PANDARoot

HOWTOs

Ver. 0 Rev. 6 (Apr 26 2007)

1. Howto install PANDARoot r1.6

There are 4 basic steps.

1. install the "mothership" - External Packages

See also:

http://cbmroot.gsi.de/installation/installation.htm

click on

-> External Packages

The download file is

http://cbmroot.gsi.de/download/100107/cbmsoft.100107.tar.gz

> *mkdir fairroot*
> *cd fairroot*
> *tar xzvf ../cbmsoft.100107.tar.gz*
> *cd [...]/fairroot/cbmsoft*
> *./configure.sh*

This will take ~1 hour or more, so you can go for a coffee.

This will compile and install pythia, geant3, geant4, proof, clhep and event generators such as pythia and pluto.

Note: this step you only have to do *once*, i.e. the first time you install pandaroot.

2. now install pandaroot itself.

Again, see also:

http://cbmroot.gsi.de/installation/installation.htm

click on

-> Installing PandaRoot

There are two ways to download the PandaRoot code: by a tar file or by svn (repository system). The tar file is usually an older release and the event generators are missing, thus we suggest to use the svn procedure.

> *cd [..]/fairroot/cbmsoft*
> *. ./SetEnv.sh*

> *cd cbmsoft*
> *svn co https://subversion.gsi.de/fairroot/pandaroot*
> *cd pandaroot*
> *./reconf*
> *cd ..*
> *mkdir build*
> *cd build*
> *../pandaroot/configure --prefix=$PWD --enable-geant3 --enable-geant4*
> *make*
> *make install*
> *. ./config.sh*

And the PandaRoot code will be compiled and (hopefully) running.

3. now set up your environment

This procedure should be followed each time you want to run PandaRoot macros. It is better first to load the external packages environment, than the PandaRoot one.

> *cd [..]/fairroot/cbmsoft*
> *. ./SetEnv.sh*
> *cd [..]/fairroot/cbmsoft/build*
> *. ./config.sh*

Now you can run all your macros and start to work.

4. now start working with pandaroot.

Here you can run a sample macro with EMC code to check if your PandaRoot installation worked or not.

After loading the environment (previous step):

> *cd [..]/fairroot/cbmsoft/pandaroot/macro/emc*

a.) run the simulation

> *root*
> ...
> *root [] .x sim_emc.C*

or

> *root sim_emc.C*

this will do a simulation (generate events with a box generator) and generate a root file as an output of the simulation.

> *sim_emc.root*

(for now, please ignore the 2nd output file, `simparams.root`)

This is a root file, so you can open a

> *root [] TBrowser t*

and investigate the tree by clicking for example

`ROOT Files -> sim_emc.root -> cbmsim -> EmcPoint -> EmcPoint.fX`

for the X coordinate in the laboratory frame.

The branch MCTrack contains the geant information of the tracks (just click on e.g. MCTrack.fPz for a plot of the z component of the track momentum)

Close root:

> *root [] .q*

Note: After you have analysed some file and you want to run something different, it is suggested to close root each time, then to load the new macro. In whis way one can avoid problems and segmentation violation.

b.) draw the detector

> *root*
> ...
> *root [] .x draw_geom.C*

or

> *root draw_geom.C*

In the canvas you can click on "view" and choose different kinds of geometry visualizations, such as OpenGL[?].

c.) load the simulation file and digitize hits (in a very preliminary way)

> *root*
> ...
> *root [] .x hit_emc.C*

or

> *root hit_emc.C*

This will write a file

> *hit_emc.root*

This is also a root file, so you can open a

> *root [] TBrowser t*

and investigate the tree by clicking for example

`ROOT Files -> hit_emc.root -> cbmsim -> EmcHit -> EmcHit.fTheta`

for a histogram of the polar angle distribution of the EMC hits.

This has been tested on several platforms (RedHat, Scientific Linux, Debian Sarge, Suse, etc., at GSI and/or stand-alone). Please don't hesitate even to try to install on your notebook (as many of us did). However, it is impossible to guarantee for *all* platforms (I think it is e.g. not tested for Ubuntu or Knoppix).

noppix).

Note:

If you get any error related to the ROOT installation regarding SQL headers, download the SQL source code from http://www.iodbc.org/index.php?page=downloads/index and do the following steps:

```
1. tar xzf libiodbc-<version>.tar.gz
2. cd libiodbc-<version>
3. ./configure
4. make (or gmake)
5. make install (as root)
```

5. Build from scratch on SL4.

There is a script to install everything (including GEANT4) from the scratch on SL4. Maybe it is useful for someone.

Please, check the prerequisite before (wget, svn, automake >= 1.09, autoconf >= 2.59, gcc, g77 >= 3.2)!

Download the two files build_full.sh and conf.tgz into whatever brand new directory. The script to build everything is: build_full.sh. Make it executable (e.g., chmod u+x build_full.sh) and launch it in execution. It will take several hours (depending on your internet connection and your hardware), but in the end you should have everything compiled and work on (just see how can you work with cbmsoft).

One more thing! As you probably noticed, it is a BASH script and not a TCSH script. Perhaps it is not a bad idea for someone who knows TCSH just to make a translation from my script to TCSH script (the script is working, but it needs a patch to release an env.csh for GEANT4).

In the end, you can find how a complete report should look like (if you look in your log dir). Just check log.tgz.

Remark: some of you might be behind a nasty firewall or proxy, which means you have not access to the port 80 used by default by wget. In that case, just download manually the tar ball of the external packages needed by PANDARoot, following this link:

http://cbmroot.gsi.de/download/100107/cbmsoft.100107.tar.gz

put it into the same directory and start again the script, after you deleted all the subdirectories within. It should work.

Links:

http://wiki.gsi.de/pub/Pandacomputing/PandaRootInstall/build_full.sh

http://wiki.gsi.de/pub/Pandacomputing/PandaRootInstall/conf.tgz

2. HOWTO upgrade PandaRoot r1.1

This part describes, how to get a new version of pandaroot, if you already have a version of pandaroot installed.

This refers *only* to step 2 in the installation procedure in PandaRootInstall. In other words, the "mothership" (i.e. the external packages) are only upgraded once in 1-2 years. In such a case, there will be a special announcement.

For the below part, you need a working version of subversion, please see PandaRootSubversion.

```
> cd [..]/fairroot/cbmsoft
> . ./SetEnv.sh
> svn co https://subversion.gsi.de/fairroot/pandaroot
> cd pandaroot
> ./reconf
> cd ..
> mkdir build
> cd build
> ../pandaroot/configure --prefix=$PWD --enable-geant3
>  make
>  make install
> .  ./config.sh
```

and the upgrade is finished.

3. HOWTO Using subversion r1.3

What is *subversion* ?

subversion is the code revision system (which formerly was CVS).

HOWTO get *subversion*

http://subversion.tigris.org/

-> click on *Downloads*

HOWTO check-out the newest pandaroot Version

```
> svn co https://subversion.gsi.de/fairroot/pandaroot
```

On GSI computers, subversion is already installed. So if you don't have subversion installed at your home institution, you can check-out PandaRoot on any lxi00n.gsi.de machine and then copy the code to your home institution.

HOWTO merge two PandaRoot versions

Now imagine that you checked out a PandaRoot version.

Then you wrote some code and changed a few files.

Meanwhile someone else changed another part of the code.

What now ?

You can merge the two versions using

```
> svn update pandaroot
```

you can generate a pandaroot version with the *newest* code (it means, it will contain your changed files and the changed files of the other person).

