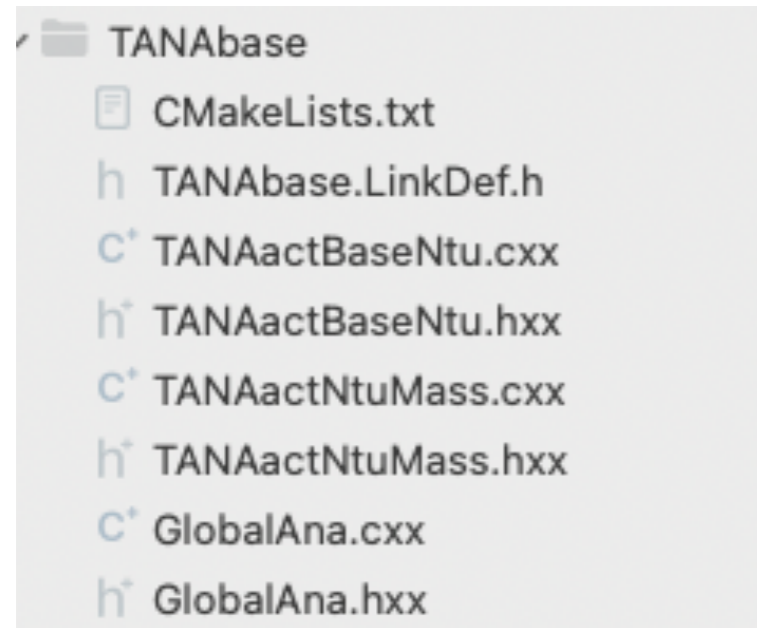


Analysis Framework

New Classes

□ Analysis folder:

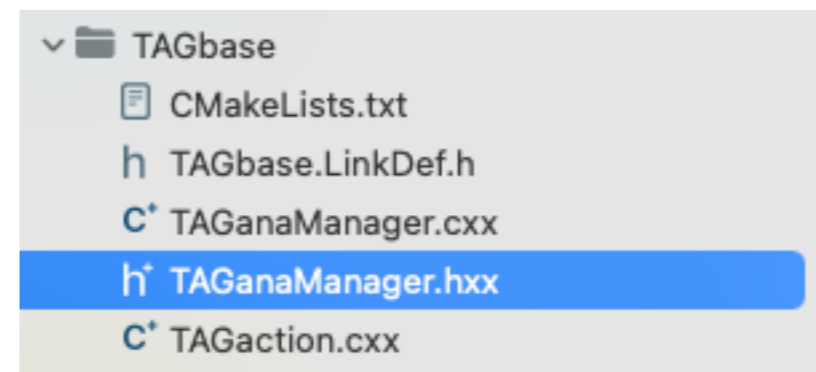
- Folder added:



- Base class for analysis: TANAactBaseNtu
- Example of analysis class: TANAactNtuMass
- Master class managing analysis: GlobalAna

□ Library folder:

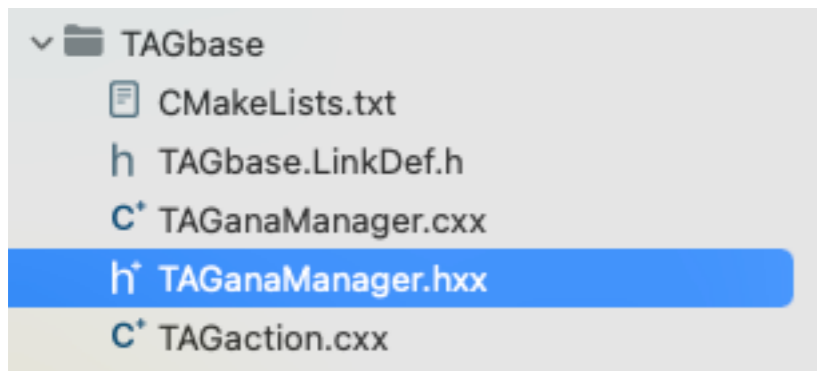
- New analysis manager class:



Analysis Manager

Library folder:

- New analysis manager class:



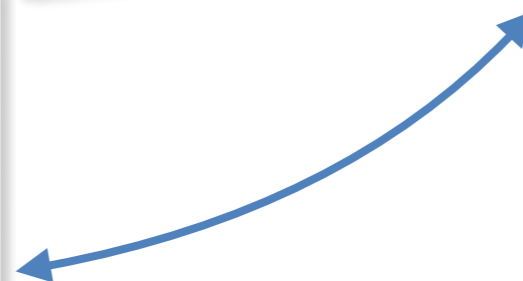
- ./config/expname/TANAdetector.cfg

```
// -+-----+
// -+-----+
//
// This is a Configuration File for FOOT analysis
//
// Campaign GSI2021
//
// -+-----+
// Parameters for Analysis
// -+-----+
MassReso:    1
PtReso:      0
Efficiency:  0
```

```
class TAGanaManager : public TAGparTools {
protected:
    TString      fkDefaultName; ///< default analysis file

    /*!
     \struct AnalysisParameter_t
     \brief Analysis parameters
     */
    struct GlbAnaParameter_t : public TObject {
        Bool_t    MassResoFlag;          ///< Mass resolution flag
        Bool_t    PtResoFlag;           ///< Momentum resolution flag
        Bool_t    EfficiencyFlag;       ///< Efficiency studies flag
    };

    GlbAnaParameter_t fAnalysisParameter; ///< analysis parameters
```



