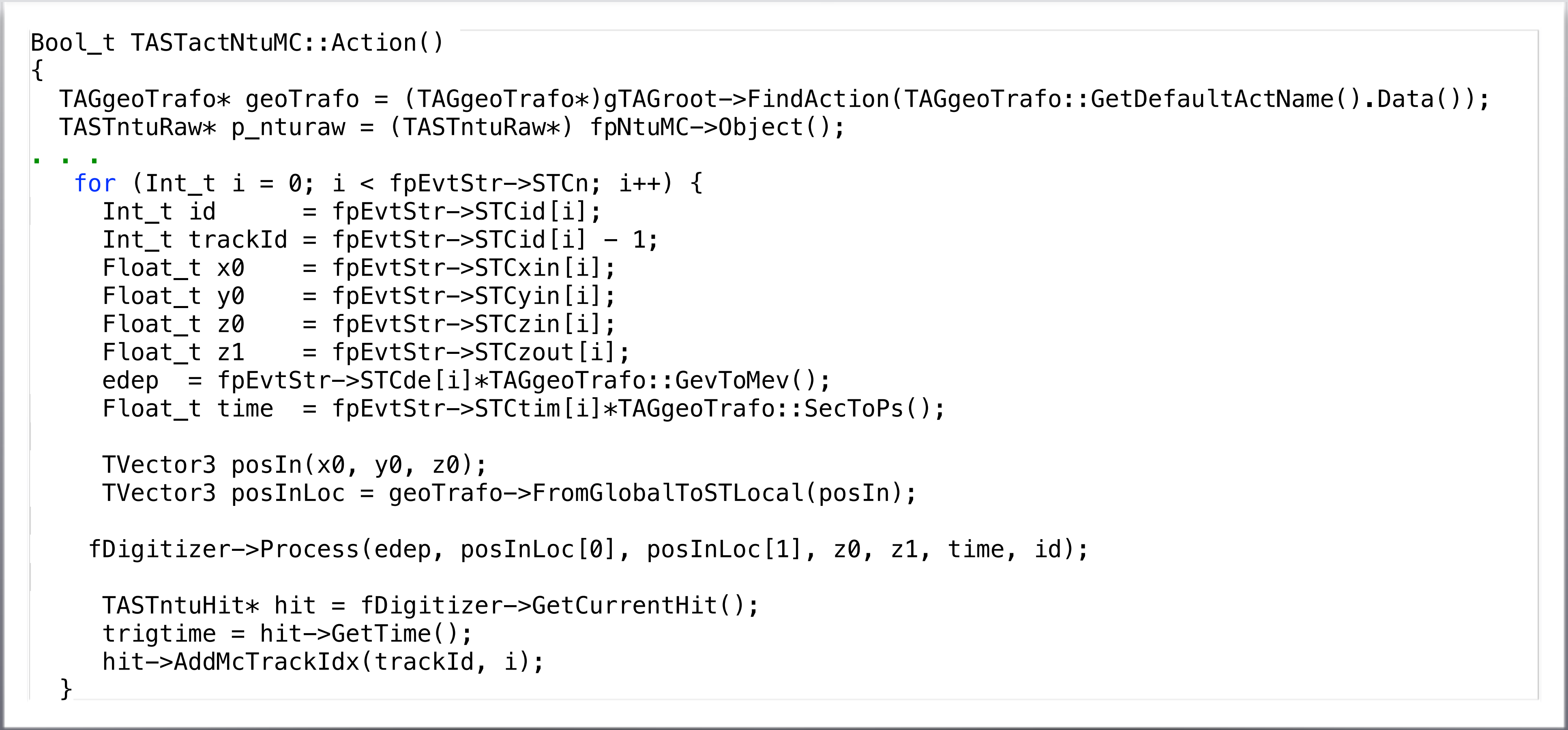


```
int main(int argc, char *argv[])
{
  . . .
   bool regFlag = false;
  . . .
   for (int i = 0; i < argc; i++){
      if(strcmp(argv[i],"-in") == 0) {
         inname = TString(argv[++i]);
      }
      if(strcmp(argv[i],"-out") == 0) {
         outname = TString(argv[++i]);
      }
      if(strcmp(argv[i],"-nev") == 0) {
         maxevpro = atoi(argv[++i]);
      }
      if(strcmp(argv[i],"-reg") == 0) {
         regFlag = atoi(argv[++i]);
      }
      if(strcmp(argv[i],"-iL") == 0) { iL = 1; }
      if(strcmp(argv[i],"-help") == 0) {
         cout<<"Conversion of fluka TXT file : usage -> Txt2NtuRoot [opts] "<<endl;
         cout<<" possible opts are:"<<endl;
         cout<<"   -in  file   : [def=In.txt] Root input file"<<endl;
         cout<<"   -out  file  : [def=Out.root] Root output file"<<endl;
         cout<<"   -sel selw   : [def=0] select ev: 1*dc + 10*lyso "<<endl;
         cout<<"   -iL         : [def=none] input file is a list of files"<<endl;
         cout<<"   -nev        : [def=Inf] Max no. of events to process"<<endl;
         cout<<"   -reg        : [def=0] save crossing region info"<<endl;
         return 1;
      }
   }
. . .
}
```