Please note that here "pandaroot" is your directory, so it is *not* the repository (https://...).

HOWTO get write access to pandaroot

For developers who want to check-in their newest version:

Please go to a linux machine with an *apache* www server installed. type:

```
> htpasswd -c -b <filename> <username> <password>
```

where you have to choose what to type for This will create a file named "filename". Take this file, attach it to an email and please send it to This file contains your username (as it will be in the subversion system) and your encrypted password (so we will not be able to see it).

4. HOWTO use the PandaRoot Database Interface r1.2

This example of how to write e.g. calibration parameters into the runtime database (rtdb, which in this case is only a root file) you can find in the /macro directory in emcsim.C:

```
CbmRuntimeDb *rtdb=fRun->GetRuntimeDb();
CbmParRootFileIo* output=new CbmParRootFileIo(kParameterMerged);
output->open("simparams.root");
rtdb->setOutput(output);
rtdb->saveOutput();
rtdb->print();
```

Later, the runtime database will be ORACLE (decided due to the existing GSI IT division support).

Example of implementation of Run-Time DB with ascii file (preliminary):

1) Create parameter container class (EmcDigiPar[?]). It is inherited from CbmParSet[?]. Implement method EmcDigiPar[?]::readline(), which defeine how parameters are read from ascii file:

```
void EmcDigiPar::readline(const char *buffer, Int_t *set, fstream *f)
{
 int use_shaped_noise_int, use_photon_statistic_int;
 sscanf(buffer, "%lf%d%lf%lf%lf%lf%d%lf%lf%lf%lf%lf%d%d%lf%lf",
&eneThr, &nBits,
 &detectedPhotonsPerMeV, &energyRange, &excessNoiseFactor,
&firstSamplePhase,
 &number_of_samples_in_waveform, &Shaping_diff_time ,
&Shaping_int_time,
 &crystal_time_constant,&incoherent_elec_noise_width_GeV, &sampleRate ,
 &use_shaped_noise_int, &use_photon_statistic_int, &sampleRate,
&threshold);

 use_shaped_noise= static_cast<bool> (use_shaped_noise_int);
 use_photon_statistic= static_cast<bool> (use_photon_statistic_int);
}
```

this class should also have accessor methods for specific parameters.

2) Define ascii file format. Reading of ascii file is distributed to 2 classes EmcDigiPar[?]::readline() and EmcParAsciiFileIO[?]::read(T* pPar, Int_t* set, Bool_t needsClear)

Each line in the header of the ascii file start from #, then goes line [EmcDigiPar] with name of parameter container and then line with values of parameters.

3) Implement class EmcParAsciiFileIO[?] To read parameters 2 methods should be implemented init() and read()

```
Bool_t EmcParAsciiFileIo::init(CbmParSet* pPar) {
  // calls the appropriate read function for the container
  const  Text_t* name=pPar->GetName();
   cout << "-I- Ascii Io init() " << pPar->GetName() <<   endl;

  if (pFile) {
    if (!strcmp(name,"EmcDigiPar")) return
read((EmcDigiPar*)pPar,0,kTRUE);
    cerr<<"initialization of "<<name<<" not possible from file!"<<endl;
    return kFALSE;
  }
  cerr<<"no input file open"<<endl;
  return kFALSE;
}
```

This is mainly copy&paste work with spesific parameter container name. read() method can be seen in the code directly (it is a little bit long to post it here).

4) Add parameter class to parameter container factory CbmEmcContFact[?]

```
CbmEmcContFact::setAllContainers()
{
CbmContainer* p2= new CbmContainer("EmcDigiPar",
                                        "Emc Digitalization
Parameters",
                                        "TestDefaultContext");
p2->addContext("TestNonDefaultContext");
containers->Add(p2);
}
```

and

```
CbmParSet* CbmEmcContFact::createContainer(CbmContainer* c)
{
  CbmParSet* p=NULL;
  if (strcmp(name,"EmcDigiPar")==0) {
  p=new EmcDigiPar(c->getConcatName().Data(),c->GetTitle(),c-
>getContext());
}
}
```

activate specific I/O method

```
void  CbmEmcContFact::activateParIo(CbmParIo* io)
{
 if (strcmp(io->IsA()->GetName(),"CbmParAsciiFileIo")==0)
 {
  EmcParAsciiFileIo* p=new EmcParAsciiFileIo(((CbmParAsciiFileIo*)io)-
>getFile());
  io->setDetParIo(p);
 }
}
```

5) In class where these parameters are used (CbmEmcHitProducer[?] for example) implement SetParContainers[?]() method

```
void CbmEmcHitProducer::SetParContainers() {

  // Get run and runtime database
  CbmRunAna* run = CbmRunAna::Instance();
  if ( ! run ) Fatal("SetParContainers", "No analysis run");
```

```
  CbmRuntimeDb* db = run->GetRuntimeDb();
  if ( ! db ) Fatal("SetParContainers", "No runtime database");

  // Get Emc digitisation parameter container
  fDigiPar = (EmcDigiPar*) db->getContainer("EmcDigiPar");

}
```

After that parameter can be used with correspondent accessor method of container class eneThr =fDigiPar->GetEnergyThresholdHit();

6) In the script which run simulation or analysis (digi_emc.C for example)

provide name of file with parameter value and set specific I/O method

```
TString digiFile = "emc.digi.par";
TString emcDigiFile = gSystem->Getenv("VMCWORKDIR");
emcDigiFile += "/emc/";
emcDigiFile += digiFile;

  CbmRunAna *fRun= new CbmRunAna();
CbmRuntimeDb* rtdb = fRun->GetRuntimeDb();
CbmParAsciiFileIo* parIo1 = new CbmParAsciiFileIo();
parIo1->open(emcDigiFile.Data(),"in");
rtdb->setFirstInput(parIo1);
```

5. PandaROOT Geometry definition r1.5

Do not forget to define a MOTHERVOLUME for your detector.
i.e. do not use the cave as the mothervolume !
otherwise you can never move your detector later if there is a geometry overlap with another detector

General

The filenames for the **geometry definition** and the **media definition** must have the suffix **.geo** and must contain a special keyword for every set:

- media (for all materials needed in Geant for the standard geometry)
- cave
- Magnet
- STS
- …

Files for media

Each medium is characterized by a name. The names of the materials and the media are identical. Together with the name all parameters must be given which are needed for the Geant routines GSMATE, GSMIXT, GSTMED and GSCKOV.
There are no predefined materials!

The following parameters are needed:

- **int ncomp** - number of components in the material (ncomp= 1 for a basic material and <1 or >1 for a mixture ; see NLMAT in Geant routine GSMIXT
- **float aw[ncomp]** - atomic weights A for the components
- **float an[ncomp]** - atomic numbers Z for the components
- **float dens** - density DENS in g cm(**-3)
- **float radleng** - radiation length RADL (only for a basic material) or
- **float wm[ncomp]** - weights WMAT of each component in a mixture (only for a mixture)
- **int sensflag** - sensitivity flag ISVOL
- **int fldflag** - fieldflag IFIELD
- **float fld** - maximum field value FIELDM in kilogauss
- **float epsil** - boundary crossing precision EPSIL
- **int npckov** - number of values used to define the optical properties of

the medium.

