# Global Reconstruction Framework

**Strategy**

**Requirements**

**Kalman Filter**

**Reconstruction**

**Reconstruction Action**

**Conclusions**

# Strategy (i)

## (adopted from FIRST)

- Backward approach:

1. Start from TW hits with atomic charge Z, assuming
   a. A = Zx2 (except for Z = 1 thus A = 1)
   b. $E_c = E_{beam}/A$ (could be given by CAL)
2. Extrapole to MSD, combine with a local track
3. Extrapole to IT, search for closest cluster
4. Extrapole to VTX, search for the closest track for a vertex matched with BM
5. Feed the Kalman filter and process
6. If good store candidate, else repeat 1a changing hypothesis

# Strategy (ii)

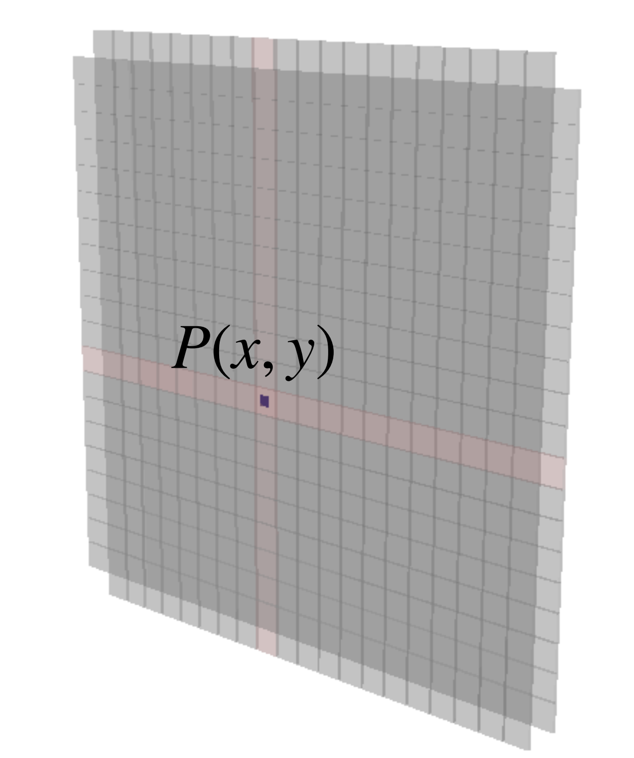
1. Start from TW hits with atomic charge Z

$$P(x,y) = f(L_{bar}, v_{propa}, T_{left/right})*$$

$$Z \propto \sqrt{\Delta E} \times \beta$$

$$\beta = \frac{L}{cToF} \; with \; L = f(\overrightarrow{B}, A, Z)$$

➡ assumption L as straight line



(*Done in TATWactNtuPoint class)

# Strategy (ii)

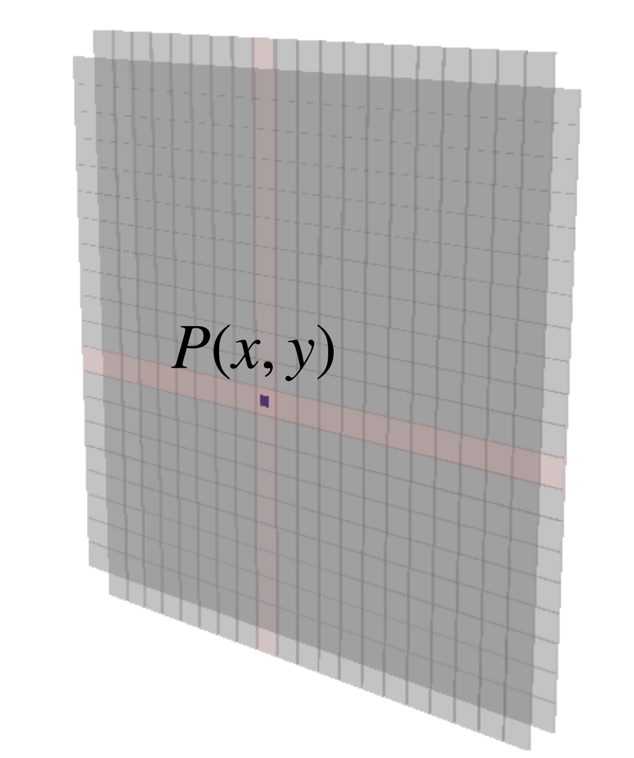
1. Start from TW hits with atomic charge Z

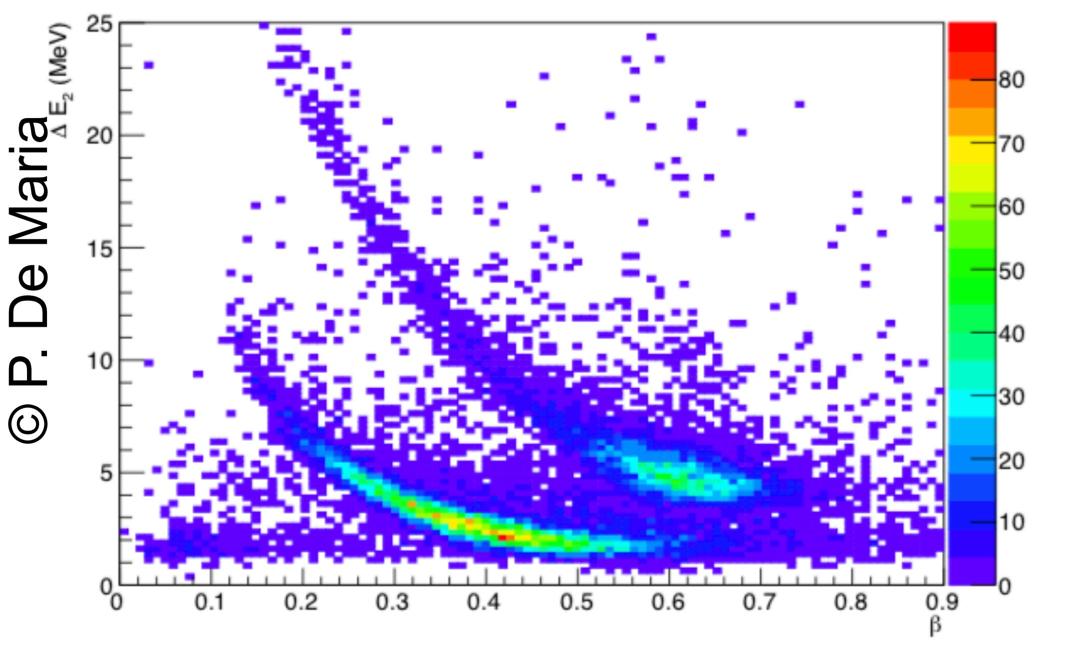
$$P(x,y) = f(L_{bar}, v_{propa}, T_{left/right})^{*}$$

$$Z \propto \sqrt{\Delta E} \times \beta$$

$$\beta = \frac{L}{cToF} \; with \; L = f(\vec{B}, A, Z)$$

➡ assumption L as straight line



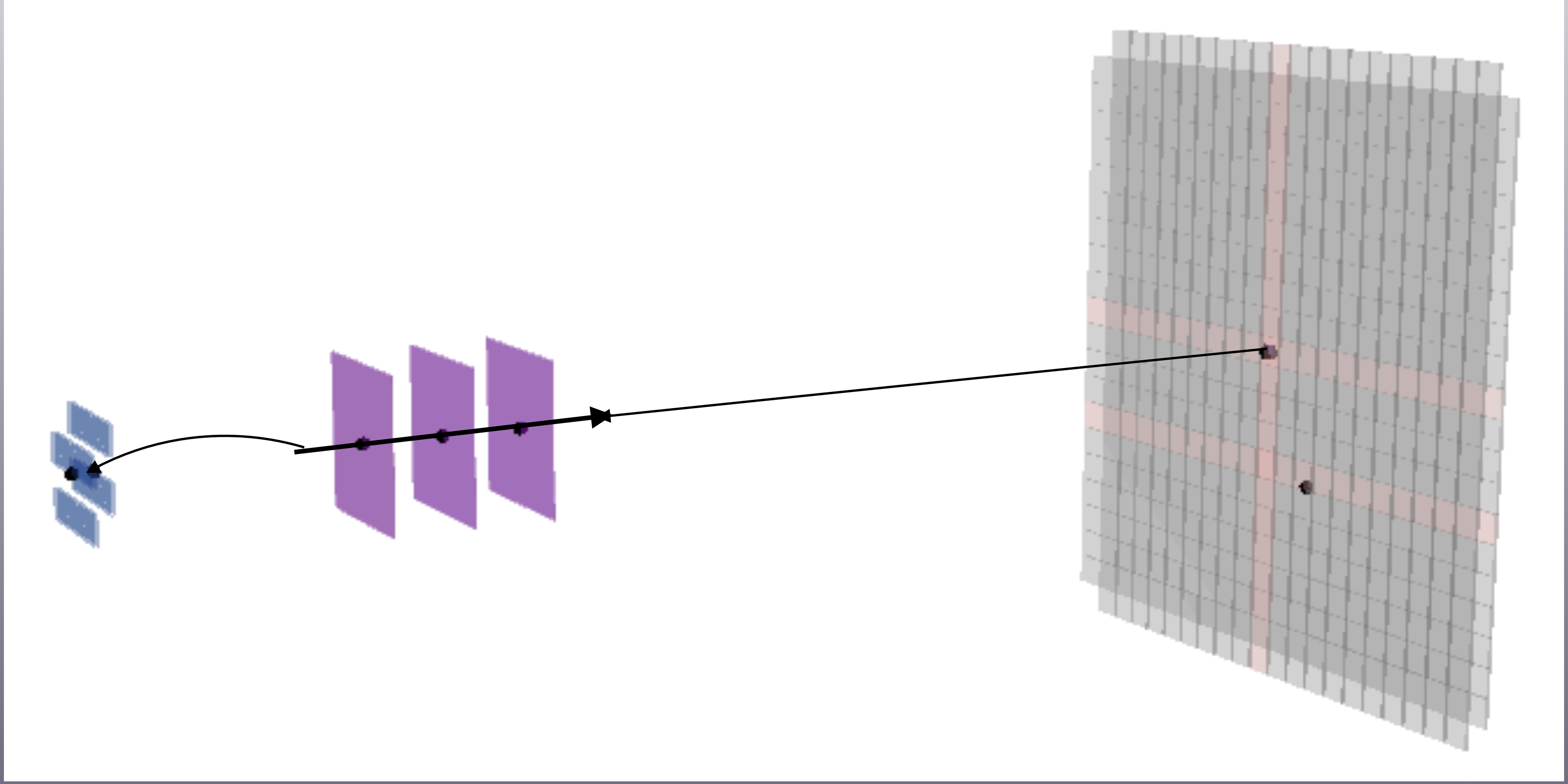


➡ Each line fits with a charge state

(*Done in TATWactNtuPoint class)