➡ Option to save or not crossing region array

MC actions (i)

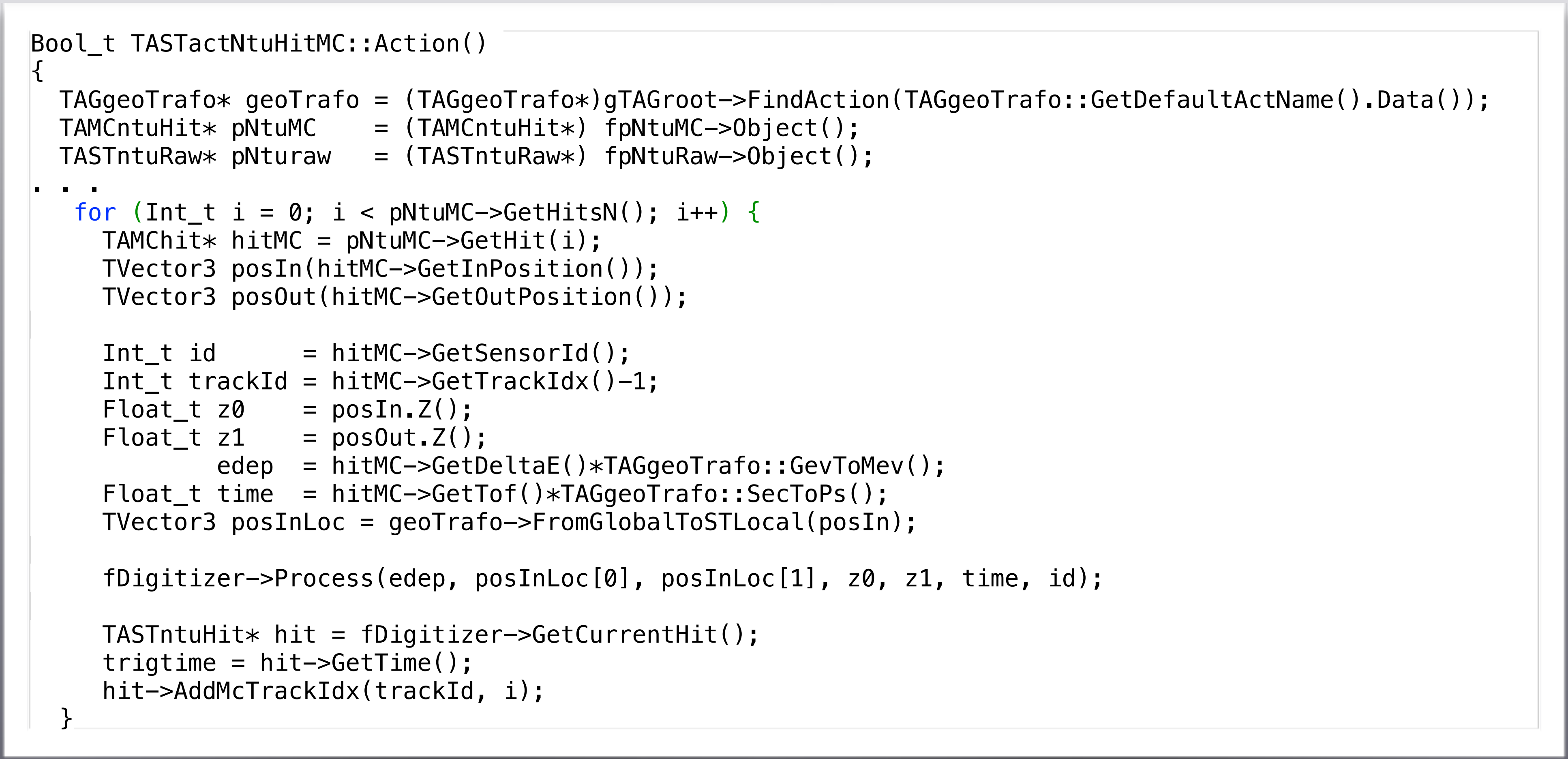
- TASTactNtuMC:



```
Bool_t TASTactNtuMC::Action()
{
  TAGgeoTrafo* geoTrafo = (TAGgeoTrafo*)gTAGroot->FindAction(TAGgeoTrafo::GetDefaultActName().Data());
  TASTntuRaw* p_nturaw = (TASTntuRaw*) fpNtuMC->Object();
. . .
   for (Int_t i = 0; i < fpEvtStr->STCn; i++) {
     Int_t id      = fpEvtStr->STCid[i];
     Int_t trackId = fpEvtStr->STCid[i] - 1;
     Float_t x0    = fpEvtStr->STCxin[i];
     Float_t y0    = fpEvtStr->STCyin[i];
     Float_t z0    = fpEvtStr->STCzin[i];
     Float_t z1    = fpEvtStr->STCzout[i];
     edep  = fpEvtStr->STCde[i]*TAGgeoTrafo::GevToMev();
     Float_t time  = fpEvtStr->STCtim[i]*TAGgeoTrafo::SecToPs();

     TVector3 posIn(x0, y0, z0);
     TVector3 posInLoc = geoTrafo->FromGlobalToSTLocal(posIn);

    fDigitizer->Process(edep, posInLoc[0], posInLoc[1], z0, z1, time, id);

     TASTntuHit* hit = fDigitizer->GetCurrentHit();
     trigtime = hit->GetTime();
     hit->AddMcTrackIdx(trackId, i);
  }
```