- ➔ Name of the different analysis modules
- ➔ Could add as many needed

- ➔ File only on GSI2021 and CNAO2023

Analysis Class (i)

□ Base analysis class: TANAactBaseNtu

```
class TANAactBaseNtu : public TAGaction {  
  
public:  
    explicit TANAactBaseNtu(const char* name          = 0,  
                           TAGdataDsc* p_irntutrack = 0,  
                           TAGparaDsc* p_geomap     = 0);  
  
    virtual ~TANAactBaseNtu();  
  
    // Create histograms  
    virtual void CreateHistogram() { return; }  
  
    // Action  
    virtual Bool_t Action()          { return true; }  
  
private:  
    TAGdataDsc*   fpNtuTrack;          ///< input global tracks  
    TAGgeoTrafo*  fpFootGeo;          ///< First geometry transformer  
    TAGparaDsc*   fpGeoMapG;          ///< geometry para dsc  
  
    Float_t       fBeamEnergyTarget;  ///< Beam energy at target  
    Float_t       fBeamEnergy;        ///< Beam energy  
    Float_t       fBeamA;             ///< Beam atomic mass  
    Float_t       fBeamZ;            ///< Beam atomic number  
    ...  
}
```

- Contains the global track container, target/beam and FOOT geometry
- Histograms creation and action method are virtual

Analysis Class (ii)

□ Analysis class template: TANAactNtuMass

```
class TANAactNtuMass : public TANAactBaseNtu {  
  
public:  
    explicit TANAactNtuMass(const char* name          = 0,  
                           TAGdataDsc* p_irntutrack = 0,  
                           TAGparaDsc* p_geomap      = 0);  
  
    virtual ~TANAactNtuMass();  
  
    // Create histograms  
    void CreateHistogram();  
  
    // Action  
    Bool_t Action();  
  
    ClassDef(TANAactNtuMass, 0)  
};
```

- ➔ Inherits from base class
- ➔ Could add as much containers and descriptors needed
- ➔ Will save automatically the created histograms
(not foreseen saving in a tree, but could be done by passing the tree of TAGactTreeWriter as argument)

Global Analysis Class (i)

Global analysis class: GlobalAna

```
class GlobalAna : public TNamed // using TNamed for the in/out files
{
public:
    // default constructor
    GlobalAna(TString expName, Int_t runNumber, TString fileNameIn, TString fileNameout, Bool_t isMC = false);

    // default destructor
    virtual ~GlobalAna();

    // Read parameters
    void ReadParFiles();

    // Create raw action
    virtual void CreateAnaAction();

    // Add required items
    virtual void AddRequiredItem();

    // Set histogram directory
    virtual void SetHistogramDir();

    // Loop events
    virtual void LoopEvent(Int_t nEvents);

    // Begin loop
    virtual void BeforeEventLoop();

    // End loop
    virtual void AfterEventLoop();

    // Open File Out
    virtual void OpenFileOut();

    // Close File Out
    virtual void CloseFileOut();

    // Create L0 branch in tree
    virtual void SetTreeBranches();

    // Goto Event
    virtual Bool_t GoEvent(Int_t iEvent);
};
```

→ Base on the structure of BaseReco class

Global Analysis Class (ii)

Global analysis class: GlobalAna

```
//  
//! Read parameters files  
void GlobalAna::ReadParFiles()  
{  
    Int_t Z_beam = 0;  
    Int_t A_beam = 0;  
    TString ion_name;  
    Float_t kinE_beam = 0.;  
  
    // Read Trafo file  
    TString parFileName = fCampManager->GetCurGeoFile(FootBaseName("TAGgeoTrafo"), fRunNumber);  
    fpFootGeo->FromFile(parFileName);  
  
    . . .  
  
    // initialise par files for start counter  
    if (TAGrecoManager::GetPar()->IncludeST() || TAGrecoManager::GetPar()->IncludeTW() || TAGrecoManager::GetPar()->IncludeCA()) {  
        fpParGeoSt = new TAGparaDsc(new TASTparGeo());  
        TASTparGeo* parGeo = (TASTparGeo*)fpParGeoSt->Object();  
        TString parFileName = fCampManager->GetCurGeoFile(FootBaseName("TASTparGeo"), fRunNumber);  
        parGeo->FromFile(parFileName.Data());  
  
        fpParConfSt = new TAGparaDsc(new TASTparConf());  
        TASTparConf* parConf = (TASTparConf*)fpParConfSt->Object();  
        parFileName = fCampManager->GetCurConfFile(FootBaseName("TASTparGeo"), fRunNumber);  
        parConf->FromFile(parFileName.Data());  
    }  
  
    . . .  
}
```