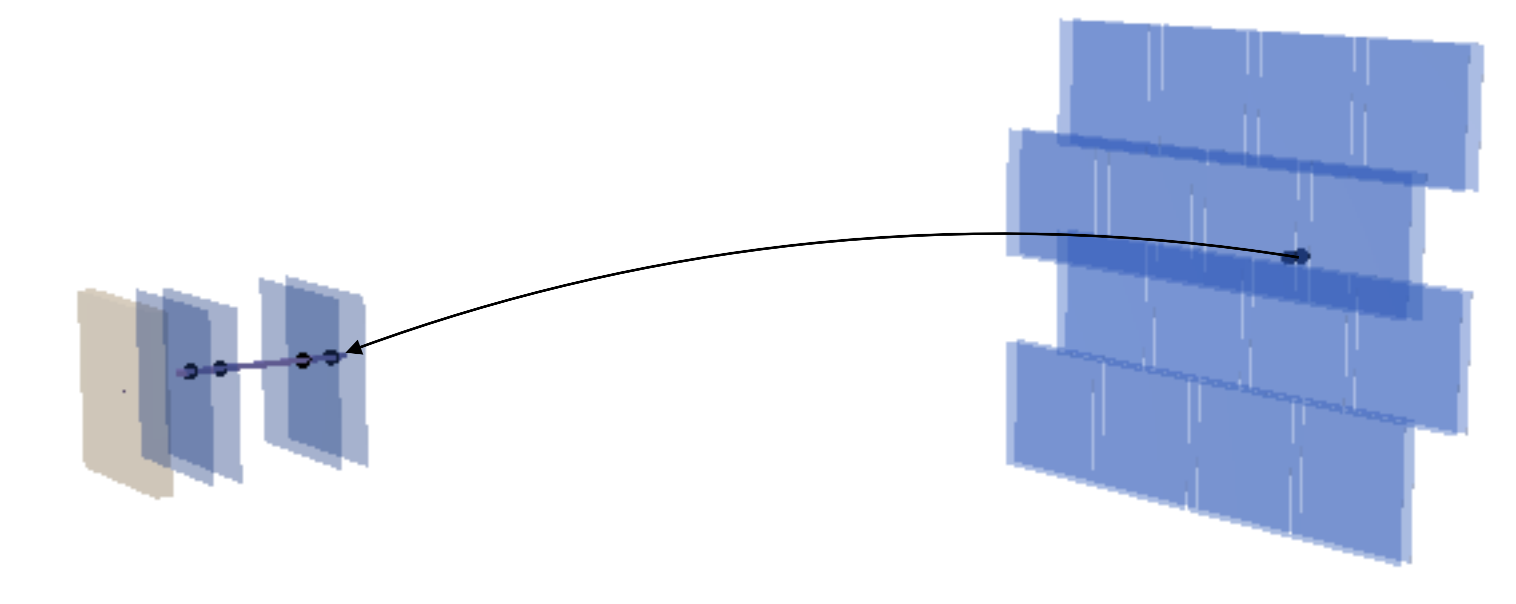
# Strategy (iii)

2. Extrapole to MSD, combine with a local track (no field)
3. Extrapole to IT, search for closest cluster (propagation in field)



# Strategy (iv)

4. Extrapole to VTX, search for the closest track for a vertex matched with BM (Propgation in Field)



5. Feed the Kalman filter and process

6. If good store candidate, else repeat 1a changing hypothesis

# Requirements

BM:

- Tracking: ✔

VTX:

- Vertexing: ✔

IT:

- Clustering: ✔

MSD:

- 1D Clustering: ✔
- 2D Clustering/Tracking: ✘

TW:

- Point reconstruction: ✔
- Atomic charge reconstruction: ✘

CAL:

- Kinetic energy reconstruction: ✘

Digitizer:

- ST+BM+VTX+IT: ✔
- MSD+TW+CAL: need more work

Rem: MC track index propagated from hits to tracks
when cluster/track exits

# Kalman Filter

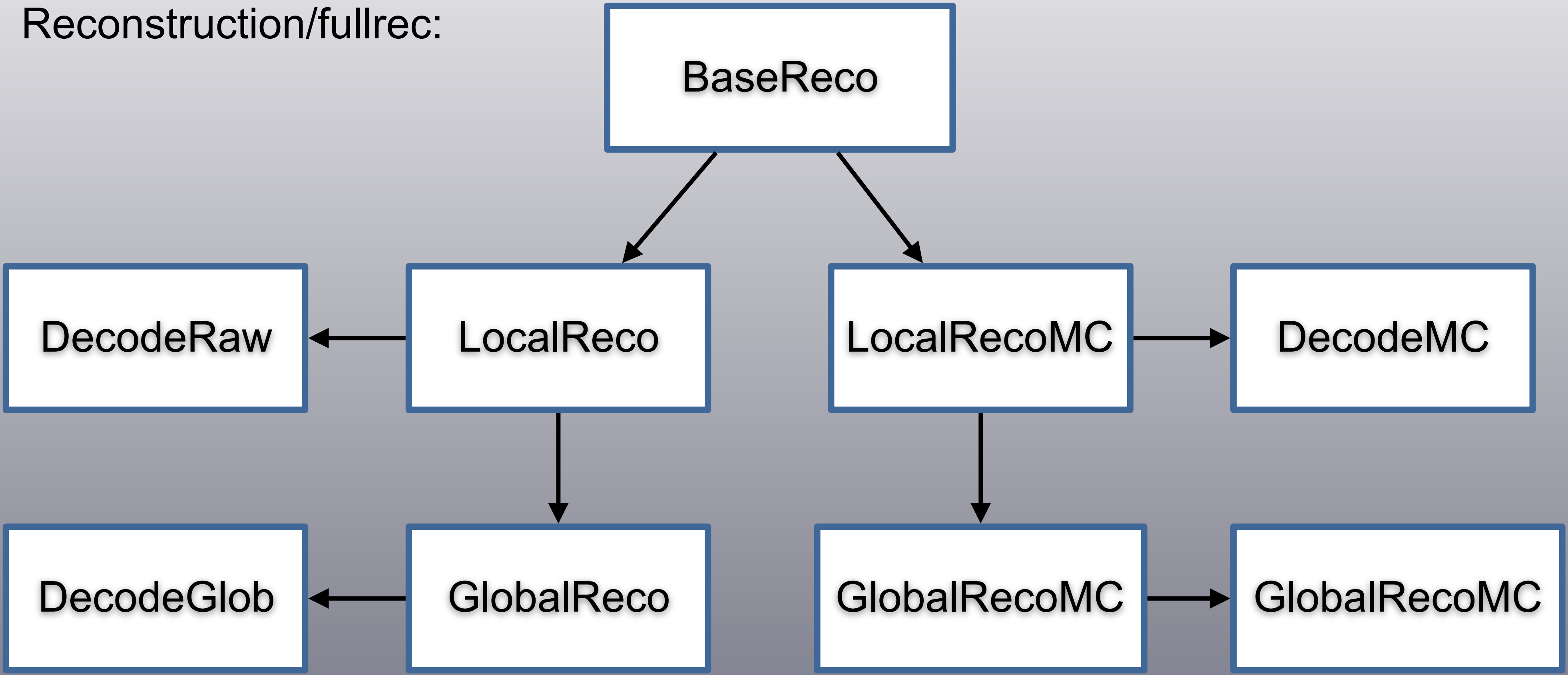
- GenFit: heavy package, need a simplified approach for better control

➡ Development of a new global reconstruction TOE (Tracking Of Ejectile):

- new Kalman filter (UKF): under development
- new propagator (RK5-6): done
- test with “scholar” data: ongoing

