Style Fixing

Coding conventions

Style

- Tabulation indentation:
  - Depending on the editor, tabulation are not equivalent to the same number of blanks

Style

- Tabulation indentation:
  - Depending on the editor, tabulation are not equivalent to the same number of blanks

```
{
  stringstream ss;

  if(TAGrecoManager::GetPar()->IncludeST()){

    TAGgeoTrafo* fpFootGeo = (TAGgeoTrafo*)gTAGroot->FindAction(TAGgeoTrafo::GetDefaultActName().Data());

    TVector3 center = fpFootGeo->GetSTCenter();
    TVector3  angle = fpFootGeo->GetSTAngles(); //invert the angles to take into account the FLUKA convention
    angle *= -1;

    if(angle.Mag()!=0){
       ss << PrintCard("ROT-DEFI", "300.", "", "",
             Form("%f",-center.X()), Form("%f",-center.Y()),
             Form("%f",-center.Z()), "st") << endl;
       if(angle.X()!=0)
    ss << PrintCard("ROT-DEFI", "100.", "", Form("%f",angle.X()),"", "", "", "st") << endl;
       if(angle.Y()!=0)
    ss << PrintCard("ROT-DEFI", "200.", "", Form("%f",angle.Y()),"", "", "", "st") << endl;
       if(angle.Z()!=0)
    ss << PrintCard("ROT-DEFI", "300.", "", Form("%f",angle.Z()),"", "", "", "st") << endl;
       ss << PrintCard("ROT-DEFI", "300.", "", "",
             Form("%f",center.X()), Form("%f",center.Y()),
             Form("%f",center.Z()), "st") << endl;

    }
  }

  return ss.str();
}
```

➡ Code is no more aligned

Style

- Different style for brackets
  - Breaking or not for method, condition, etc…

Style

- Different style for brackets
  - Breaking or not for method, condition, etc…

```
// initialise par files for vertex
if (TAGrecoManager::GetPar()->IncludeVT()) {
   fpParGeoVtx = new TAGparaDsc(new TAVTparGeo());
   TAVTparGeo* parGeo = (TAVTparGeo*)fpParGeoVtx->Object();
   TString parFileName = fCampManager->GetCurGeoFile(FootBaseName("TAVTparGeo"), fRunNumber);
   parGeo->FromFile(parFileName.Data());

   fpParConfVtx = new TAGparaDsc(new TAVTparConf());
   TAVTparConf* parConf = (TAVTparConf*)fpParConfVtx->Object();
   parFileName = fCampManager->GetCurConfFile(FootBaseName("TAVTparGeo"), fRunNumber);
   parConf->FromFile(parFileName.Data());

fVtxTrackingAlgo = parConf->GetAnalysisPar().TrackingAlgo;
fVtxFlagPostTrack = parConf->GetAnalysisPar().PostTrackingFlag;
```

Style

- Different style for brackets
  - Breaking or not for method, condition, etc…

```
// initialise par files for vertex
if (TAGrecoManager::GetPar()->IncludeVT() ) {
   fpParGeoVtx = new TAGparaDsc(new TAVTparGeo());
   TAVTparGeo* parGeo = (TAVTparGeo*)fpParGeoVtx->Object();
   TString parFileName = fCampManager->GetCurGeoFile(FootBaseN
   parGeo->FromFile(parFileName.Data());

   fpParConfVtx = new TAGparaDsc(new TAVTparConf());
   TAVTparConf* parConf = (TAVTparConf*)fpParConfVtx->Object()
   parFileName = fCampManager->GetCurConfFile(FootBaseName("TA
   parConf->FromFile(parFileName.Data());

fVtxTrackingAlgo = parConf->GetAnalysisPar().TrackingAlgo;
fVtxFlagPostTrack = parConf->GetAnalysisPar().PostTrackingF
```

```
if( TAGrecoManager::GetPar()->IncludeVT() )
{
  vtxparGeo = (TAVTparGeo*) fpVDparGeo->Object();
  vtxNtuHit = (TAVTntuHit*) fpVTNtuHit->Object();
  vtxNtuCluster = (TAVTntuCluster*) fpVTNtuCluster->Object();
  vtxNtuVertex = (TAVTntuVertex*) fpVTNtuVertex->Object();
  vtNtuTrack = (TAVTntuTrack*) fpVTNtuTrack->Object();
}

if( TAGrecoManager::GetPar()->IncludeIT() ) {
  itparGeo = (TAITparGeo*) fpITparGeo->Object();
  itNtuCluster = (TAITntuCluster*) fpITNtuCluster->Object();
}

if( TAGrecoManager::GetPar()->IncludeMSD() )
{
  msdparGeo = (TAMSDparGeo*) fpMSDparGeo->Object();
  msdNtuPoint = (TAMSDntuPoint*) fpMSDNtuPoint->Object();
  msdNtuCluster = (TAMSDntuCluster*) fpMSDNtuCluster->Object();
}
```