The variable npckov is 0 for all media except some special media used for the Rich where the tracking of the Cerenkov photons is necessary. These media have additinal parameters

- **float ppckov[npckov] -** photon momentum in eV
- **float absco[npckov] -** absorption length in case of dielectric and of absorption probabilities in case of a metal
- **float effic[npckov] -** detection efficiency
- **float rindex[npckov] -** refraction index for a dielectric, rindex[0]=0 for a metal

Remark: In the present program version a mixture may contain a maximum of 5 components. If this is not sufficient one has to change MAXCOMP in hgeomedium.h.

The following parameters are normally not read. The default values are -1 and the real values are automatically calculated by Geant. If you want to set these values by yourself, you must type the keyword AUTONULL in your media file. After this keyword all media must contain these additional 4 parameters at the end.

- **float madfld** - maximum angular deviation TMAXFD due to field
- **float maxstep** - maximum step permitted STEMAX
- **float maxde** - maximum fractional energy loss DEEMAX
- **float minstep** - minimum value for step STMIN

Comments can be placed before a medium in a new line starting with //

Examples:

ALUMINIUM$	1 0	26.98	13	2.7	7.25485	0	0	0	0.0001
corresponds to name	ncomp npckov	aw	an	dens	radleng	sensflag	fldflag	fld	epsil
C4F10	-2 0 0	12.01 1	19. 3.	6. .001	9.	7.35485	4.		10.

corresponds to

name	ncomp	aw(1)	aw(2)	an(1)	an(2)	dens	wm(1)	wm(2)
	sensflag	fldflag	fld	epsil				
	npckov							

Files for geometry

Each file consists of a list of volumes. Each volume is characterized by its name and a list of parameters. Comments (marked with // at the beginning of a line) are allowed only at the beginning or the end of a volume.

A detector may consist of several modules sitting in a keepin volume and each module consists of several daughters, grand-daughters and so on. First the keepin volume will be read and afterwards each module with all volumes inside. The general construction (keepin volumes or not, number of modules) is for each detector hardwired in the program (see chapter **Implemented sets**). All inner parts of a module are only defined through the input file. You are free to change the names (with some restrictions), media, shapes, dimensions, and the tree itself, e.g. by inserting volumes, without recompiling the program. The only restriction is that the program will not create a daughter before it has created its mother, thus the mother has to show up in the list before its daughters.

Important: All volumes are created in Geant with the 'ONLY' flag!

. The general input for a volume (except CAVE) looks like:

- name of the volume
- inout flag (only for keepin volumes and modules, not for subvolumes of a module)
- name of the mother-volume
- Geant shape of the volume
- name of the medium (= material)
- points or parameters from technical drawings (depends on the shape)
- position (x y z) of the coordinate system, in which the points are given, in the coordinate system of the mother
- 3x3 rotational matrix of the coordinate system listed rowwise as a vector

For explanation:

Names of volumes: For each detector the names of the keepin-volumes and the modules are hardwired in the program (see chapter **Implemented sets**).

The inner parts of a module have names starting with a detector specific string. Each volume to be created in Geant has a name with at least 4 characters (upper case letters). Only the first 4 characters are used in Geant. All volumes of which several copies exist have names with 5 or more characters. These additionally characters are always digits ranging from 1 to the maximum number of volumes with the same 4-character Geant name. For example Tof 1 contains 8 identical cells T01S1 ... T01S8. The volume T01S is created only once via GSVOLU(...) but positioned 8 times via GSPOS(...). The names of the sensitive volumes must have a special structure which is defined for each detector in chapter **Implemented sets**.

Inout flag: All keepin volumes and mothervolumes of a detector may have an inout flag. If this inout flag is 1 the volume is created otherwise not. If the flag is not given in the input file the program (next line after name of volume) the program sets it automatically to 1. By setting this flag to 0 in the input file you can prevent for example the creation of a module in a special sector. This flag is not stored in the database. When reading from database the flag is always set to 1 by the program.

Name of the mother: Each volume is positioned in a mother. The name of the mother has 4 characters (in upper case letters) plus eventually a 5^{th} character if several copies of the mother exist. This 5^{th} character has to be always 1. (In Geant only 4 characters are used, but 5 characters are necessary to find the mother in Oracle.)

Geant Shape: Each volume has a shape (4 characters in upper case letters). The shapes yet implemented in the program are BOX , PGON, PCON, TRAP, TRD1, TUBE, TUBS, CONE, CONS, SPHE, ELTU.

Medium: Each volume is filled with a medium given by the name of the medium. This medium must be defined in the medium file.

Points: Each volume has parameters describing the dimensions. The number of these parameters and their meaning depend on the shape of the volume. This is explained in chapter **Shapes**.

Coordinate system: Each volume has a coordinate system in which the parameters are given. Its position and orientation relative to the coordinate system of the mother volume is described by a translation vector T and a 3x3 rotation matrix R according to the equation x=R*x'+T where x is the vector in the coordinate system of the mother, and x' the vector in the coordinate system of the daughter. The matrix R is listed rowwise as a vector with 9 components. Each shape has its own intrinsic orientation which is defined similar to the Geant definition except for TRAP and TRD1 (see chapter **Shapes**). **Important**: The rotation matrix should be given with a precision of 10**-6. Otherwise you would get a lot of warnings from Geant that the coordinate system is not orthogonal. As long as all volumes in a module are not rotated you may use the coordinate system of the module as the base coordinate

system in which all points are given. More information about the coordinate transformations you can find in the document **Hades Geometry and Database** (http://www-win.gsi.de/hadesora/geometry/Geometry.htm) of Michael Dahlinger.

Important: Volumes of which several copies exist have a special input structure: For the first volume all information above is read. The input for the other copies looks like:

- name of the volume
- inout flag if is the volume is a keepin volume or a module (may be missing)
- name of the mother
- translation vector of the coordinate transformation
- rotation matrix of the coordinate transformation

The other information is not necessary because it has to be identical.

Remark: A (main) routine **hgeoeuler.cc** to calculate the rotation matrix from Euler angles can be found in the module 'ggeo' in the Hades-CVS.

Shapes

A variable number of 'points' (one in each line of the input file), each having a maximum of 3 values, can be stored in the program. Apart from the shapes with 8 corners (BOX , TRAP, TRD1) the parameter are at least very similar to the ones in Geant. Except for a TRAP and a TRD1 the intrinsic coordinate systems are defined as in Geant.

BOX

This shapes has 8 corners described by the x, y, z coordinates.

TRAP

This shapes has 8 corners described by the x, y, z coordinates. The corners are counted clockwise starting at the lower left corner of the bottom plane. The intrinsic coordinate system of a TRAP is different from the one in Geant. The y-axis points from the smaller side in x-direction to the larger one. A TRAP not rotated in a Box has the same intrinsic coordinate system as the BOX. In Geant the y- and x-axis point in the opposite directions.

TRD1

This shapes has 8 corners described by the x, y, z coordinates. The intrinsic coordinate system of a TRD1 is the same as for a TRAP. That's different from the definition in Geant.

PGON

This shape has a variable number of 'points'.

point 0:	NZ number of planes perpendicular to the z-axis where the section is given
point 1:	azimuthal angle PHI1 at which the volume begins
	opening angle DPHI of the volume
	number NPDV of sides of the cross section between the phi limits
point 2ff:	z coordinate Z of the section
	inner radius RMIN at position z
	outer radius RMAX at position z

PCON

This shape has a variable number of 'points'.

point 0:	NZ number of planes perpendicular to the z-axis where the section is given
point 1:	azimuthal angle PHI1 at which the volume begins
	opening angle DPHI of the volume
point 2ff:	z coordinate Z of the section
	inner radius RMIN at position z
	outer radius RMAX at position z

SPHE

This shape has 3 'points' with 2 parameters each.