# Global reconstruction
## (new framework)

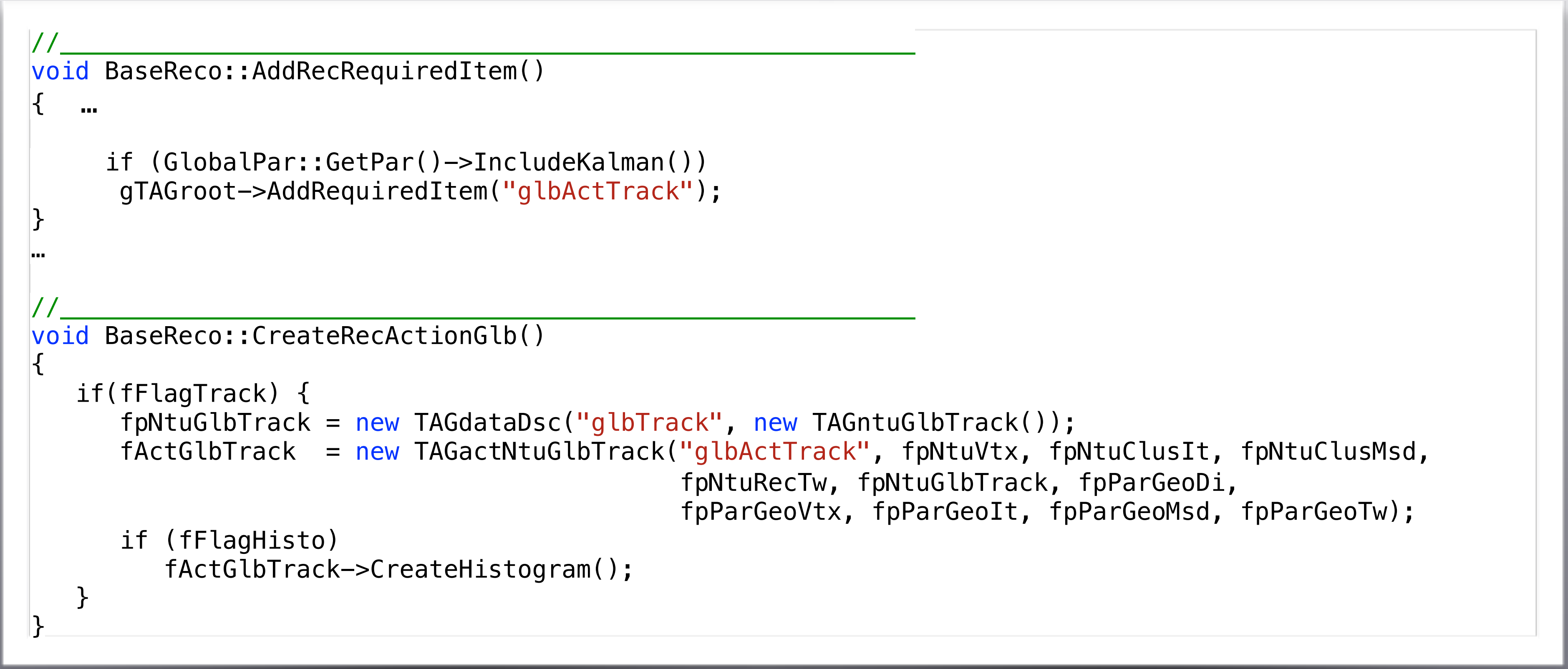
- Reconstruction/fullrec:



- Global reconstruction inherits from local reconstruction
- Dedicated actions for global reconstruction

# Global reconstruction (i)

- BaseReco: change name of base class

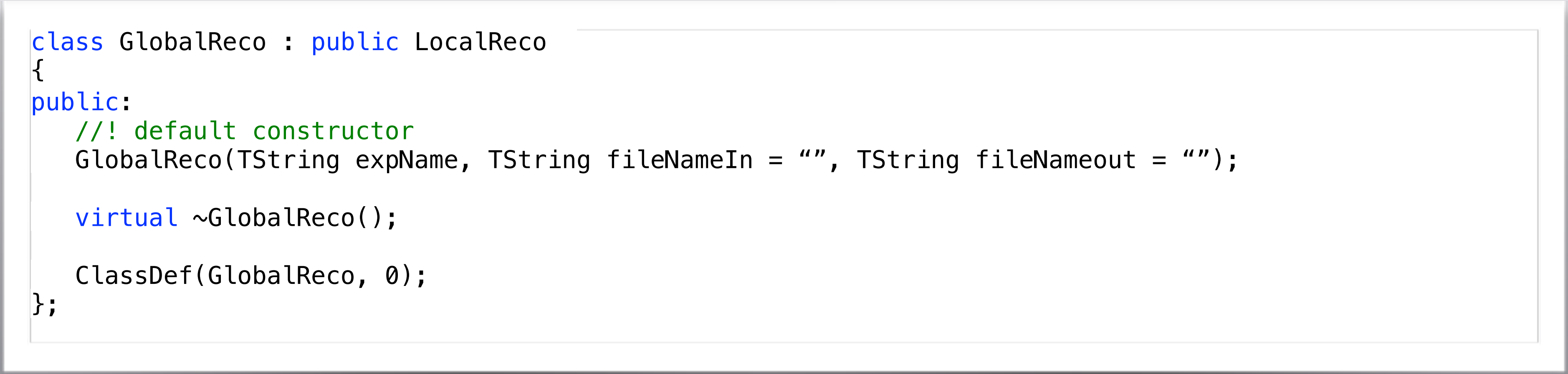


```
//____________________________________________________________
void BaseReco::AddRecRequiredItem()
{   …

     if (GlobalPar::GetPar()->IncludeKalman())
      gTAGroot->AddRequiredItem("glbActTrack");
}
…

//____________________________________________________________
void BaseReco::CreateRecActionGlb()
{
   if(fFlagTrack) {
      fpNtuGlbTrack = new TAGdataDsc("glbTrack", new TAGntuGlbTrack());
      fActGlbTrack  = new TAGactNtuGlbTrack("glbActTrack", fpNtuVtx, fpNtuClusIt, fpNtuClusMsd,
                                            fpNtuRecTw, fpNtuGlbTrack, fpParGeoDi,
                                            fpParGeoVtx, fpParGeoIt, fpParGeoMsd, fpParGeoTw);
      if (fFlagHisto)
         fActGlbTrack->CreateHistogram();
   }
}
```

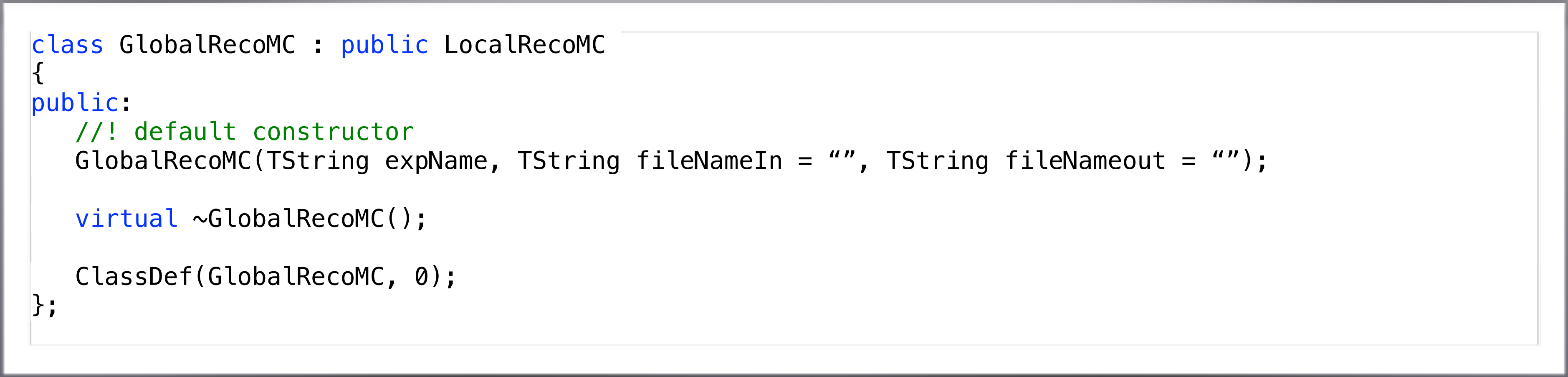
- Add dedicated classes for global reconstruction
  - ➡ TAGntuGlbTrack: global track container
  - ➡ TAGactNtuGlbTrack: global tracking action

# Global reconstruction (ii)

- GlobalReco:



```
class GlobalReco : public LocalReco
{
public:
   //! default constructor
   GlobalReco(TString expName, TString fileNameIn = “”, TString fileNameout = “”);

   virtual ~GlobalReco();

   ClassDef(GlobalReco, 0);
};
```

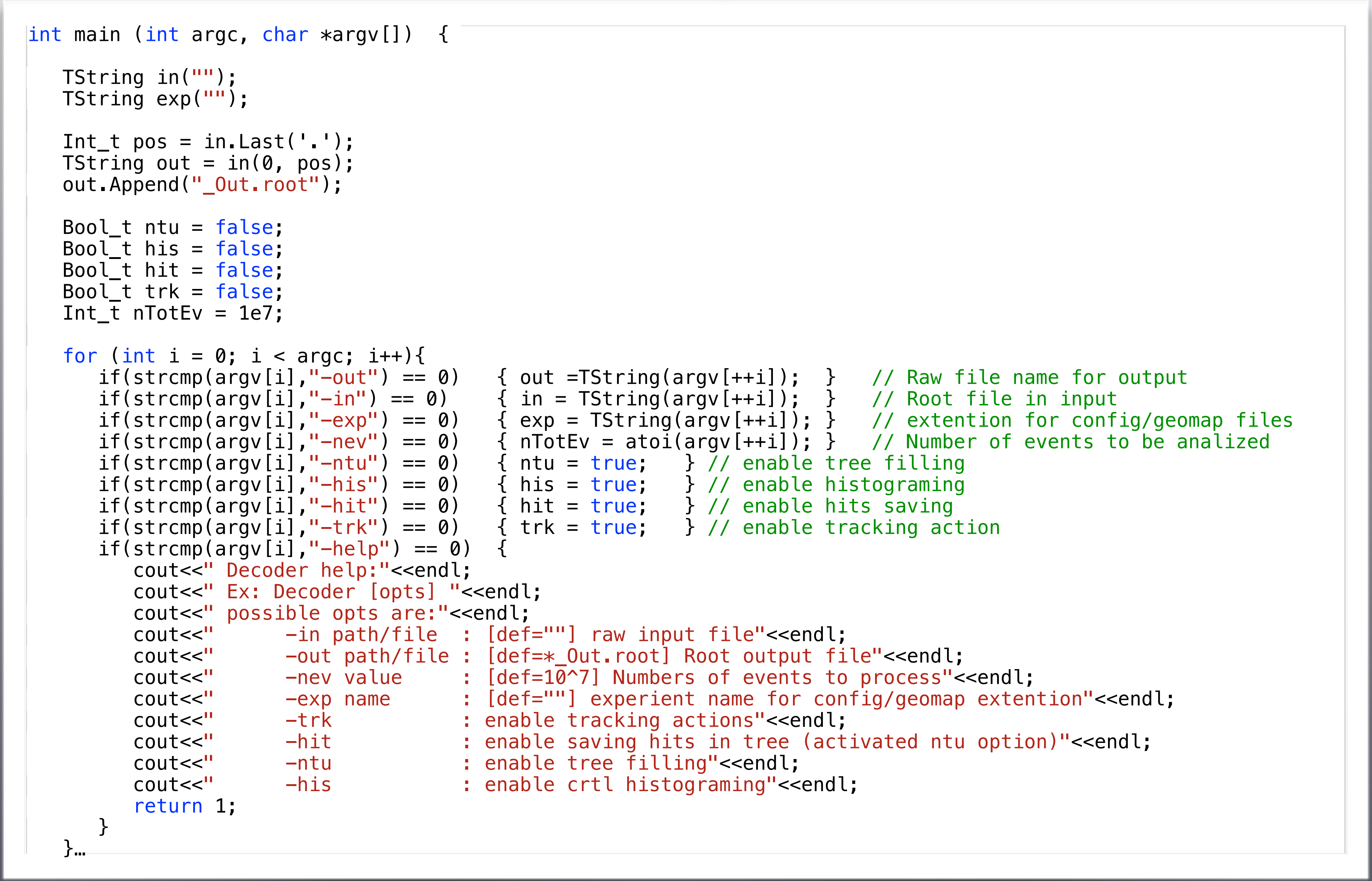


```
class GlobalRecoMC : public LocalRecoMC
{
public:
   //! default constructor
   GlobalRecoMC(TString expName, TString fileNameIn = “”, TString fileNameout = “”);

   virtual ~GlobalRecoMC();

   ClassDef(GlobalRecoMC, 0);
};
```