Style

- Different style for brackets
  - Breaking or not for method, condition, etc…

```
// initialise par files for vertex
if (TAGrecoManager::GetPar()->IncludeVT() ) {
   fpParGeoVtx = new TAGparaDsc(new TAVTparGeo());
   TAVTparGeo* parGeo = (TAVTparGeo*)fpParGeoVtx->Object();
   TString parFileName = fCampManager->GetCurGeoFile(FootBaseN
   parGeo->FromFile(parFileName.Data());

   fpParConfVtx = new TAGparaDsc(new TAVTparConf());
   TAVTparConf* parConf = (TAVTparConf*)fpParConfVtx->Object()
   parFileName = fCampManager->GetCurConfFile(FootBaseName("TA
   parConf->FromFile(parFileName.Data());

fVtxTrackingAlgo = parConf->GetAnalysisPar().TrackingAlgo;
fVtxFlagPostTrack = parConf->GetAnalysisPar().PostTrackingF
```

```
if( TAGrecoManager::GetPar()->IncludeVT() )
{
  vtxparGeo = (TAVTparGeo*) fpVDparGeo->Object();
  vtxNtuHit = (TAVTntuHit*) fpVTNtuHit->Object();
  vtxNtuCluster = (TAVTntuCluster*) fpVTNtuCluster->Object();
  vtxNtuVertex = (TAVTntuVertex*) fpVTNtuVertex->Object();
  vtNtuTrack = (TAVTntuTrack*) fpVTNtuTrack->Object();
}

if( TAGrecoManager::GetPar()->IncludeIT() ) {
  itparGeo = (TAITparGeo*) fpITparGeo->Object();
  itNtuCluster = (TAITntuCluster*) fpITNtuCluster->Object();
}

if( TAGrecoManager::GetPar()->IncludeMSD() )
{
  msdparGeo = (TAMSDparGeo*) fpMSDparGeo->Object();
  msdNtuPoint = (TAMSDntuPoint*) fpMSDNtuPoint->Object();
  msdNtuCluster = (TAMSDntuCluster*) fpMSDNtuCluster->Object();
}

if( TAGrecoManager::GetPar()->IncludeBM() && bmNtuTrack->GetTracksN() == 1 )
  {
  TABMtrack* bmtrack = bmNtuTrack->GetTrack(0);

  if( TAGrecoManager::GetPar()->IncludeVT() && vtxNtuVertex->GetVertexN() == 1 )
    {
    TAVTvertex* vtxvertex = vtxNtuVertex->GetVertex(0);
    if(vtxvertex->IsBmMatched()){
      fpHisCorr_bm_vt_X->Fill(vtxvertex->GetVertexPosition().X(),bmtrack->GetOrigin().X());
      fpHisCorr_bm_vt_Y->Fill(vtxvertex->GetVertexPosition().Y(),bmtrack->GetOrigin().Y());
      fpHisCorr_bm_vt_XY->Fill(vtxvertex->GetVertexPosition().X(),bmtrack->GetOrigin().Y());
    }
  }
```