Point 0:	inner radius RMIN of the shell
	outer radius RMAX of the shell
point1:	starting polar angle THE1 of the shell
	ending polar angle THE2 of the shell

point 2:	starting azimuthal angle PHI1 of the shell
	ending azimuthal angle PHI2 of the shell

TUBE

This shape has 3 'points'.

point 0:	x, y, z coordinate of the center of the circle at the beginning of the tube
point 1:	inner radius RMIN of the tube
	outer radius RMAX of the tube
point 2:	x, y, z coordinate of the center of the circle at the end of the tube

TUBS

This shape has 4 'points'.

point 0:	x, y, z coordinate of the center of the circle at the beginning of the tubs
point 1:	inner radius RMIN of the tubs
	outer radius RMAX of the tubs
point 2:	x, y, z coordinate of the center of the circle at the end of the tubs
point 3:	starting angle PHI1 of the segment
	ending angle PHI2 of the segment

CONE

This shape has 4 'points'.

point 0:	x, y, z coordinate of the center of the circle at the beginning of the cone
point 1:	inner radius RMN1 at the beginning of the cone
	outer radius RMX1at the beginning of the cone
point 2:	x, y, z coordinate of the center of the circle at the end of the cone
point 3:	inner radius RMN2 at the end of the cone
	outer radius RMX2 at the end of the cone

CONS

This shape has 5 'points'.

Point 0:	x, y, z coordinate of the center of the circle at the beginning of the cons
point 1:	inner radius RMN1 at the beginning of the cons outer radius RMX1 at the beginning of the cons
point 2:	x, y, z coordinate of the center of the circle at the end of the cons
point 3:	inner radius RMN2 at the end of the cons outer radius RMX2 at the end of the cons
point 4:	starting angle PHI1 of the segment ending angle PHI2 of the segment

ELTU

This shape has 3 'points'.

point 0:	x, y, z coordinate of the center of the ellipsoid at the beginning of the eltu
point 1:	semi-axis P1 along x semi-axis P2 along y
point 2:	x, y, z coordinate of the center of the ellipsoid at the end of the eltu

Warning: A TUBE, TUBS, CONE, ELTU can not be rotated by different x- and y-values of the starting and ending circles. They have to be identical. A rotation can only be described by a rotation matrix.

TORUS

This shape has 5 'points'. (not available in native Geant3!!)

point 0:	R - axial radius
point 1:	Rmin - inner radius
point 2:	Rmax - outer radius
point 3:	Phi1 - starting phi
point 4:	Dphi - phi extent

Conversion between xml and Pandaroot.

Example: Muon detector

The muon detector is defined as a plane of Iarocci Tubes (PIT) and a plane of Sensitive Strips (PSS). A description of the geometry can be found here (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/muon.ppt), with two applications:

- 3 planes placed after thze magnetic field
- 2 planes, placed one before one after the magnet yoke

XML

In the Xml version of the muon detector only one plane is present. There are 5 different files. One defines the global dimension variables, while for each detector (PIT and PSS) there are other two files, one for the dimensions one for the geometry definitions.

- muon.xml: Definition of the global MUON detector dimensions

  (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/muon.xml)

- PITDim.xml: Definition of the Iarocci Tube dimensions

  (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/PITDim.xml)

- PITMain.xml: Geometry definition of Iarocci Tubes

  (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/PITMain.xml)

- PSSDim.xml: Definition of the Sensitive Stripe dimensions

  (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/PSSDim.xml)

- PSSMain.xml: Geometry definition of the Sensitive Strips

  (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/PSSMain.xml)

PANDAroot

In the PANDAroot framework an ASCII file is provided that contains all the volumes. We have two different files, one per kind of geometry. Each file was created as ASCII output by a ROOT macro, where one can set the dimension variables and just let it run. For the three-plane geometry we have:

- Muon_design_geometry2.C: Macro to design the geometry of the Muon detector: 3 planes (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/Muon_design_geometry2.C)
- muon_3out.geo: Geometry ASCII file for the Muon detector: 3 planes (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/muon_3out.geo)

For the in-out geometry we have:

- Muon_design_geometry.C: Macro to design the geometry of the Muon detector:

in/out planes (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/Muon_design_geometry.C)

- muon_inout.geo: Geometry ASCII file for the Muon detector: in/out planes (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/muon_inout.geo)

All the volumes are inheritated by a mother volume.
The first volume is the **CAVE** (defined by the global cave.geo file in the geometry directory), filled with air. In the cave we define a muon plane (**muop1#1**), filled by air, thus the noryl box (**muop1b#1**), which contains the alluminium (**muop1al**) which contains the CF4_CH4 gas tubes (**muop1t#1**) which contain the tungsten wire (**muop1wi**).

```
CAVE
 |
 muop1#1 muop1#2 muop1#3 muop1#4 muop1#5 muop1#6 muop1#7 muop1#8
  |
  muop1b#1 muop1b#2 ... muop1b#13
   |
   muop1al
    |
    muop1t#1 muop1t#2 ... muop1t#8
     |
     muop1wi
```

After that one volume is defined, in terms of materials, geometry and daughters, it is possible to simply copy rotated and translated in the space without rewritting everything. One has to use the same number followed by #2 #3 and so on. In this case one has only to define the transation/rotation matrix. All the daughter volumes will be copied in the structure automatically.

Example: after the definition of one sector (**muop1#1**), one only has to write:

```
muop1#2
cave
-1332.189 1332.189 350.000
0.707107 -0.707107 0.000000 0.707107 0.707107 0.000000 0.000000 0.000000
1.000000
//*********************************
muop1#3
cave
-1884.000 0.000 350.000
0.000000 -1.000000 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000
1.000000
//*********************************
...
```

instead of shousand of lines, with a strong reduction of disk space and CPU time for initialising the geometry.

Conversion between Geant4 and Pandaroot.

Example: EMC detector

The EMC detector is defined as a set of PWO crystals, as shown here (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/emc.ppt).

One table with the geometrical definitions of all the crystal can be downloaded here (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/emc_module123.dat).
All the numbers represent the GEANT4 TRAP parmeters.
Modules 1 and 2 belong to the barrel part (forward and backward part), while module 3 belongs to the forward endcup.

XML

In XML there is only a big file, which contains all the crystal defined separately.

- emcBarrelA.xml(http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/emcBarrelA.xml): Xml EMC definition

GEANT4

In Geant4 one class can be used, which loads the parameters from the ASCII file and and create all the G4Traps.

- DetectorConstruction.cc(http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/DetectorConstruction.cc) : Geant4 class for EMC geometry definition

PandaROOT

In PandaROOT it is possible to reproduce the Geant4 code, simply by using the related ROOT classes (TGeoTrap instead of G4Trap and so on.
As example we can look at the CbmEmc.cxx (http://wiki.gsi.de/pub/Pandacomputing/PandaRootGeometry/CbmEmc.cxx), which defines the simulation definition of the EMC.
In the **CbmEmc::ConstructGeometry** method the geometry is defined. We can see that the code is quite similar to the Geant4, whit small changes.
The volume **Emc12** contains the volumes **EmcLayer1** and **EmcLayer2**, the barrel parts, while **Emc3** is the endcup.