- Virtual interface for executables, should NOT be modified
- WARNING: for local reconstruction, *IncludeKalman* must be off in FootGlobal.par

# Global reconstruction (iii)

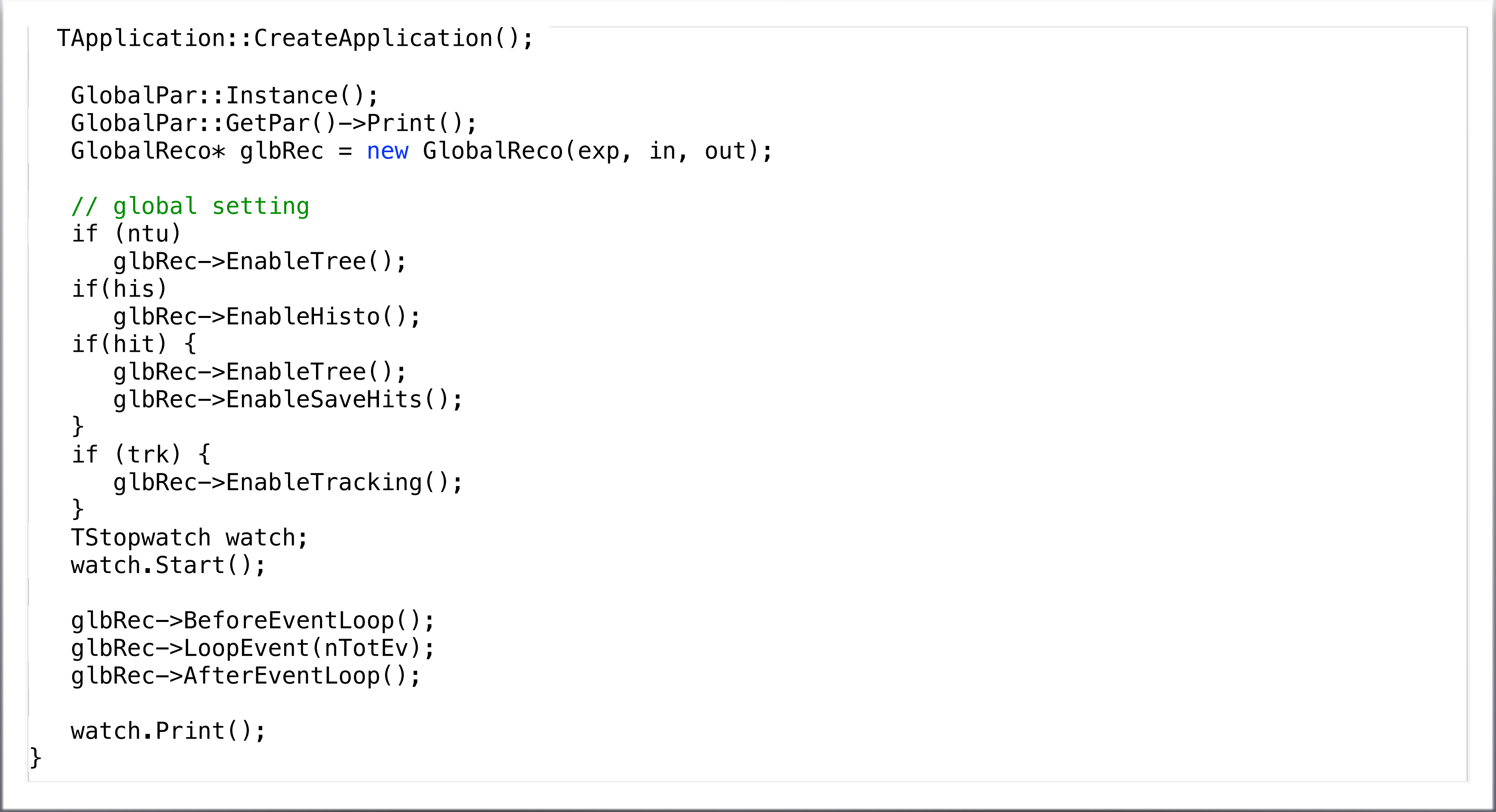
- Executable: DecodeGlob(MC)



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

   TString in("");
   TString exp("");

   Int_t pos = in.Last('.');
   TString out = in(0, pos);
   out.Append("_Out.root");

   Bool_t ntu = false;
   Bool_t his = false;
   Bool_t hit = false;
   Bool_t trk = false;
   Int_t nTotEv = 1e7;

   for (int i = 0; i < argc; i++){
      if(strcmp(argv[i],"-out") == 0)   { out =TString(argv[++i]);  }   // Raw file name for output
      if(strcmp(argv[i],"-in") == 0)    { in = TString(argv[++i]);  }   // Root file in input
      if(strcmp(argv[i],"-exp") == 0)   { exp = TString(argv[++i]); }   // extention for config/geomap files
      if(strcmp(argv[i],"-nev") == 0)   { nTotEv = atoi(argv[++i]); }   // Number of events to be analized
      if(strcmp(argv[i],"-ntu") == 0)   { ntu = true;   } // enable tree filling
      if(strcmp(argv[i],"-his") == 0)   { his = true;   } // enable histograming
      if(strcmp(argv[i],"-hit") == 0)   { hit = true;   } // enable hits saving
      if(strcmp(argv[i],"-trk") == 0)   { trk = true;   } // enable tracking action
      if(strcmp(argv[i],"-help") == 0)  {
         cout<<" Decoder help:"<<endl;
         cout<<" Ex: Decoder [opts] "<<endl;
         cout<<" possible opts are:"<<endl;
         cout<<"      -in path/file  : [def=""] raw input file"<<endl;
         cout<<"      -out path/file : [def=*_Out.root] Root output file"<<endl;
         cout<<"      -nev value     : [def=10^7] Numbers of events to process"<<endl;
         cout<<"      -exp name      : [def=""] experient name for config/geomap extention"<<endl;
         cout<<"      -trk           : enable tracking actions"<<endl;
         cout<<"      -hit           : enable saving hits in tree (activated ntu option)"<<endl;
         cout<<"      -ntu           : enable tree filling"<<endl;
         cout<<"      -his           : enable crtl histograming"<<endl;
         return 1;
      }
   }…
```

# Global reconstruction (iii)

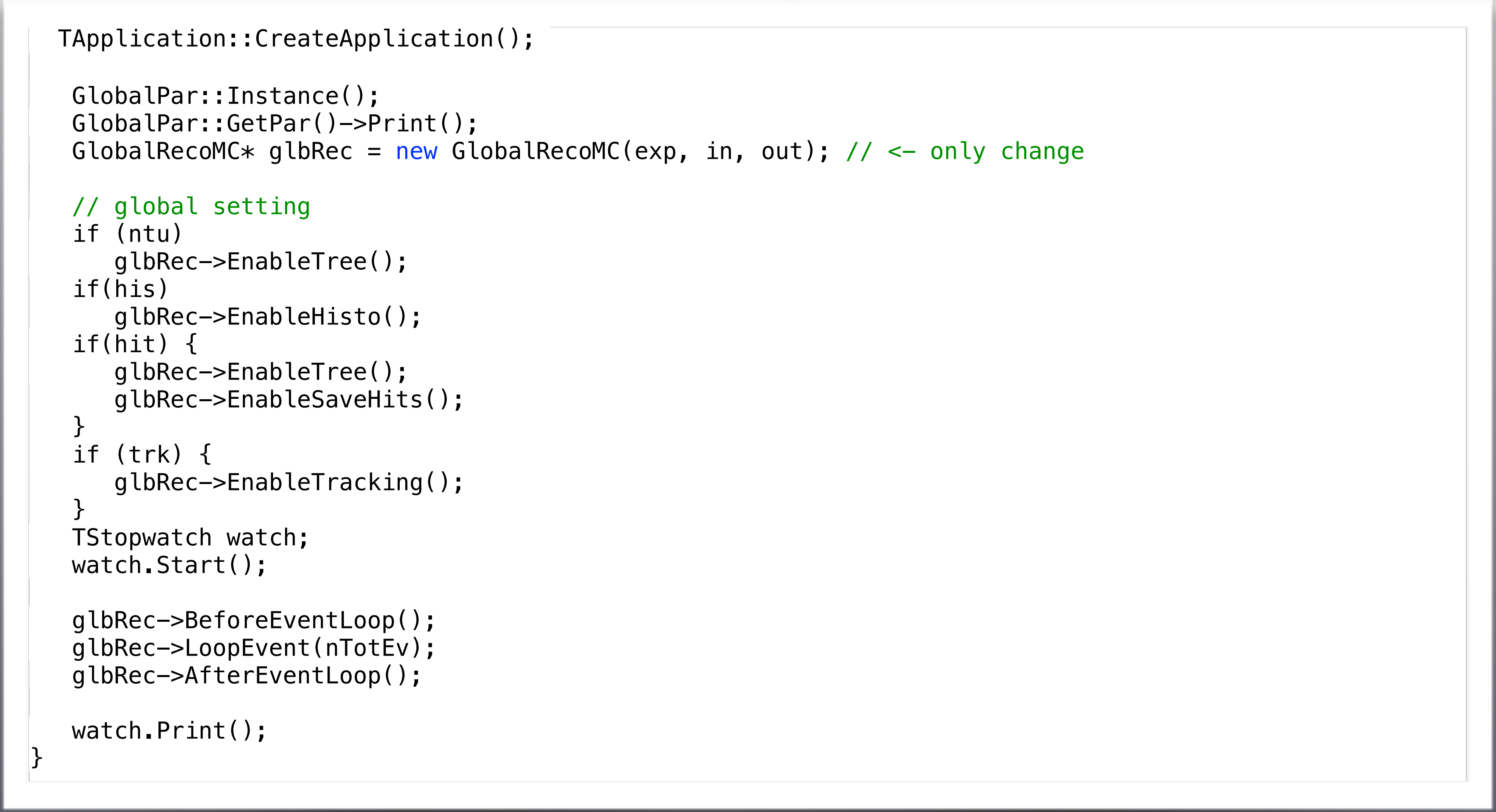
- Executable: DecodeGlob