MC actions (ii)

- TASTactNtuHitMC:



```cpp
Bool_t TASTactNtuHitMC::Action()
{
  TAGgeoTrafo* geoTrafo = (TAGgeoTrafo*)gTAGroot->FindAction(TAGgeoTrafo::GetDefaultActName().Data());
  TAMCntuHit* pNtuMC    = (TAMCntuHit*) fpNtuMC->Object();
  TASTntuRaw* pNturaw   = (TASTntuRaw*) fpNtuRaw->Object();
. . .
   for (Int_t i = 0; i < pNtuMC->GetHitsN(); i++) {
     TAMChit* hitMC = pNtuMC->GetHit(i);
     TVector3 posIn(hitMC->GetInPosition());
     TVector3 posOut(hitMC->GetOutPosition());

     Int_t id      = hitMC->GetSensorId();
     Int_t trackId = hitMC->GetTrackIdx()-1;
     Float_t z0    = posIn.Z();
     Float_t z1    = posOut.Z();
             edep  = hitMC->GetDeltaE()*TAGgeoTrafo::GevToMev();
     Float_t time  = hitMC->GetTof()*TAGgeoTrafo::SecToPs();
     TVector3 posInLoc = geoTrafo->FromGlobalToSTLocal(posIn);

     fDigitizer->Process(edep, posInLoc[0], posInLoc[1], z0, z1, time, id);

     TASTntuHit* hit = fDigitizer->GetCurrentHit();
     trigtime = hit->GetTime();
     hit->AddMcTrackIdx(trackId, i);
  }
```