In order to make the code simpler and faster it was decided to use the "copy" scheme. The endcup was divided in 4 parts. Only the first 90 degrees are defined, the other three parts are simply defined by copying the main volume with the proper rotation matrix:

```
for (Int_t n=1;n<=3;n++){ <br> TGeoRotation rot1;
rot1.RotateZ(90*n);
vcave->AddNode(flayer3, n+1,new TGeoCombiTrans(0., 0., 0., new
TGeoRotation(rot1)) );
}
```

For the barrel part only rows of 10 "crystals" are defined, which should correspond to 22.5 degrees. The other crystals are defined by simply copying the main volume 16 times with the correct rotation matrix:

```
for (Int_t n=1;n<=15;n++){<br> TGeoRotation? rot1;
rot1.RotateZ(22.5*n);
vcave->AddNode(flayer12, n+1,new TGeoCombiTrans?(0., 0., 0., new TGeoRotation?
(rot1)) );
}
```

For the CPU a copied volume is lighter with respect to the "original" one because it does not require a full initialisation.
In the EMC case the endcup initialization requires almost 1/4 of the "official" time, while for the barrel part we gain a factor 15. This gain, in the case of several thousands of volumes, can be very helpful in order to avoid a waiting of tens of minutes while running the simulation.

6. PandaROOT Event Generators

6.1 EvtGen r1.12

EvtGen is an event generator used by several collaboration (BaBar, CLEO, D0, Belle...), well suited for the physics of B-meson decays, for complex sequencial decays and CP violating decays. Further informations can be found on the website (http://www.slac.stanford.edu/~lange/EvtGen/).

Installation

The EvtGen software is included in between the PandaRoot packages, under the directory pgenerators/EvtGen$.
The source code is in there but it is not compiled by the global Makefile, so you have to do it by yourself.

Before installing the code, one should set some environmental variables to make everything works. The followings have to be set each time one want to use EvtGen, so it is better tom write the following definition even in your ".bashrc" or ".profile" file.
One has to set the CERN_ROOT, ROOTSYS and CLHEP_BASE_DIR. For CERN_ROOT one has to find in the computer where the files libpythia*.a, libphotos*.a and libpdf*.a stay, and even clhep libraries.
For example, at GSI they stay in /misc/cbmsoft/Debian3.1/cern/cernlib/2005/lib. One has to set:

- export CERN_ROOT=/misc/cbmsoft/Debian3.1/cern/cernlib/2005
- export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CLHEP_LIB_DIR

(the ../lib has not to be written for CERN_ROOT).
CLHEP_BASE_DIR and ROOTSYS are already set by the SetEnv.sh of the PandaROOT external packages.
It is better to put the previous lines in your .profile, so that they are loaded each time you open a linux shell.

After the environment setting one should configure the system. First one has to go into the EvtGen directory:

- cd pandaroot/pgenerators/EvtGen

thus configure:

- . configure

The configure script will start asking you several questions:

- Enter Name of c++ compiler (type "gcc", or your c++ compiler name)
- Enter name of fortran compiler (type "g77", or your fortran compiler name)
- Enter any other options that you would like to use in compilation (just press

Enter)
- Please enter any others that you would like to use (just press Enter)

And that's all for the configuration procedure.

In order to see if everything works, we can compile the test macro included with the code. Just type:

- make
- make bin

and the script testMyEvt should appear without problems.
Just go to the "test" directory and launch the test macro:

- cd test
- ../testEvtGen test1 1000

and 1000 events will be generated from the decay chain contained in the TEST1.DEC file.
If you open the test1.root file with ROOT and browse the TBrowser, you can find several histograms produced during the event processing.

Event generation

The main assumption in the following lines is that the reader knows how to use EvtGen, or better how to define a decay chain in the .DEC file.
It will be explained how to write an output file that can be loaded from the PandaROOT framework, to put it into the transport model (GEANT3, GEANT4...) and to produce a readable root file.

In order to generate events in the PandaROOT format one should use the EvtStdHep class output.

The file "pandaEvtGen.cc" (http://wiki.gsi.de/pub/Pandacomputing/EvtGen/pandaEvtGen.cc) in the main EvtGen directory shows an easy example, with the decay eta_c -> gamma gamma.
The file test/ETACTOGG.DEC define the decay chain. This is a simple phase space distribution:

```
Decay eta_c
1.000 gamma gamma PHSP;
Enddecay
End
```

In order to produce an output file just compile the file pandaEvtGen. In the main EvtGen directory just type:

- make
- make panda
- . pandaEvtGen

You will have an output file "output.evt" (http://wiki.gsi.de/pub/Pandacomputing/EvtGen/output.evt), that can be used as input for

the CbmEvtGenGenerator class of PandaROOT, in order to perform the full simulation and hit digitization analysis.
This is an example on how it should look like.

```
0   3
  N      Id Ist   M1   M2   DF   DL      px      py      pz       E       t       x       y       z
  0     441   2   -1   -1    1    2  0.0000  0.0000  0.0000  2.9808  0.0000  0.0000  0.0000  0.0000
  1      22   1    0    0   -1   -1 -1.2185 -0.7610  0.3966  1.4904  0.0000  0.0000  0.0000  0.0000
  2      22   1    0    0   -1   -1  1.2185  0.7610 -0.3966  1.4904  0.0000  0.0000  0.0000  0.0000


1   3
  N      Id Ist   M1   M2   DF   DL      px      py      pz       E       t       x       y       z
  0     441   2   -1   -1    1    2  0.0000  0.0000  0.0000  2.9551  0.0000  0.0000  0.0000  0.0000
  1      22   1    0    0   -1   -1 -0.5996 -0.7311  1.1354  1.4776  0.0000  0.0000  0.0000  0.0000
  2      22   1    0    0   -1   -1  0.5996  0.7311 -1.1354  1.4776  0.0000  0.0000  0.0000  0.0000

...
```

You can play with the pandaEvtGen.cc file, just changing the number of events or the reactions. You could use the test/ETACTOPI0PI0ETA.DEC file, where the following decay is defined:

```
Decay eta_c
1.000 pi0 pi0 eta PHSP;
Enddecay

Decay pi0
1.000 gamma gamma PHSP;
Enddecay

Decay eta
1.000 gamma gamma PHSP;
Enddecay
End
```

where the η_c at rest decay in η_c -> $\pi^\circ\pi^\circ\eta$, and all the π° and η mesons are forced to decay into two photons.

The important thing is just to use the same output structure which is defined in the pandaEvtGen.cc file, thus:

```
out << i << "\t" << evtstdhep.getNPart();
out <<evtstdhep<<endl;
```

Please remember that after you change the pandaEvtGen.cc file, you have first to make and after to make panda. If you type only make panda you will not load your newest changes.

After the output file is created, one has only to propagate it into the transport model, thus inside pandaROOT.

Simulation inside PandaROOT

The interface between EvtGen and PandaROOT consists on the CbmEvtGenGenerator class. This has to be used in the simulation files, as shown in the following lines:

```
CbmPrimaryGenerator* primGen = new CbmPrimaryGenerator();
```

```
fRun->SetGenerator(primGen);

CbmEvtGenGenerator* evtGen = new CbmEvtGenGenerator("output.evt");
primGen->AddGenerator(evtGen);
```

where output.evt is our output file with the full path.
The file sim_emc_evtgen.C (http://wiki.gsi.de/pub/Pandacomputing/EvtGen/sim_emc_evtgen.C) is an example on how to load our output.evt file.
Only the EMC geometry is defined and no magnetic field is present, the simulation macro is used just to check the response of the generator.

The simulation file was processed and EMC digitized hits were produced. Here (http://wiki.gsi.de/pub/Pandacomputing/EvtGen/EvtGen_etac.ppt) you can find several slides showing the response of the test EMC to two photons and six photons events. We can see that everything seems to work properly!!!!