```
  TApplication::CreateApplication();

   GlobalPar::Instance();
   GlobalPar::GetPar()->Print();
   GlobalReco* glbRec = new GlobalReco(exp, in, out);

   // global setting
   if (ntu)
      glbRec->EnableTree();
   if(his)
      glbRec->EnableHisto();
   if(hit) {
      glbRec->EnableTree();
      glbRec->EnableSaveHits();
   }
   if (trk) {
      glbRec->EnableTracking();
   }
   TStopwatch watch;
   watch.Start();

   glbRec->BeforeEventLoop();
   glbRec->LoopEvent(nTotEv);
   glbRec->AfterEventLoop();

   watch.Print();
}
```

# MC Global reconstruction (iii)

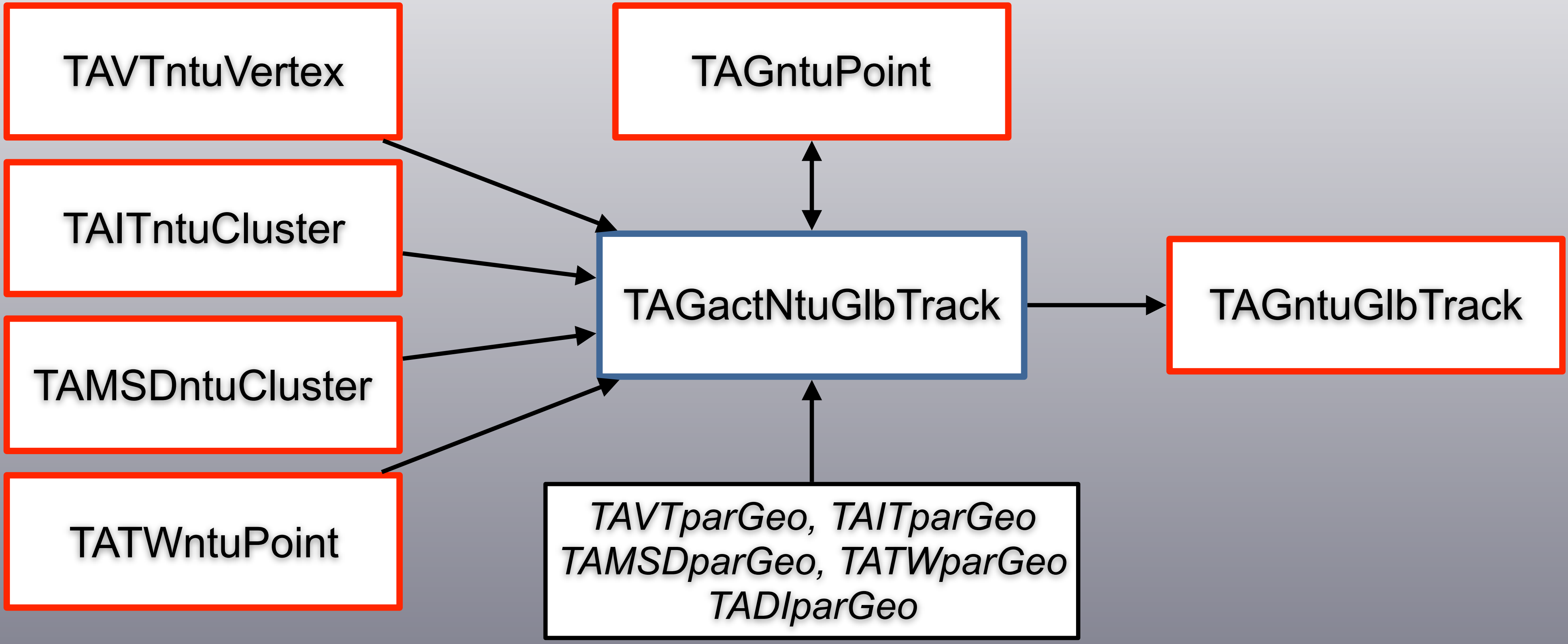
- Executable: DecodeGlobMC



```
  TApplication::CreateApplication();

   GlobalPar::Instance();
   GlobalPar::GetPar()->Print();
   GlobalRecoMC* glbRec = new GlobalRecoMC(exp, in, out); // <- only change

   // global setting
   if (ntu)
      glbRec->EnableTree();
   if(his)
      glbRec->EnableHisto();
   if(hit) {
      glbRec->EnableTree();
      glbRec->EnableSaveHits();
   }
   if (trk) {
      glbRec->EnableTracking();
   }
   TStopwatch watch;
   watch.Start();

   glbRec->BeforeEventLoop();
   glbRec->LoopEvent(nTotEv);
   glbRec->AfterEventLoop();

   watch.Print();
}
```

# Global Reconstruction Action (i)

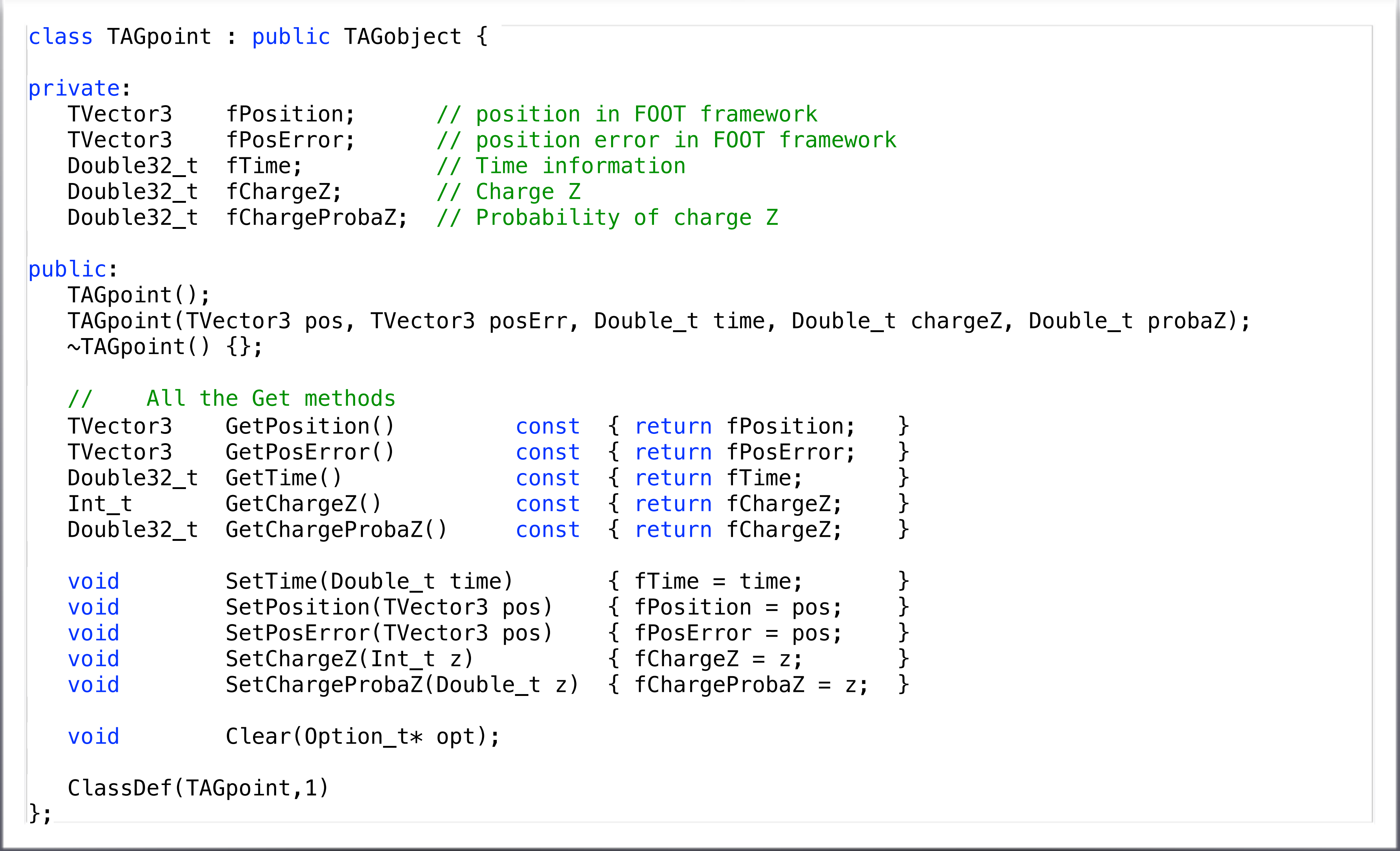
Scheme:



- Containers as input from VTX, IT, MSD and TW
- Container as output global tracks
- Intermediate containers TAGntuPoint

# Global Reconstruction Action (ii)

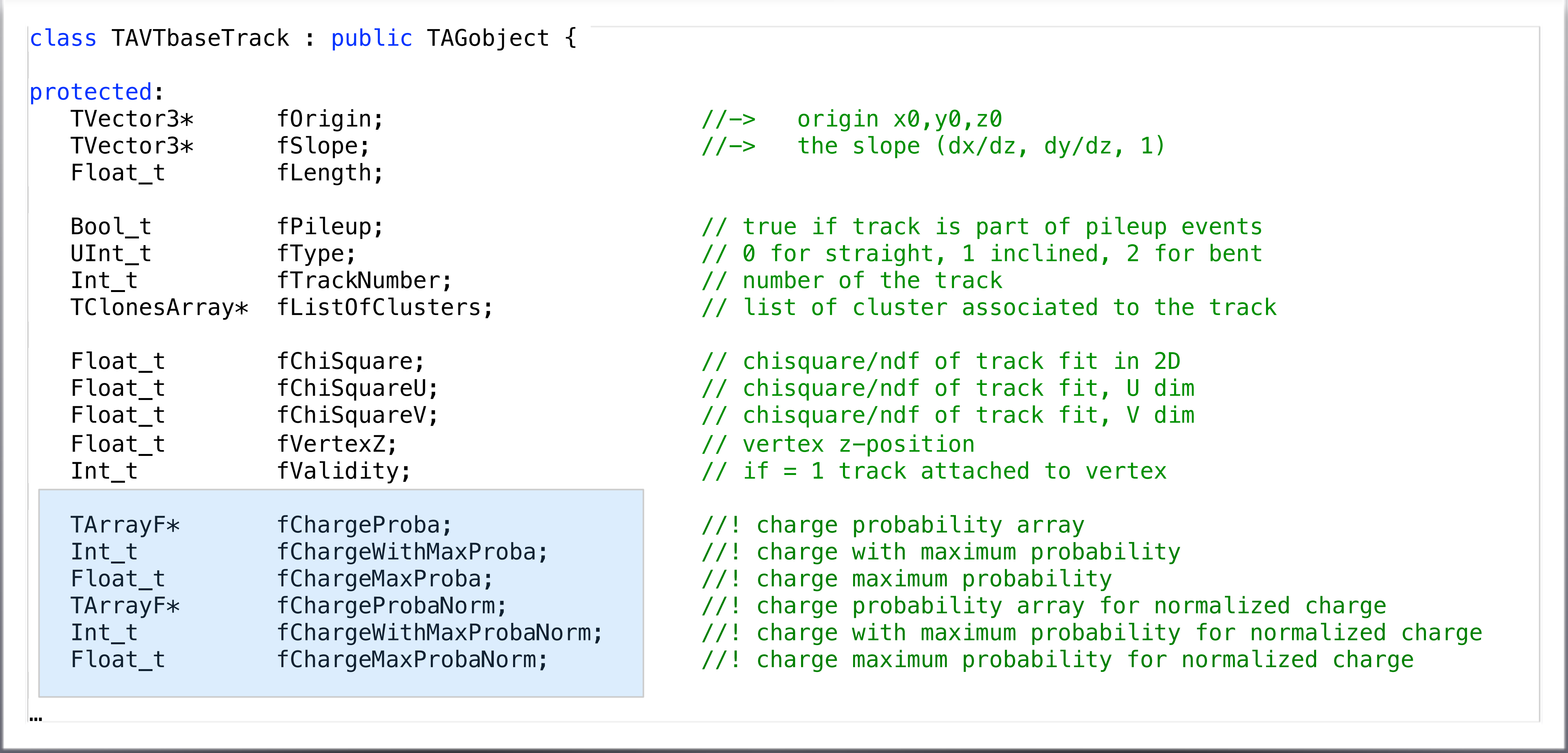
- TAGpoint:



```
class TAGpoint : public TAGobject {

private:
   TVector3    fPosition;      // position in FOOT framework
   TVector3    fPosError;      // position error in FOOT framework
   Double32_t  fTime;          // Time information
   Double32_t  fChargeZ;       // Charge Z
   Double32_t  fChargeProbaZ;  // Probability of charge Z

public:
   TAGpoint();
   TAGpoint(TVector3 pos, TVector3 posErr, Double_t time, Double_t chargeZ, Double_t probaZ);
   ~TAGpoint() {};

   //    All the Get methods
   TVector3    GetPosition()        const  { return fPosition;   }
   TVector3    GetPosError()        const  { return fPosError;   }
   Double32_t  GetTime()            const  { return fTime;       }
   Int_t       GetChargeZ()         const  { return fChargeZ;    }
   Double32_t  GetChargeProbaZ()    const  { return fChargeZ;    }

   void        SetTime(Double_t time)       { fTime = time;       }
   void        SetPosition(TVector3 pos)    { fPosition = pos;    }
   void        SetPosError(TVector3 pos)    { fPosError = pos;    }
   void        SetChargeZ(Int_t z)          { fChargeZ = z;       }
   void        SetChargeProbaZ(Double_t z)  { fChargeProbaZ = z;  }

   void        Clear(Option_t* opt);

   ClassDef(TAGpoint,1)
};
```

# Global Reconstruction Action (iii)

- Vertex tracks:

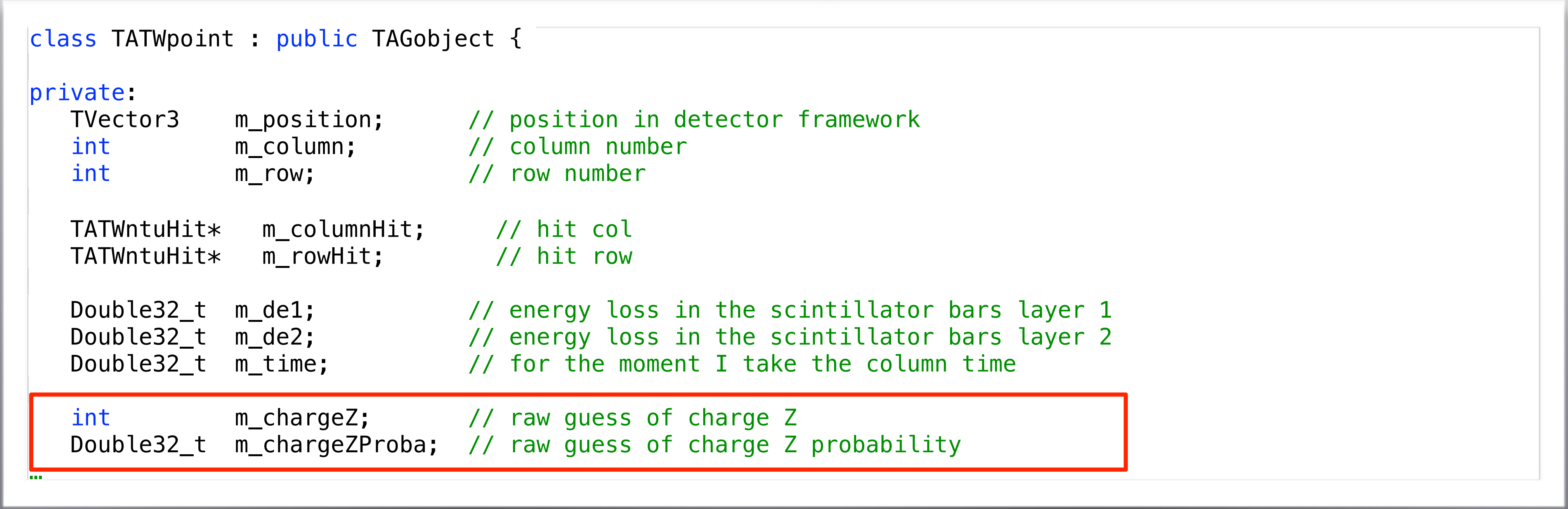


```
class TAVTbaseTrack : public TAGobject {

protected:
   TVector3*      fOrigin;                       //->   origin x0,y0,z0
   TVector3*      fSlope;                        //->   the slope (dx/dz, dy/dz, 1)
   Float_t        fLength;

   Bool_t         fPileup;                       // true if track is part of pileup events
   UInt_t         fType;                         // 0 for straight, 1 inclined, 2 for bent
   Int_t          fTrackNumber;                  // number of the track
   TClonesArray*  fListOfClusters;               // list of cluster associated to the track

   Float_t        fChiSquare;                    // chisquare/ndf of track fit in 2D
   Float_t        fChiSquareU;                   // chisquare/ndf of track fit, U dim
   Float_t        fChiSquareV;                   // chisquare/ndf of track fit, V dim
   Float_t        fVertexZ;                      // vertex z-position
   Int_t          fValidity;                     // if = 1 track attached to vertex

   TArrayF*       fChargeProba;                  //! charge probability array
   Int_t          fChargeWithMaxProba;           //! charge with maximum probability
   Float_t        fChargeMaxProba;               //! charge maximum probability
   TArrayF*       fChargeProbaNorm;              //! charge probability array for normalized charge
   Int_t          fChargeWithMaxProbaNorm;       //! charge with maximum probability for normalized charge
   Float_t        fChargeMaxProbaNorm;           //! charge maximum probability for normalized charge