➡ Loop over hit objects, same way for all other detectors

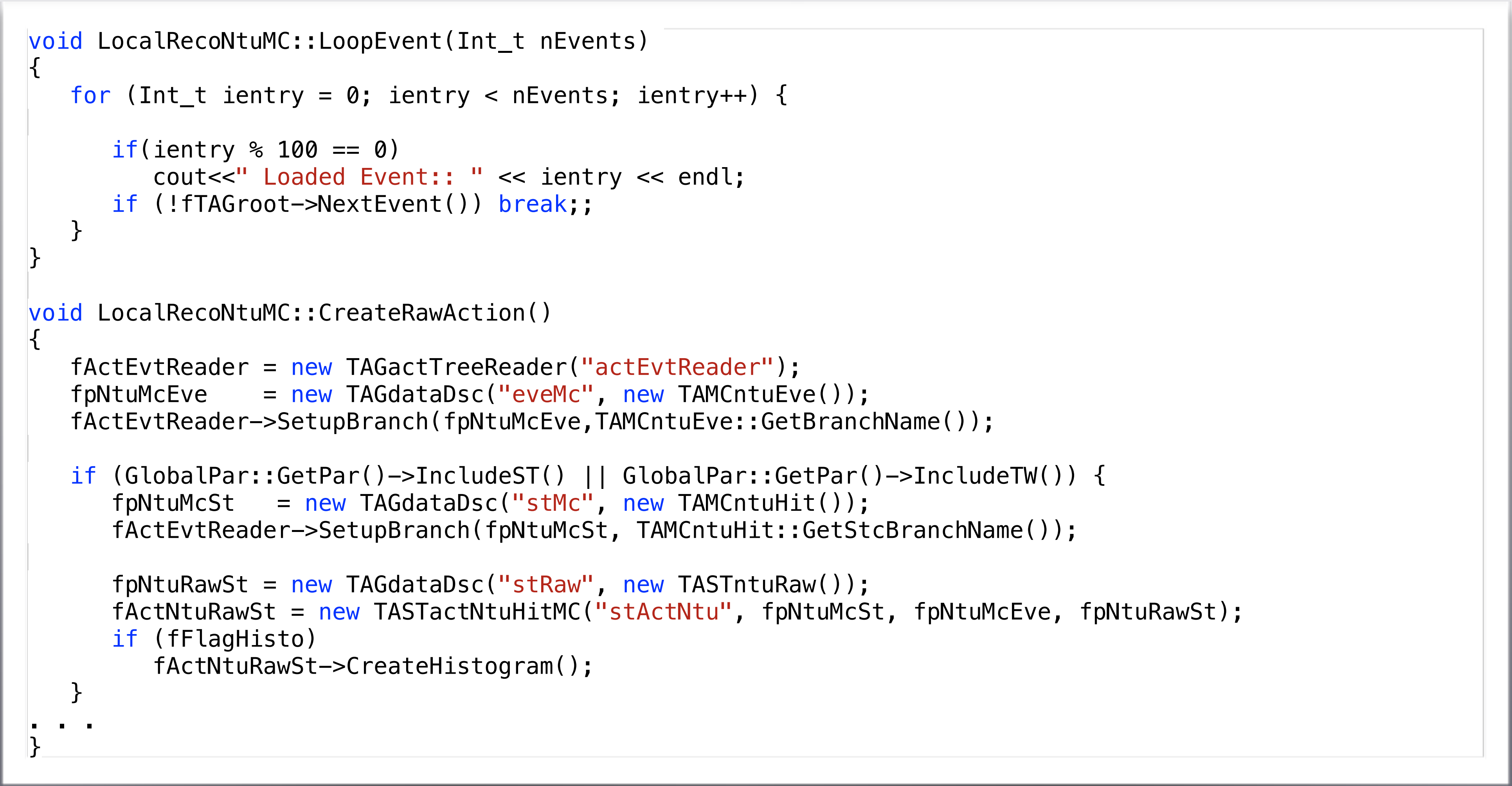
Local Reco (i)

- LocalRecoMC:

```
void LocalRecoMC::LoopEvent(Int_t nEvents)
{
. . .
   for (Long64_t ientry = 0; ientry < nEvents; ientry++) {

      fTree->GetEntry(ientry);

      if(ientry % 100 == 0)
         cout<<" Loaded Event:: " << ientry << endl;
      if (!fTAGroot->NextEvent()) break;
   }
}

void LocalRecoMC::CreateRawAction()
{
   fpNtuMcEve = new TAGdataDsc("eveMc", new TAMCntuEve());
   fActNtuMcEve = new TAMCactNtuEve("eveActNtuMc", fpNtuMcEve, fEvtStruct);


   if (GlobalPar::GetPar()->IncludeST()) {
      fpNtuRawSt = new TAGdataDsc("stRaw", new TASTntuRaw());
      fActNtuRawSt = new TASTactNtuMC("stActNtu", fpNtuRawSt, fEvtStruct);
      if (fFlagHisto)
         fActNtuRawSt->CreateHistogram();

      fpNtuMcSt   = new TAGdataDsc("stMc", new TAMCntuHit());
      fActNtuMcSt = new TAMCactNtuStc("stActNtuMc", fpNtuMcSt, fEvtStruct);
   }
. . .
}
```