One important thing: The number of processed events in the simulation macro should not exceed the event number in the evt file. In the contrary case the analysis will crash and the output file will be broken (no safe exit).
This problem will be removed as soon as possible.

Code

- CbmEvtGenGenerator.h (http://wiki.gsi.de/pub/Pandacomputing/EvtGen/CbmEvtGenGenerator.h)
- CbmEvtGenGenerator.cxx (http://wiki.gsi.de/pub/Pandacomputing/EvtGen/CbmEvtGenGenerator.cxx)

6.2 DPM Event Generators r1.4

DPM (Dual Parton Model) generator is used to produce background events using DPM model.

Installation

he Dpm software is included in between the PandaRoot packages, under the directory pgenerators/DpmEvtGen.
The source code is in there but it is not compiled by the global Makefile, so you have to do it by yourself.

Only the normal environment variables are needed to compile the code, so one has only to launch the SetEnv.sh script, and after to go to the DpmEvtGen directory and compile with:

- cd pandaroot/pgenerators/DpmEvtGen
- make

Some directories will be created to compile the code, and at last you will find in the main directory the executable DPMGen.

Event generation

After the executable is created, one has only to execute it, and that's all.

- ./DPMGen

Few questions will be asked:

```
Give as seed a large float number (eg. 123456.):
Enter  P_lab(GeV/c),
Enter  N_Events
```

and the output file Background-micro.root will be created.

The file can be browsed using ROOT as usual. All the events are stored with a TParticle structure.

After the output file is created, one has only to propagate it into the transport model, thus inside pandaROOT.

Simulation inside PandaROOT

The interface between Dpm and PandaROOT consists on the PndDpmGenerator class. This is included in the PGen library and it has to be used in the simulation files, as shown in the following lines.

First of all one has to load the correct library in the simulation macro. So add, after the other library calls, the line:

```
gSystem->Load("libPGen");
```

Now, in the generator part of your macro, one has to write the following lines:

```
CbmPrimaryGenerator* primGen = new CbmPrimaryGenerator();
fRun->SetGenerator(primGen);

PndDpmGenerator* dpmGen = new PndDpmGenerator("$PATH/Background-
micro.root");
primGen->AddGenerator(dpmGen);
```

where Background-micro.root is our output root file, and $PATH is the full path.

And that is all.

One important thing: The number of processed events in the simulation macro should not exceed the event number in the evt file. In the contrary case the analysis will crash and the output file will be broken (no safe exit).

7. HOWTO loop over raw hits r1.1

Example: if you would like to get all detectors which fired in this event, and then loop over all these detector hits and print the energy which was deposited in the detector (which then will be converted to the ADC value in the digitized hits).

```
// get manager
 CbmRootManager* ioman = CbmRootManager::Instance();
 if ( ! ioman ) {
   cout << “error“ << endl;
}

// get input array (raw hits)
fPointArray = (TClonesArray*) ioman->GetObject("EmcPoint");
if ( ! fPointArray ) {
  cout << “error“ << endl;
}

// loop over raw hits
Int_t nPoints = fPointArray->GetEntriesFast();
for (Int_t iPoint=0; iPoint<nPoints; iPoint++) {
  point  = (CbmEmcPoint*) fPointArray->At(iPoint);
  cout << point->GetEnergyLoss()<< endl;
}
```

8. HOWTO loop over digitized hits r1.1

Example: if you would like to digitize your hits, i.e. for a detector which fired (in your data) you want the detector position from the geometry file, and then use in your further analysis these detector position instead of the true GEANT MC hit position.

```
// get the hit array
fDigiArray = new TClonesArray("CbmEmcHit");
ioman->Register("EmcHit","Emc",fDigiArray,kTRUE);

// read the geometry file
CbmEmcReader read(“emc_geo.dat“);

for(module=1; module<=read.GetMaxModules(); module++)
  for(row=1; row<=read.GetMaxRows(module); row++)
    for(crystal=1; crystal<=read.GetMaxCrystals(module,row);crystal++)
{
      // get the geometry data for 1 module
      DataG4 data = read.GetData(module,row,crystal);
      for(Int_t copy=1; copy<=20; copy++) {
        detId =  module*100000000 + row*1000000 + copy*10000 + crystal;
        emcX[detId] = data.posX;
        emcY[detId] = data.posY;
        emcZ[detId] = data.posZ;
        ...
```

Note that in this case the "copy" variable defines the index of the EMC subsection (i.e. the EMC is subdivided into 20 identical subsections which are rotated and mirrored).

9. PandaRoot Track Finder r1.3

Premlimnary thoughts on how to define a **local** track inside PandaRoot.

The idea is to have one **local** track structure per subdetector, and then all **local** tracks are fitted to a **global** track.

1. Conformal Map Track Finder

his is a README for the stand-alone version of `ftf`,

the fast track finder used by the Level-3 trigger system of the STAR experiment at RHIC.

It is based upon the **conformal mapping** algorithm.

Note that this is not a Kalman filter. The output of the track finder is one helix per track. The dE/dx can be calculated as truncated mean (if ADC values are given as input) but the helix is not modified. For both TPC and STT (as detectors with not much material) this approximation is sufficient. For muon detectors (material layers) and inhomogenous magnetic field a Kalman filter is needed. Another exeption for which a conformal mapping might not work are low pT tracks (maybe pT<50 MeV/c) for which not one single helix, but several helices of different curvature radii might be found by the conformal map.

INSTALL

download the distribution file which is called `ftf.1.6.tar.gz`

```
> tar xzvf ftf.1.6.tar.gz
> cd ftf-1.6
> make
```

The executable is called `ftf`.

MAIN LOOP

The track finder main loop is called in `sl3StandAlone.cc`.

Here the hit data are read from a data input file.

```
FILE* datafile = fopen("tpc_panda.dat", "r");
for ( i=0 ;  ; i++ ) {
  if ( fscanf ( datafile, "%f %f %f %d", &x,&y,&z,&row) == EOF ) break
;
  ...
```

and the track finder is called by using

```
  sectorTime = tracker.process ( );
```

Input Data

In this stand-alone version, input data can be detector hits in ascii format.

```
x y z padrow
```

Example:

```
7.57 -18.21 7.18 8
7.77 -18.63 7.35 8
7.98 -19.06 7.53 10
8.19 -19.48 7.70 11
...
```

The `padrow` number is needed to guide the track finder into a certain direction (away from the primary vertex).

Without a `padrow` number, there would be no preference direction, and the track finder would have to keep (for `n` hits) `n x (n-1)` hit-hit combinations when building the track and searching for the next hit. Processing time and memory amount would increase by a large factor.

However, a padrow number is related to the radius of a given hit.

It means: if we only have a data input file with `xyz` hits, we can generate easily `x,y,z,padrow` e.g. by using a `awk` command.

```
awk '{print ($2),($3),($4),int(((sqrt($2*$2+$3*$3)-15.0)/54.)*108.)}'
data_xyz.txt > data_xyz_padrow.dat
```

This generates a padrow number for the PANDA TPC.

The example data file `tpc_panda.dat` contains Panda TPC MC true hits for Geant3 for 1000 muons of pT=1 GeV/C (generated by Stefano Spataro with the TPC code of Sebastian Neubert and Christian Höppner).