Style

- Different style for brackets
  - Breaking or not for method, condition, etc…

```
// initialise par files for vertex
if (TAGrecoManager::GetPar()->IncludeVT() ) {
   fpParGeoVtx = new TAGparaDsc(new TAVTparGeo());
   TAVTparGeo* parGeo = (TAVTparGeo*)fpParGeoVtx->Object();
   TString parFileName = fCampManager->GetCurGeoFile(FootBaseN
   parGeo->FromFile(parFileName.Data());

   fpParConfVtx = new TAGparaDsc(new TAVTparConf());
   TAVTparConf* parConf = (TAVTparConf*)fpParConfVtx->Object()
   parFileName = fCampManager->GetCurConfFile(FootBaseName("TA
   parConf->FromFile(parFileName.Data());

fVtxTrackingAlgo = parConf->GetAnalysisPar().TrackingAlgo;
fVtxFlagPostTrack = parConf->GetAnalysisPar().PostTrackingF
```

```
if( TAGrecoManager::GetPar()->IncludeVT() )
{
  vtxparGeo = (TAVTparGeo*) fpVDparGeo->Object();
  vtxNtuHit = (TAVTntuHit*) fpVTNtuHit->Object();
  vtxNtuCluster = (TAVTntuCluster*) fpVTNtuCluster->Object();
  vtxNtuVertex = (TAVTntuVertex*) fpVTNtuVertex->Object();
  vtNtuTrack = (TAVTntuTrack*) fpVTNtuTrack->Object();
}

if( TAGrecoManager::GetPar()->IncludeIT() ) {
  itparGeo = (TAITparGeo*) fpITparGeo->Object();
  itNtuCluster = (TAITntuCluster*) fpITNtuCluster->Object();
}

if( TAGrecoManager::GetPar()->IncludeMSD() )
{
  msdparGeo = (TAMSDparGeo*) fpMSDparGeo->Object();
  msdNtuPoint = (TAMSDntuPoint*) fpMSDNtuPoint->Object();
  msdNtuCluster = (TAMSDntuCluster*) fpMSDNtuCluster->Object();
}

if( TAGrecoManager::GetPar()->IncludeBM() && bmNtuTrack->GetTracksN() == 1 )
  {
  TABMtrack* bmtrack = bmNtuTrack->GetTrack(0);

  if( TAGrecoManager::GetPar()->IncludeVT() && vtxNtuVertex->GetVertexN() == 1 )
    {
    TAVTvertex* vtxvertex = vtxNtuVertex->GetVertex(0);
    if(vtxvertex->IsBmMatched()){
      fpHisCorr_bm_vt_X->Fill(vtxvertex->GetVertexPosition().X(),bmtrack->GetOrigin().X());
      fpHisCorr_bm_vt_Y->Fill(vtxvertex->GetVertexPosition().Y(),bmtrack->GetOrigin().Y());
      fpHisCorr_bm_vt_XY->Fill(vtxvertex->GetVertexPosition().X(),bmtrack->GetOrigin().Y());
    }
  }
```

➡ Not homogeneous

Tool (i)

Tool (i)

- Style fixing tool: *astyle*

Tool (i)

- Style fixing tool: *astyle*
  - Documentation here

Tool (i)

- Style fixing tool: *astyle*
  - Documentation here
  - Many options are available

Tool (i)

- Style fixing tool: *astyle*
  - Documentation here
  - Many options are available
  - Recursive fixing over folders is available

Tool (i)

- Style fixing tool: *astyle*
  - Documentation here
  - Many options are available
  - Recursive fixing over folders is available
  - Fix the code for a given style

Tool (i)

- Style fixing tool: *astyle*
  - Documentation here
  - Many options are available
  - Recursive fixing over folders is available
  - Fix the code for a given style

- Styles:

`--style=allman / --style=bsd / --style=break / -A1`

Allman style uses broken braces.

```
int Foo(bool isBar)
{
    if (isBar)
    {
        bar();
        return 1;
    }
    else
        return 0;
}
```

Tool (i)

- Style fixing tool: *astyle*
  - Documentation here
  - Many options are available
  - Recursive fixing over folders is available
  - Fix the code for a given style

- Styles:

--style=allman / --style=bsd / --style=break / -A1

Allman style uses broken braces.

```
int Foo(bool isBar)
{
    if (isBar)
    {
        bar();
        return 1;
    }
    else
        return 0;
}
```