Local Reco (ii)

- LocalRecoNtuMC:

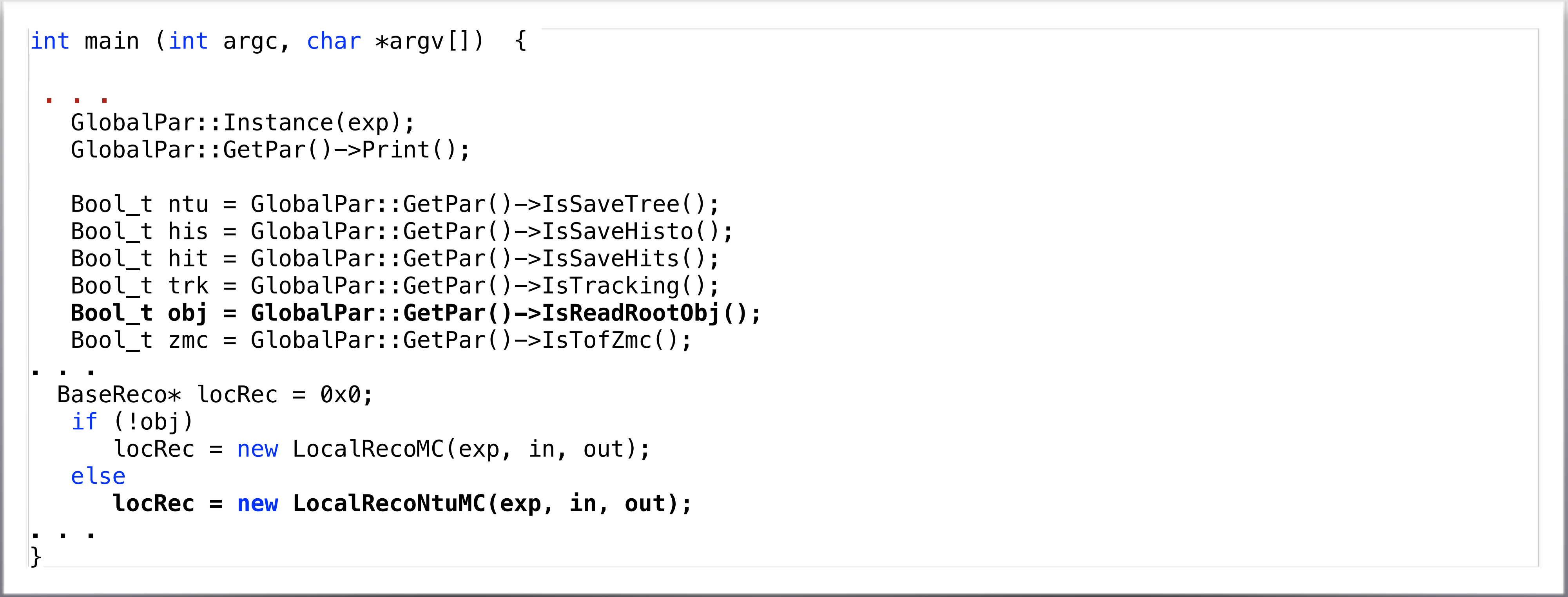


```
void LocalRecoNtuMC::LoopEvent(Int_t nEvents)
{
   for (Int_t ientry = 0; ientry < nEvents; ientry++) {

      if(ientry % 100 == 0)
         cout<<" Loaded Event:: " << ientry << endl;
      if (!fTAGroot->NextEvent()) break;;
   }
}

void LocalRecoNtuMC::CreateRawAction()
{
   fActEvtReader = new TAGactTreeReader("actEvtReader");
   fpNtuMcEve    = new TAGdataDsc("eveMc", new TAMCntuEve());
   fActEvtReader->SetupBranch(fpNtuMcEve,TAMCntuEve::GetBranchName());

   if (GlobalPar::GetPar()->IncludeST() || GlobalPar::GetPar()->IncludeTW()) {
      fpNtuMcSt   = new TAGdataDsc("stMc", new TAMCntuHit());
      fActEvtReader->SetupBranch(fpNtuMcSt, TAMCntuHit::GetStcBranchName());

      fpNtuRawSt = new TAGdataDsc("stRaw", new TASTntuRaw());
      fActNtuRawSt = new TASTactNtuHitMC("stActNtu", fpNtuMcSt, fpNtuMcEve, fpNtuRawSt);
      if (fFlagHisto)
         fActNtuRawSt->CreateHistogram();
   }
. . .
}
```