The example data file `tpc_star.dat` contains STAR TPC cluster center-of-gravity (i.e. TPC cluster finder was already run) for a central Au+Au collision at 130 AGeV.

bField

It is currently set to _B_=2.0 Tesla (central region only).

If you want to change, you have to edit `FtfPara.cxx`.

```
void FtfPara::setDefaults(void){
  ...
  bField = 2.0F;
  ...
}
```

Differences between STAR and PANDA

Parameter	STAR	Panda

bField	0.5 T	2.0 T
inner radius	15 cm	60 cm
outer radius	40 cm	200 cm
number of padrows	45	54

Edit

Parameters

are optimized for STAR geometry yet, and will be optimized soon for Panda geometry.

Note that Panda TPC tracks are shorter than STAR TPC tracks by a factor of 4.6

see `FtfPara.cxx`:

```
   modRow          = 1    ;
   infoLevel       = 0 ;
   hitChi2Cut      = 500.F  ;
   goodHitChi2     = 100.F ;
   trackChi2Cut    = 250.F ;
   maxChi2Primary  = 0. ;
   segmentRowSearchRange = 1 ;
   trackRowSearchRange   = 3 ;
   dEdx              = 0     ;
   dEdxNTruncate     = 20    ;
   dphi              = 0.10F * modRow ;
   deta              = 0.10F * modRow ;
   dphiMerge         = 0.02F  ;
   detaMerge         = 0.02F  ;
   distanceMerge     = 2. ;
   etaMin            = -2.5F  ;
   etaMinTrack       = -2.2F  ;
   etaMax            =  2.5F  ;
   etaMaxTrack       =  2.2F  ;
   eventReset        =  1     ;
   getErrors         =  0     ;
   fillTracks        =  1     ;
   ghostFlag         =  0     ;
   goBackwards       =  0     ;
   goodDistance      =  1.F * modRow ;
   init              =  0 ;
   mergePrimaries    =  1    ;
   parameterLocation =  1    ;
   phiMin            =  (float)(-0.000001/toDeg)  ;
   phiMinTrack       =  (float)(-0.000001/toDeg)  ;
   phiMax            = (float)(360.2/toDeg)  ;
   phiMaxTrack       = (float)(360.2/toDeg)  ;
   maxDistanceSegment = 100.F * modRow ;
   minHitsPerTrack   = 5      ;
   nHitsForSegment   = 2      ;
   nEta              = 60     ;
   nEtaTrack         = 60     ;
   nPhi              = 20     ;
   nPhiTrack         = 60     ;
   nPrimaryPasses    = 1      ;
```

```
   nSecondaryPasses  = 0      ;
   vertexConstrainedFit = 0 ;
   rowInnerMost      = 1      ;
   rowOuterMost      = 45     ;
   rowStart          = 45     ;
   rowEnd            =  1     ;
   segmentMaxAngle   = 10.F/toDeg ;
   szFitFlag         = 1      ;
   xyErrorScale      = 1.0F   ;
   szErrorScale      = 1.0F   ;
   bField            = 2.0F   ;
   phiShift          = 0.0    ;
   ptMinHelixFit     = 100.F  ;
   rVertex           = 0.F    ;
   xVertex           = 0.F    ;
   yVertex           = 0.F    ;
   zVertex           = 0.F    ;
   dxVertex          = 0.005F ;
   dyVertex          = 0.005F ;
   phiVertex         = 0.F    ;
   maxTime           = 1.e18 ; // age of the Universe
```

Known bugs

Segmentation fault from glibc when too many tracks in one single loop.

Further information

Nucl.Instr.Meth. A499(2003)778 link

Nucl.Instr.Meth. A453(2000)397 link

Nucl.Instr.Meth. A380(1996)582 link

2. Track Structure

The following is a proposal for discussion.

```
trackID         // unique track number within the event
listOfHits
listOfTrackReps // = list of individual fits
nHits()           // number of detector hits assigned to this track
                // note: a hit can be assigned to more than one track

SetPrimaryTrackRep() // which is the fit we believe in?

parAtStart()    // gives (x,p,q/p) at first hit for primary TrackRep
parAtEnd()      //

parAtStart(int id) // gives (x,p,q/p) at TrackRep with index id
parAtEnd(int id)   //
```

```
nDedxHits       // number of hits used for the truncated mean dedx
calculation
nFitHits        // number of hits used for the helix fit
dedx            // deposited energy per unit length

chargeSign      // positive or negative charge

firstHit        // pointer to the first hit on track
lastHit         // pointer to the last hit on track
                // note: there are no pointers to any other hits
length          // in s-z plane
eventID         // Sebastian's (back-)pointer for pile-up events in TPC
```

Note that e.g. with such a model it is not possible to assign "outlier" hits to a track.

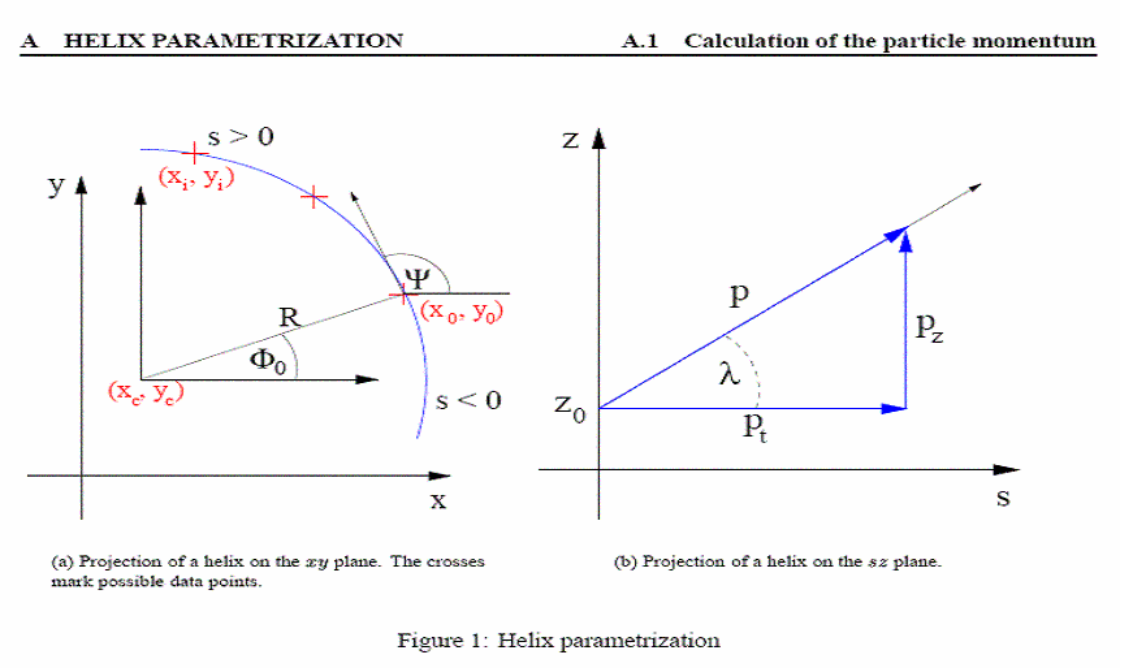
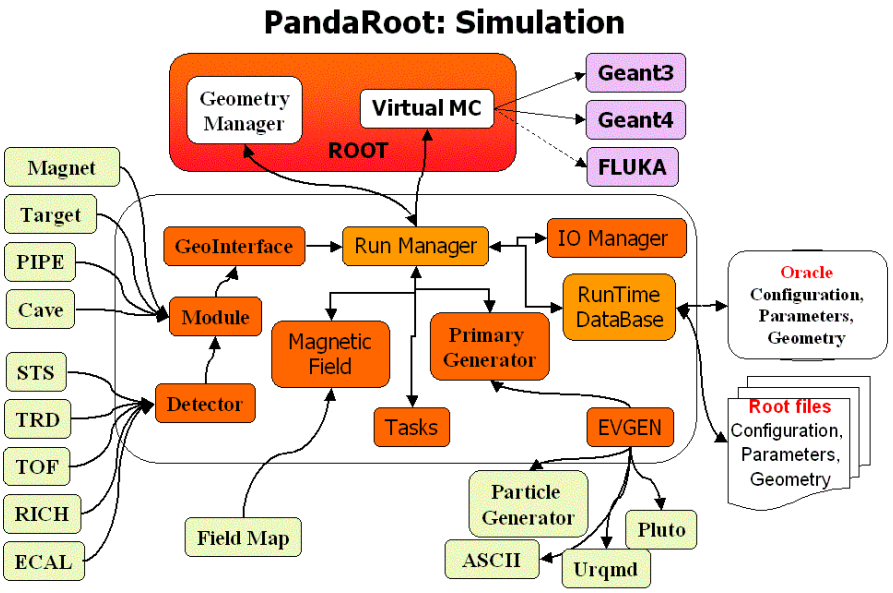
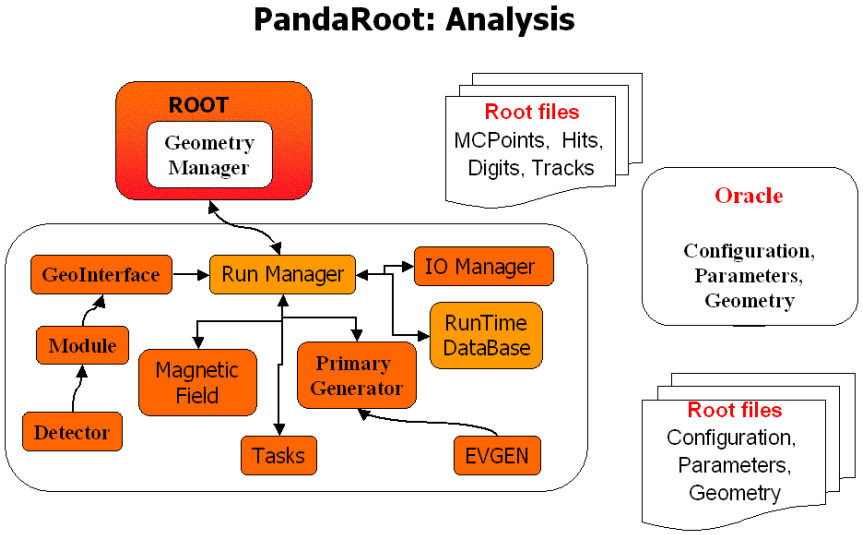