…
```

- Possibility to have information about atomic charge Z (TAVTparCal)
- Need a calibration run

# Global Reconstruction Action (iv)

- Time of Flight point:

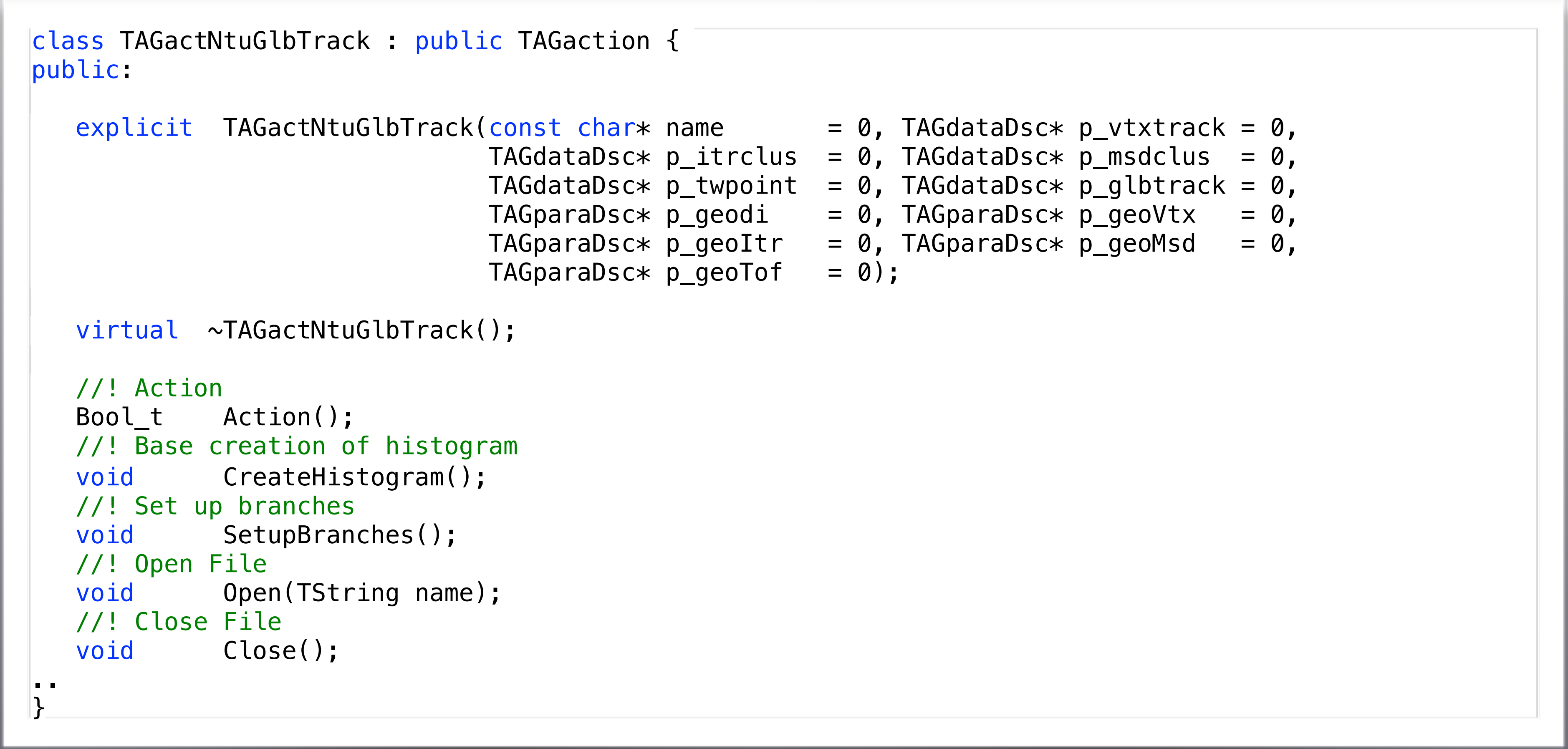


```
class TATWpoint : public TAGobject {

private:
   TVector3    m_position;      // position in detector framework
   int         m_column;        // column number
   int         m_row;           // row number

   TATWntuHit*   m_columnHit;     // hit col
   TATWntuHit*   m_rowHit;        // hit row

   Double32_t  m_de1;           // energy loss in the scintillator bars layer 1
   Double32_t  m_de2;           // energy loss in the scintillator bars layer 2
   Double32_t  m_time;          // for the moment I take the column time

   int         m_chargeZ;       // raw guess of charge Z
   Double32_t  m_chargeZProba;  // raw guess of charge Z probability
…
```

- We NEED a atomic charge identification in TW within a magnetic field: (Marco)

# Global Reconstruction Action (v)

- TAGactNtuGlbTrack: action (i)



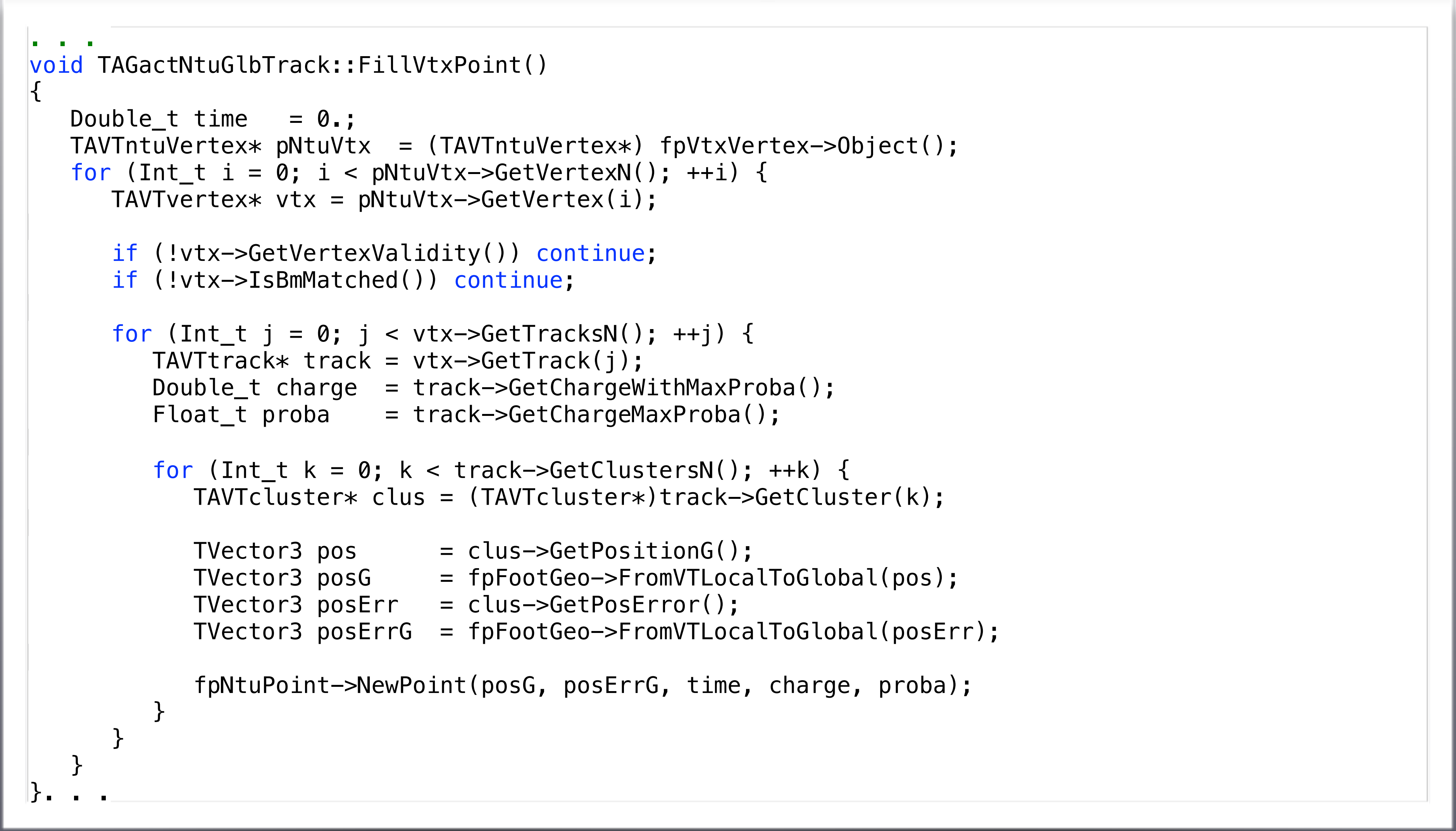
```
class TAGactNtuGlbTrack : public TAGaction {
public:

   explicit  TAGactNtuGlbTrack(const char* name       = 0, TAGdataDsc* p_vtxtrack = 0,
                               TAGdataDsc* p_itrclus  = 0, TAGdataDsc* p_msdclus  = 0,
                               TAGdataDsc* p_twpoint  = 0, TAGdataDsc* p_glbtrack = 0,
                               TAGparaDsc* p_geodi    = 0, TAGparaDsc* p_geoVtx   = 0,
                               TAGparaDsc* p_geoItr   = 0, TAGparaDsc* p_geoMsd   = 0,
                               TAGparaDsc* p_geoTof   = 0);

   virtual  ~TAGactNtuGlbTrack();