- ➡ Directly read object from tree with TAGactTreeReader class and then create raw hits with TA*actNtuHitMC classes
- ➡ Do not need to copy EVENT_STRUCT in TAMCntuHit containers
- ➡ Will simplify interface for global reconstruction from level0 tree

MC Executable

- DecodeMC:

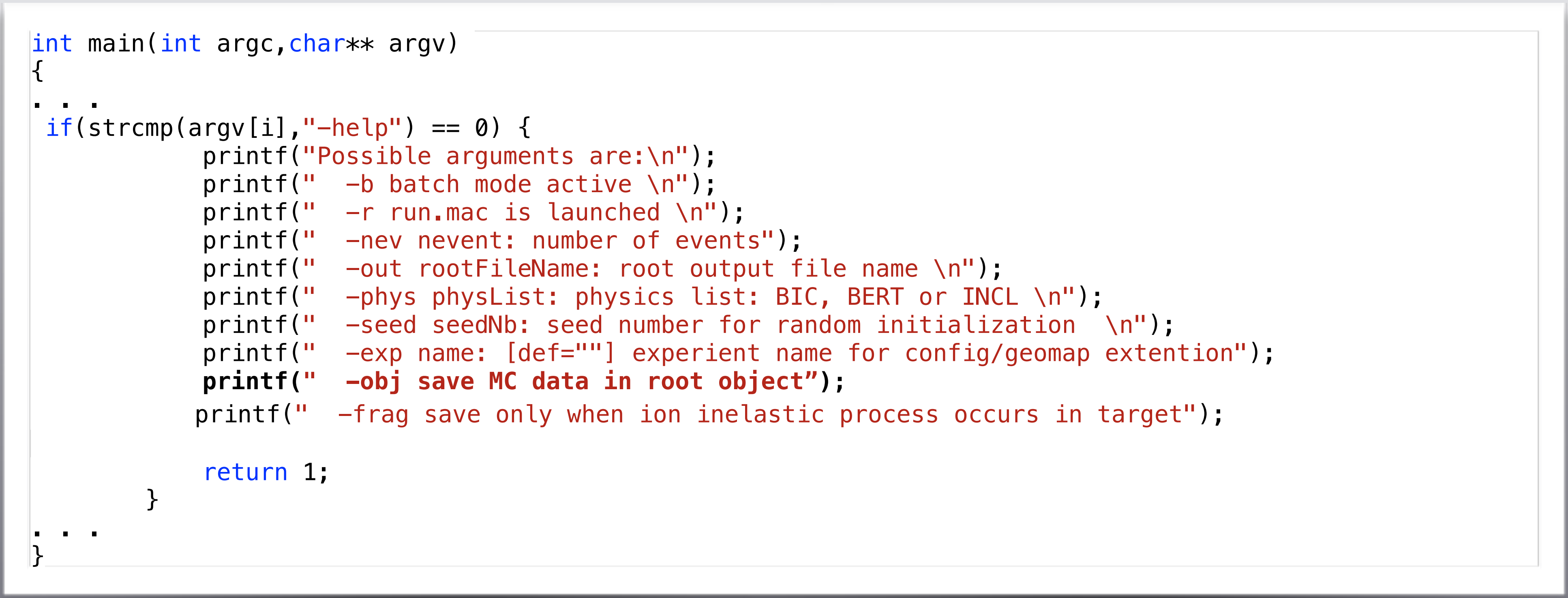


```
int main (int argc, char *argv[])  {

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

   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();
. . .
  BaseReco* locRec = 0x0;
   if (!obj)
      locRec = new LocalRecoMC(exp, in, out);
   else
      locRec = new LocalRecoNtuMC(exp, in, out);
. . .
}
```