--style=java / --style=attach / -A2

Java style uses attached braces.

```
int Foo(bool isBar) {
    if (isBar) {
        bar();
        return 1;
    } else
        return 0;
}
```

Tool (i)

- Style fixing tool: *astyle*
  - Documentation here
  - Many options are available
  - Recursive fixing over folders is available
  - Fix the code for a given style

- Styles:

--style=allman / --style=bsd / --style=break / -A1

Allman style uses broken braces.

```
int Foo(bool isBar)
{
    if (isBar)
    {
        bar();
        return 1;
    }
    else
        return 0;
}
```

--style=java / --style=attach / -A2

Java style uses attached braces.

```
int Foo(bool isBar) {
    if (isBar) {
        bar();
        return 1;
    } else
        return 0;
}
```

--style=kr / --style=k&r / --style=k/r / -A3

Kernighan & Ritchie style uses linux braces. Opening braces are broken from namespaces, classes, and function definitions. The braces are attached to everything else, including arrays, structs, enums, and statements within a function.

Using the k&r option may cause problems because of the &. This can be resolved by enclosing the k&r in quotes (e.g. --style="k&r") or by using one of the alternates --style=kr or --style=k/r.

```
int Foo(bool isBar)
{
    if (isBar) {
        bar();
        return 1;
    } else
        return 0;
}
```

Tool (i)

- Style fixing tool: *astyle*
  - Documentation here
  - Many options are available
  - Recursive fixing over folders is available
  - Fix the code for a given style

- Styles:

--style=allman / --style=bsd / --style=break / -A1

Allman style uses broken braces.

```
int Foo(bool isBar)
{
    if (isBar)
    {
        bar();
        return 1;
    }
    else
        return 0;
}
```

--style=java / --style=attach / -A2

Java style uses attached braces.

```
int Foo(bool isBar) {
    if (isBar) {
        bar();
        return 1;
    } else
        return 0;
}
```

➡ 16 different styles available

--style=kr / --style=k&r / --style=k/r / -A3

Kernighan & Ritchie style uses linux braces. Opening braces are broken from namespaces, classes, and function definitions. The braces are attached to everything else, including arrays, structs, enums, and statements within a function.

Using the k&r option may cause problems because of the &. This can be resolved by enclosing the k&r in quotes (e.g. --style="k&r") or by using one of the alternates --style=kr or --style=k/r.

```
int Foo(bool isBar)
{
    if (isBar) {
        bar();
        return 1;
    } else
        return 0;
}
```

Tool (ii)

- Options (excerpt): (i)

--pad-comma / -xg

Insert space padding after commas. This is not needed if pad-oper is used. Any end of line comments will remain in the original column, if possible. Note that there is no option to unpad. Once padded, they stay padded.

```
if (isFoo(a,b))
    bar(a,b);
```

becomes:

```
if (isFoo(a, b))
    bar(a, b);
```

Tool (ii)

- Options (excerpt): (i)

--pad-comma / -xg

Insert space padding after commas. This is not needed if pad-oper is used. Any end of line comments will remain in the original column, if possible. Note that there is no option to unpad. Once padded, they stay padded.

```
if (isFoo(a,b))
    bar(a,b);
```

becomes:

```
if (isFoo(a, b))
    bar(a, b);
```

--remove-braces / -xj

Remove braces from conditional statements (e.g. 'if', 'for', 'while'...). The statement must be a single statement on a single line. If --add-braces or --add-one-line-braces is also used the result will be to add braces. Braces will not be removed from "One True Brace Style", --style=1tbs.

```
if (isFoo)
{
    isFoo = false;
}
```

becomes:

```
if (isFoo)
    isFoo = false;
```

Tool (iii)

- Options (excerpt): (ii)

--align-reference=type

```
char* foo1;
char & foo2;
string ^s1;
```

becomes (with align-pointer=type):

```
char* foo1;
char& foo2;
string^ s1;
```

Tool (iii)

- Options (excerpt): (ii)

--align-reference=type

```
char* foo1;
char & foo2;
string ^s1;
```

becomes (with align-pointer=type):

```
char* foo1;
char& foo2;
string^ s1;
```

--align-pointer=type

```
char* foo1;
char & foo2;
string ^s1;
```

becomes (with align-pointer=type):

```
char* foo1;
char& foo2;
string^ s1;
```

Tool (iii)