➔ Read all geomaps/config files for all included detectors

Global Analysis Class (iii)

□ Global analysis class: GlobalAna

```
//  
//! Set L0 tree branches for reading back  
void GlobalAna::SetTreeBranches()  
{  
    const Char_t* name = FootActionDscName("TAGactTreeReader");  
    fActEvtReader = new TAGactTreeReader(name);  
  
    if (TAGrecoManager::GetPar()->IncludeST()) {  
        fpNtuHitSt = new TAGdataDsc(new TASTntuHit());  
        fActEvtReader->SetupBranch(fpNtuHitSt);  
  
        if (fFlagMC) {  
            fpNtuMcSt = new TAGdataDsc(FootDataDscMcName(kST), new TAMCntuHit());  
            fActEvtReader->SetupBranch(fpNtuMcSt, FootBranchMcName(kST));  
        }  
    }  
  
    if (TAGrecoManager::GetPar()->IncludeBM()) {  
        fpNtuTrackBm = new TAGdataDsc(new TABMntuTrack());  
        fActEvtReader->SetupBranch(fpNtuTrackBm);  
        if (fFlagMC) {  
            fpNtuMcBm = new TAGdataDsc(FootDataDscMcName(kBM), new TAMCntuHit());  
            fActEvtReader->SetupBranch(fpNtuMcBm, FootBranchMcName(kBM));  
        }  
    }  
    ...  
}
```

➔ Read all containers (excluded all hits except for ST) for the included detectors

Global Analysis Class (iv)

□ Global analysis class: GlobalAna

```
//  
//! Create reconstruction actions  
void GlobalAna::CreateAnaAction()  
{  
    // place here your beloved analysis class  
    if ((TAGrecoManager::GetPar()->IncludeTOE() || TAGrecoManager::GetPar()->IncludeKalman())) {  
        if (fAnaManager->GetAnalysisPar().MassResoFlag)  
            fActGlbAna = new TANAactNtuMass("anaActMass", fpNtuGlbTrack, fpParGeoG);  
    }  
}  
  
//  
//! Add required reconstruction actions in list  
void GlobalAna::AddRequiredItem()  
{  
    // Add the required analysis class  
    if ((TAGrecoManager::GetPar()->IncludeTOE() || TAGrecoManager::GetPar()->IncludeKalman())) {  
        if (fAnaManager->GetAnalysisPar().MassResoFlag)  
            gTAGroot->AddRequiredItem("anaActMass");  
    }  
}
```

- ➔ Create and require the dedicated class analysis when flag is on in the analysis manager
- ➔ Can have more than one analysis module

Global Analysis Class (ivb)

Global analysis class: example

```
//  
//! Create reconstruction actions  
void GlobalAna::CreateAnaAction()  
{  
  
    if (fAnaManager->GetAnalysisPar().AnalysisCuts)  
    {  
        fActGlbCuts = new TANAactNtuSelectionCuts(fTrkCutsMap, fEventCutsMap, "anaActCuts")  
        fActGlbTrackCounts = new TANAactNtuGlbTrackCounts(fTrackGlbCounts_reco, fTrackGlbCounts)  
  
        if (fFlagMC){  
            fActMCref = new TANAactNtuMCref(fTrackMCrefMap, fEventMCrefMap, fTrackMCrefCounts, "anaActMCref")  
        }  
  
        fActCrossSection = new TANAactCrossSection("anaActCrossSection", fFlagMC, fpNtuGlbTrackCounts, fActGlbTrackCounts)  
        fActCrossSection->CreateHistogram();  
    }  
}
```

- ➔ Presented at last Analysis & tracking [meeting](#)
- ➔ Interfaced with RDataFrame tools

Global Analysis executable

Global analysis main: DecoceGlbAnalysis

```
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 analyzed
    if(strcmp(argv[i], "-nsk") == 0) { nSkipEv = atoi(argv[++i]); } // Number of events to be skip
    if(strcmp(argv[i], "-run") == 0) { runNb = atoi(argv[++i]); } // Run Number
    if(strcmp(argv[i], "-mc") == 0) { mc = true; } // reco from MC local reco data

    . . .
}

TApplication::CreateApplication();

TAGrecoManager::Instance(exp);
TAGrecoManager::GetPar()->FromFile();
TAGrecoManager::GetPar()->Print();

// check input file exists
if(in.IsNull() || gSystem->AccessPathName(in.Data())) {
    Error("main()", "Input file does not exist or is null");
    exit(-1);
}

GlobalAna* glbAna = new GlobalAna(exp, runNb, in, out, mc);
. . .
}
```

➔ Copy of the DecodeGlb main, take same arguments

```
DecodeGlbAnalysis -in run4287_GlbS_70kEvts_Out.root -out MassAnalysis.root -exp GSI2021 -run 4287
```

Conclusions

- New analysis framework
- A template analysis class

- Implement now dedicated analysis class and update the analysis manager accordingly