➡ Retrieve *IsReadRootObj* flag from global par to call LocalRecoNtuMC class

G4 executable

- TAGsimulation:



```
int main(int argc,char** argv)
{
. . .
 if(strcmp(argv[i],"-help") == 0) {
           printf("Possible arguments are:\n");
           printf("  -b batch mode active \n");
           printf("  -r run.mac is launched \n");
           printf("  -nev nevent: number of events");
           printf("  -out rootFileName: root output file name \n");
           printf("  -phys physList: physics list: BIC, BERT or INCL \n");
           printf("  -seed seedNb: seed number for random initialization  \n");
           printf("  -exp name: [def=""] experient name for config/geomap extention");
           printf("  -obj save MC data in root object”);
           printf("  -frag save only when ion inelastic process occurs in target");

           return 1;
       }
. . .
}
```

➡ Add option -*obj* to save MC data in root object

Fluka Reader (i)

- TAGactTreeReader:

```
class TAGactTreeReader : public TAGactionFile {
  public:
    explicit        TAGactTreeReader(const char* name=0);
    virtual         ~TAGactTreeReader();

    void            SetupBranch(TAGdataDsc* p_data, const char* branch);

    virtual Int_t   Open(const TString& name, Option_t* option="READ", const TString treeName="tree",
                         Bool_t dscBranch = true);
. . .
};
```

➡ Add flag to read back branches that do not inherited from TAGdataDsc (i.e. EVENT_STRUCT)

Fluka Reader (ii)

- LocalRecoMC:

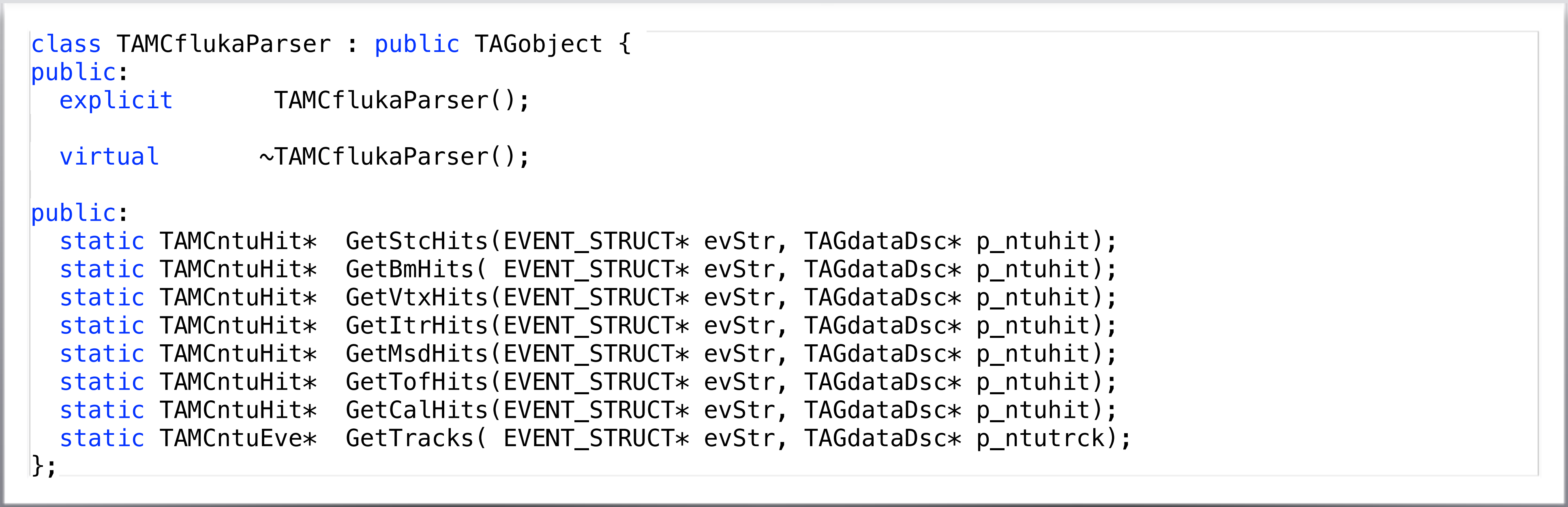
```
//____________________________________________________________
void LocalRecoMC::OpenFileIn()
{
  fActEvtReader->Open(GetName(), "READ", "EventTree", false);
  fTree = fActEvtReader->GetTree();

   Evento *ev  = new Evento();
   ev->FindBranches(fTree, fEvtStruct);
}
. . .
//____________________________________________________________
void LocalRecoMC::LoopEvent(Int_t nEvents)
{
   for (Long64_t ientry = 0; ientry < nEvents; ientry++) {
      if(ientry % 100 == 0)
         cout<<" Loaded Event:: " << ientry << endl;
      if (!fTAGroot->NextEvent()) break;
   }
}

//____________________________________________________________
void LocalRecoMC::CreateRawAction()
{
  fActEvtReader = new TAGactTreeReader(“actEvtReader");
. . .
}
```