- Options (excerpt): (ii)

--align-reference=type

```
char* foo1;
char & foo2;
string ^s1;
```

becomes (with align-pointer=type):

```
char* foo1;
char& foo2;
string^ s1;
```

--align-pointer=type

```
char* foo1;
char & foo2;
string ^s1;
```

becomes (with align-pointer=type):

```
char* foo1;
char& foo2;
string^ s1;
```

--squeeze-lines=#

Remove superfluous empty lines exceeding the given number.

```
void Foo()
{
    foo1 = 1;
}
```

```
void Foo2()
{
    foo1 = 1;
}
```

becomes:

```
void Foo()
{
    foo1 = 1;
}


void Foo2()
{
    foo1 = 1;
```

Tool (iv)

- Options (excerpt): (iii)

--keep-one-line-blocks / -O

Don't break one-line blocks.

```
// Getter methods
//! Get hit index
Int_t               GetHitIdx()       const         { return  fHitIdx;         }
//! Get pixel index
Int_t               GetPixelIndex()   const         { return  fPixelIndex;     }
//! Get pixel line
Int_t               GetPixelLine()    const         { return  fPixelLine;      }
//! Get pixel column
Int_t               GetPixelColumn()  const         { return  fPixelColumn;    }
```

Tool (iv)

- Options (excerpt): (iii)

`--keep-one-line-blocks / -O`

Don't break one-line blocks.

```
// Getter methods
//! Get hit index
Int_t               GetHitIdx()       const         { return  fHitIdx;         }
//! Get pixel index
Int_t               GetPixelIndex()   const         { return  fPixelIndex;     }
//! Get pixel line
Int_t               GetPixelLine()    const         { return  fPixelLine;      }
//! Get pixel column
Int_t               GetPixelColumn()  const         { return  fPixelColumn;    }
```

`--indent=spaces / --indent=spaces=# / -s#`

Indent using # **spaces** per indent (e.g. -s**3** --indent=spaces=**3**). # must be between 2 and 20. Not specifying # will result in a default of 4 spaces per indent.

with indent=spaces=3

```
void Foo() {
...if (isBar1
.........&& isBar2)     // indent of this line can be changed with min-conditional-indent
......bar();
}
```

Application

Application

- Command lines (none modified file is kept under xxx.orig)

Application

- Command lines (none modified file is kept under xxx.orig)
  - astyle --style=kr --*remove-braces* --pad-comma --align-reference=type --align-pointer=type —squeeze-lines=1 --indent=spaces=3 **--recursive '*.cxx'**

Application

- Command lines (none modified file is kept under xxx.orig)
  - astyle --style=kr --*remove-braces* --pad-comma --align-reference=type --align-pointer=type —squeeze-lines=1 --indent=spaces=3 **--recursive '*.cxx'**
  - astyle --style=kr --*keep-one-line-blocks* --pad-comma --align-reference=type --align-pointer=type ---squeeze-lines=1 --indent=spaces=3 **--recursive '*.hxx'**