   //! Action
   Bool_t    Action();
   //! Base creation of histogram
   void      CreateHistogram();
   //! Set up branches
   void      SetupBranches();
   //! Open File
   void      Open(TString name);
   //! Close File
   void      Close();
..
}
```

- Possibility to read back TTree (fgStdAloneFlag option)

➡If needed, another global action could be implemented (with a switch in GlobalPar)

# Global Reconstruction Action (vi)

- TAGactNtuGlbTrack: action (ii)

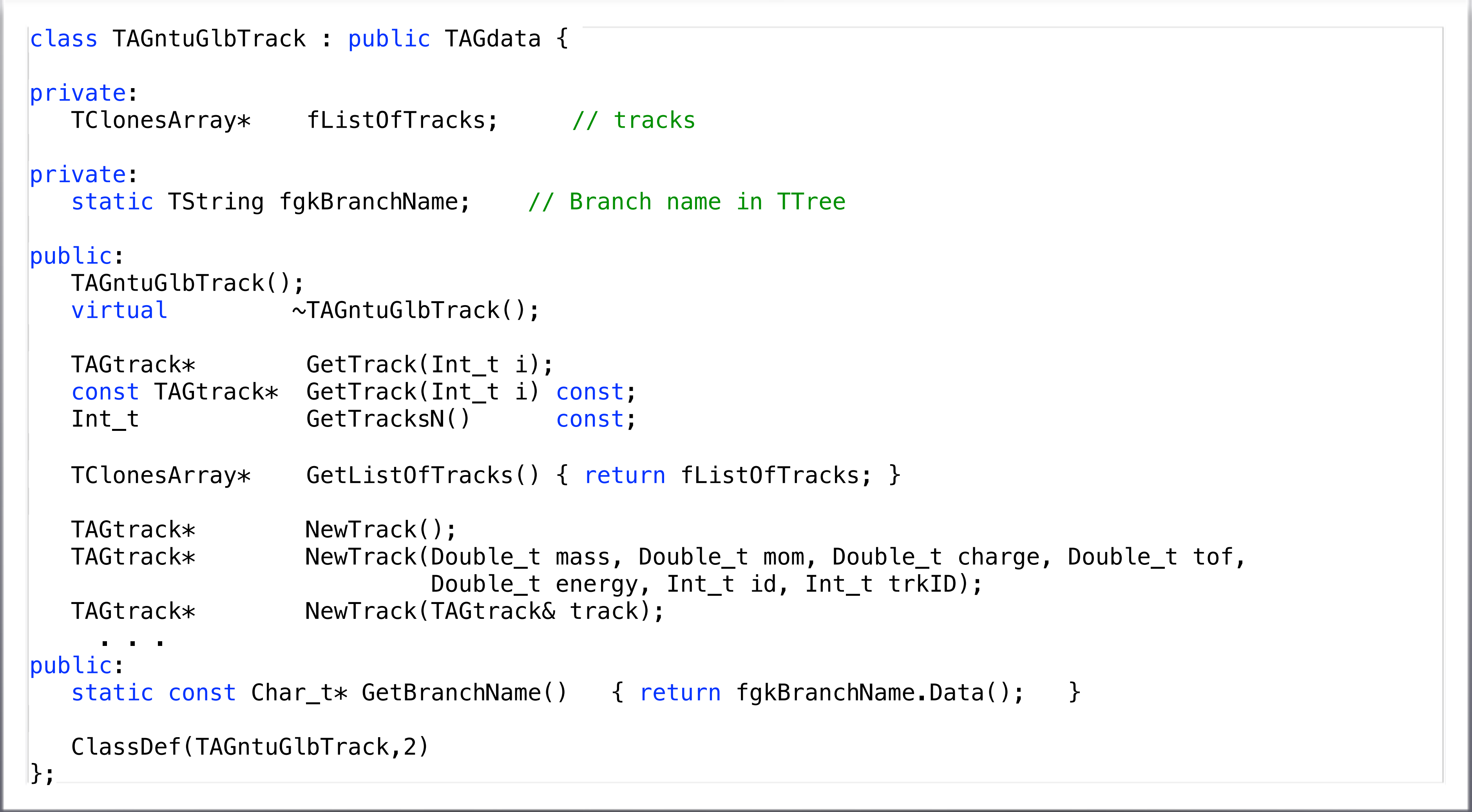


```
. . .
void TAGactNtuGlbTrack::FillVtxPoint()
{
   Double_t time    = 0.;
   TAVTntuVertex* pNtuVtx  = (TAVTntuVertex*) fpVtxVertex->Object();
   for (Int_t i = 0; i < pNtuVtx->GetVertexN(); ++i) {
      TAVTvertex* vtx = pNtuVtx->GetVertex(i);

      if (!vtx->GetVertexValidity()) continue;
      if (!vtx->IsBmMatched()) continue;

      for (Int_t j = 0; j < vtx->GetTracksN(); ++j) {
         TAVTtrack* track = vtx->GetTrack(j);
         Double_t charge  = track->GetChargeWithMaxProba();
         Float_t proba    = track->GetChargeMaxProba();

         for (Int_t k = 0; k < track->GetClustersN(); ++k) {
            TAVTcluster* clus = (TAVTcluster*)track->GetCluster(k);

            TVector3 pos      = clus->GetPositionG();
            TVector3 posG     = fpFootGeo->FromVTLocalToGlobal(pos);
            TVector3 posErr   = clus->GetPosError();
            TVector3 posErrG  = fpFootGeo->FromVTLocalToGlobal(posErr);

            fpNtuPoint->NewPoint(posG, posErrG, time, charge, proba);
         }
      }
   }
}. . .
```

- For the moment only filling TAGpoint

# Global Reconstruction Action (vii)

- TAGntuGlbTrack: container

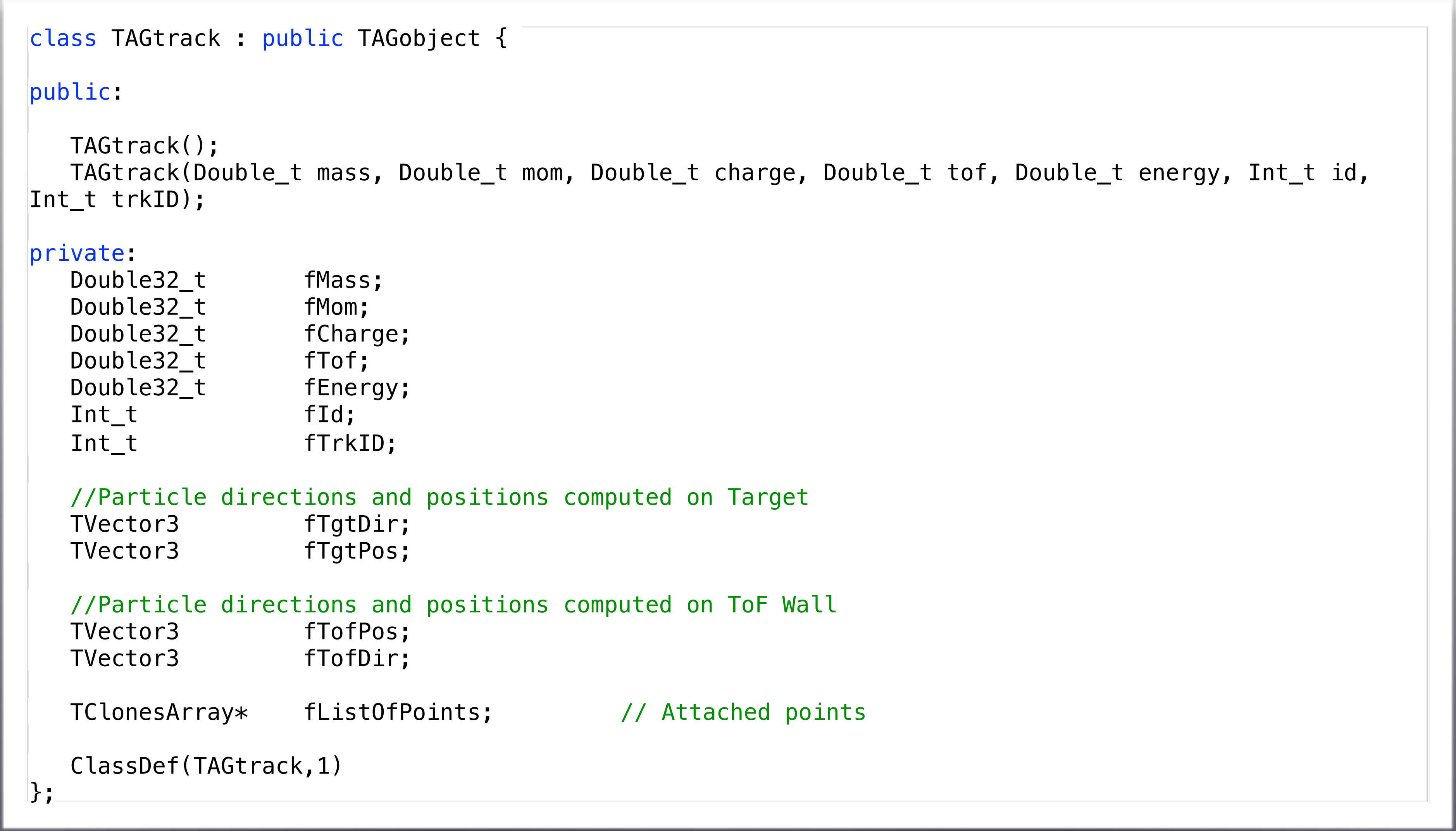


```
class TAGntuGlbTrack : public TAGdata {

private:
   TClonesArray*    fListOfTracks;     // tracks

private:
   static TString fgkBranchName;     // Branch name in TTree

public:
   TAGntuGlbTrack();
   virtual         ~TAGntuGlbTrack();

   TAGtrack*        GetTrack(Int_t i);
   const TAGtrack*  GetTrack(Int_t i) const;
   Int_t            GetTracksN()      const;

   TClonesArray*    GetListOfTracks() { return fListOfTracks; }

   TAGtrack*        NewTrack();
   TAGtrack*        NewTrack(Double_t mass, Double_t mom, Double_t charge, Double_t tof,
                             Double_t energy, Int_t id, Int_t trkID);
   TAGtrack*        NewTrack(TAGtrack& track);
     . . .
public:
   static const Char_t* GetBranchName()   { return fgkBranchName.Data();   }

   ClassDef(TAGntuGlbTrack,2)
};
```

- Class adopted from FIRST

# Global Reconstruction Action (viii)

- TAGtrack:



```
class TAGtrack : public TAGobject {

public:

   TAGtrack();
   TAGtrack(Double_t mass, Double_t mom, Double_t charge, Double_t tof, Double_t energy, Int_t id,
Int_t trkID);

private:
   Double32_t       fMass;
   Double32_t       fMom;
   Double32_t       fCharge;
   Double32_t       fTof;
   Double32_t       fEnergy;
   Int_t            fId;
   Int_t            fTrkID;

   //Particle directions and positions computed on Target
   TVector3         fTgtDir;
   TVector3         fTgtPos;

   //Particle directions and positions computed on ToF Wall
   TVector3         fTofPos;
   TVector3         fTofDir;

   TClonesArray*    fListOfPoints;         // Attached points

   ClassDef(TAGtrack,1)
};
```

  - Class adopted from FIRST

# Strategy (ia)
## (adopted from FIRST)

- Foreward approach:
  1. Start from VTX track and a point detected in TW with atomic charge Z, assuming
     a. A = Zx2 (except for Z = 1 thus  A = 1)
     b. $E_c = E_{beam}/A$
  2. Extrapole to IT, search for closest cluster, if not go back to 1, changing VTX track
  3. Extrapole to MSD, combine with a local track
  4. Extrapole to TW, search for the closest point
  5. Process the Kalman filter
  6. Using next hit in TW with atomic charge Z’, go to 1

# Conclusions

- New framework for Global reconstruction relying on local reconstruction
- Strategy relies on FIRST experiment with given requirements

  Urgent: need Z identification in TW (Marco)  and tracks in MSD (Gianluigi ?)
- New Kalman filter

➡ Start to study a “real” global reconstruction efficiency