➡ Using TAGactTreeReader interface to read back Fluka structure

➡ Loop Event identical for MC/raw data, could be put in base class now

Fluka Reader (iii)

- TAMCflukaParser:



```
class TAMCflukaParser : public TAGobject {
public:
  explicit       TAMCflukaParser();

  virtual       ~TAMCflukaParser();

public:
  static TAMCntuHit*  GetStcHits(EVENT_STRUCT* evStr, TAGdataDsc* p_ntuhit);
  static TAMCntuHit*  GetBmHits( EVENT_STRUCT* evStr, TAGdataDsc* p_ntuhit);
  static TAMCntuHit*  GetVtxHits(EVENT_STRUCT* evStr, TAGdataDsc* p_ntuhit);
  static TAMCntuHit*  GetItrHits(EVENT_STRUCT* evStr, TAGdataDsc* p_ntuhit);
  static TAMCntuHit*  GetMsdHits(EVENT_STRUCT* evStr, TAGdataDsc* p_ntuhit);
  static TAMCntuHit*  GetTofHits(EVENT_STRUCT* evStr, TAGdataDsc* p_ntuhit);
  static TAMCntuHit*  GetCalHits(EVENT_STRUCT* evStr, TAGdataDsc* p_ntuhit);
  static TAMCntuEve*  GetTracks( EVENT_STRUCT* evStr, TAGdataDsc* p_ntutrck);
};
```

➡ Fill TAMCntuHit and TAMCntuEve containers from the Fluka structure

➡ Can convert on fly Fluka structure into root object, avoiding using all TAMCactNtu* actions

➡ All methods are static

Fluka Reader (iv)

- TATWactNtuHitMC (as an example):

```
bool TATWactNtuHitMC::Action() {

   if(FootDebugLevel(1))
     cout << "TATWactNtuHitMC::Action() start" << endl;

   TAMCntuHit* pNtuMC   = 0x0;
   TAMCntuHit* pNtuStMC = 0x0;
   TAMCntuEve* pNtuEve  = 0x0;

  if (fEventStruct == 0x0) {
    pNtuMC   = (TAMCntuHit*) fpNtuMC->Object();
    pNtuStMC = (TAMCntuHit*) fpNtuStMC->Object();
    pNtuEve  = (TAMCntuEve*) fpNtuEve->Object();
  } else {
    pNtuMC   = TAMCflukaParser::GetTofHits(fEventStruct, fpNtuMC);
    pNtuStMC = TAMCflukaParser::GetStcHits(fEventStruct, fpNtuStMC);
    pNtuEve  = TAMCflukaParser::GetTracks(fEventStruct, fpNtuEve);
  }
. . .
}
```

➡ If Fluka structure not nil, convert, on fly, into root object thanks to parser

➡ Only keep one MC readout class, avoid copy/paste

➡ Not yet committed cos not fully tested, could be applied for all other detectors

Conclusion

- Reconstruction from root object available
- Flag/option to chose btw both readers
- Re-Tests done for all detectors
- Test done also with Geant4 (add some options)

➡ Make a decision btw Txt2Root and Txt2NtuRoot for next MC files

➡ Avoiding copy/paste actions

➡ Could keep Fluka structure for old MC files with the help of the parser, but need some work and tests.