Application

- Command lines (none modified file is kept under xxx.orig)
  - astyle --style=kr --*remove-braces* --pad-comma --align-reference=type --align-pointer=type —squeeze-lines=1 --indent=spaces=3 **--recursive '*.cxx'**
  - astyle --style=kr --*keep-one-line-blocks* --pad-comma --align-reference=type --align-pointer=type ---squeeze-lines=1 --indent=spaces=3 **--recursive '*.hxx'**

```
397 //_____________________________________________________________
398 //! Loop over events
399 //!
400 void TAGactFastDecode::IncludeNeededClasses()
401 {
402    if( TAGrecoManager::GetPar()->IncludeBM() )
403       bmNtuTrack = (TABMntuTrack*) fpBMNtuTrack->Object();
404
405    if( TAGrecoManager::GetPar()->IncludeVT() ) {
406       vtxparGeo = (TAVTparGeo*) fpVDparGeo->Object();
407       vtxNtuHit = (TAVTntuHit*) fpVTNtuHit->Object();
408       vtxNtuCluster = (TAVTntuCluster*) fpVTNtuCluster->Object();
409       vtxNtuVertex = (TAVTntuVertex*) fpVTNtuVertex->Object();
410       vtNtuTrack = (TAVTntuTrack*) fpVTNtuTrack->Object();
411    }
412
413    if( TAGrecoManager::GetPar()->IncludeIT() ) {
414       itparGeo = (TAITparGeo*) fpITparGeo->Object();
415       itNtuCluster = (TAITntuCluster*) fpITNtuCluster->Object();
416    }
417    if( TAGrecoManager::GetPar()->IncludeMSD() ) {
418       msdparGeo = (TAMSDparGeo*) fpMSDparGeo->Object();
419       msdNtuPoint = (TAMSDntuPoint*) fpMSDNtuPoint->Object();
420       msdNtuCluster = (TAMSDntuCluster*) fpMSDNtuCluster->Object();
421    }
422
423    if( TAGrecoManager::GetPar()->IncludeTW() )
424       twNtuPoint = (TATWntuPoint*) fpTWNtuPoint->Object();
425
426    if( TAGrecoManager::GetPar()->IncludeCA() )
427       caNtuCluster = (TACAntuCluster*) fpCANtuCluster->Object();
428
429    InitializeContainers();
430
431    return;
432 }
433
```

```
406 //! Loop over events
407 //!
408 void TAGactFastDecode::IncludeNeededClasses()
409 {
410    if( TAGrecoManager::GetPar()->IncludeBM() )
411       bmNtuTrack = (TABMntuTrack*) fpBMNtuTrack->Object();
412
413    if( TAGrecoManager::GetPar()->IncludeVT() )
414    {
415       vtxparGeo = (TAVTparGeo*) fpVDparGeo->Object();
416       vtxNtuHit = (TAVTntuHit*) fpVTNtuHit->Object();
417       vtxNtuCluster = (TAVTntuCluster*) fpVTNtuCluster->Object();
418       vtxNtuVertex = (TAVTntuVertex*) fpVTNtuVertex->Object();
419       vtNtuTrack = (TAVTntuTrack*) fpVTNtuTrack->Object();
420    }
421
422    if( TAGrecoManager::GetPar()->IncludeIT() ) {
423       itparGeo = (TAITparGeo*) fpITparGeo->Object();
424       itNtuCluster = (TAITntuCluster*) fpITNtuCluster->Object();
425    }
426    if( TAGrecoManager::GetPar()->IncludeMSD() )
427    {
428       msdparGeo = (TAMSDparGeo*) fpMSDparGeo->Object();
429       msdNtuPoint = (TAMSDntuPoint*) fpMSDNtuPoint->Object();
430       msdNtuCluster = (TAMSDntuCluster*) fpMSDNtuCluster->Object();
431    }
432
433    if( TAGrecoManager::GetPar()->IncludeTW() )
434       twNtuPoint = (TATWntuPoint*) fpTWNtuPoint->Object();
435
436    if( TAGrecoManager::GetPar()->IncludeCA() )
437       caNtuCluster = (TACAntuCluster*) fpCANtuCluster->Object();
438
439    InitializeContainers();
440
441    return;
442 }
```

Conclusions

Conclusions

- Fixing style

Conclusions

- Fixing style
  - Light tool, easy to install

Conclusions

- Fixing style
  - Light tool, easy to install
  - Very versatile (many options)

Conclusions

- Fixing style
  - Light tool, easy to install
  - Very versatile (many options)
  - In two command lines, fix style in all classes

Conclusions

- Fixing style
  - Light tool, easy to install
  - Very versatile (many options)
  - In two command lines, fix style in all classes

➡ Could be used for Shoe, fixing regularly the style and commit (once per week/month)

Conclusions

- Fixing style
  - Light tool, easy to install
  - Very versatile (many options)
  - In two command lines, fix style in all classes

➡ Could be used for Shoe, fixing regularly the style and commit (once per week/month)

➡ Have a common coding convention

(“my” convention for the moment, discussion needed 😀,
already started with Yun and Roberto)

Backup