Figure 1: Helix parametrization

10. PANDARoot MODEL r1.1

Schematic description of dependencies inside PandaRoot.





11. PandaRoot B-field Map r1.3 (update needed)

In order to use B-field map, follow the steps (considering you already have the bfield directory among your software packages): 1. download the source files I attached to this post into `$VMCWORKDIR/bfield/'; 2. download one of the B-field map files (ASCII or ROOT) in the same diroctory; 3. compile PandaRoot framework; 4. insert in your simulation macro the following lines:

```
gSystem->Load("libBField");

... <some other code>

PFMS1 *fMagField = new PFMS1();
fMagField->Init("bfield.<extension>");
fRun->SetField(fMagField);
```

where can be `map' or `root'. The last three points should be implemented just before `fRun->Init();' line, since the first line should be at the top of your macro.

If everything is OK, then you should receive a message:

```
B-field map loaded with 2941101 values
```

just before initialization of the simulation run. If the value is different (meaning "0") or you got a message as:

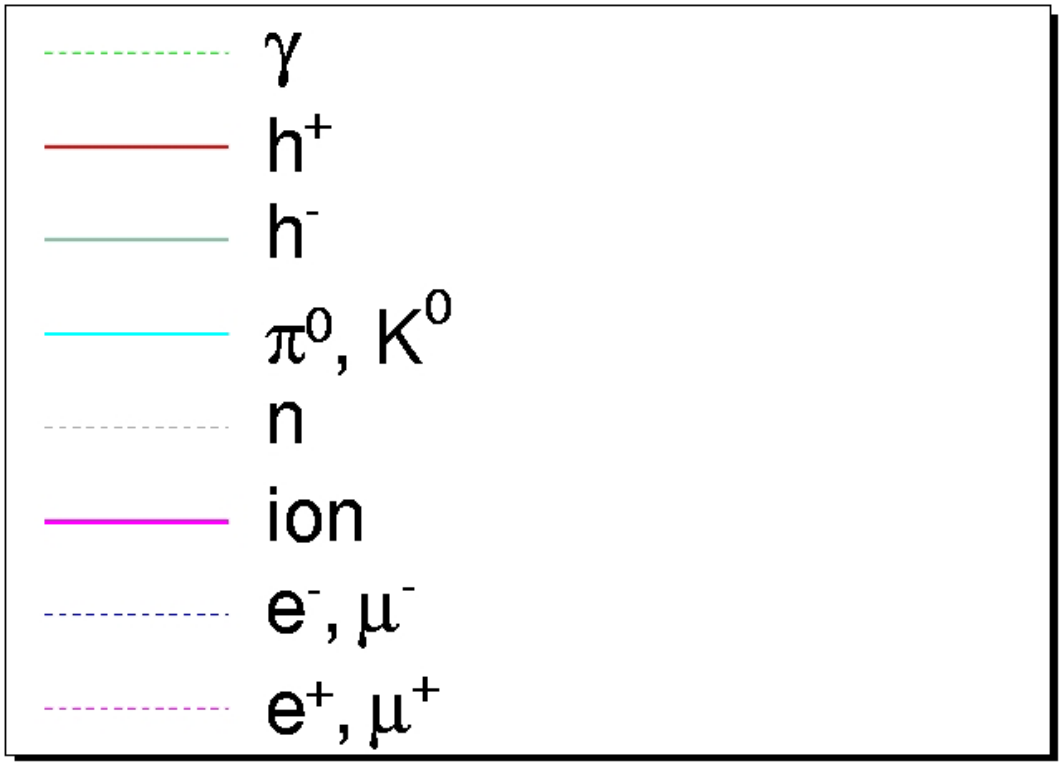
```
Error: B-field map file not found! The results are impredictable!
  To solve this problem, check if you have the B-field map file
  in the specific directory $VMCWORKDIR/bfield

it means your B-field map was not found in the specified directory.
Since the map file belongs to that specific package, be careful to put
it in that specific directory, be it the ASCII file or the ROOT file.
```

12. Visualization of trajectories r1.1

In order to visualize trajectories of particles one should modify the run macro in order to enable the storing of the trajectories during the simulation. This can be done in the following way :

```
...
fRun->SetStoreTraj(kTRUE);
fRun->Init();
CbmTrajFilter *trajFilter = CbmTrajFilter::Instance();
trajFilter->SetSteSizeCut(1.); // 1 cm
trajFilter->SetEnergyCut(1.); // Etot > 1 GeV
fRun->Run(1);
}
```

The trajectories will be stored as a branch in the output tree (GeoTracks). The color codes of the particles are unfortunately hardcoded and are illustrated on the next picture.



Example of the drawing macro

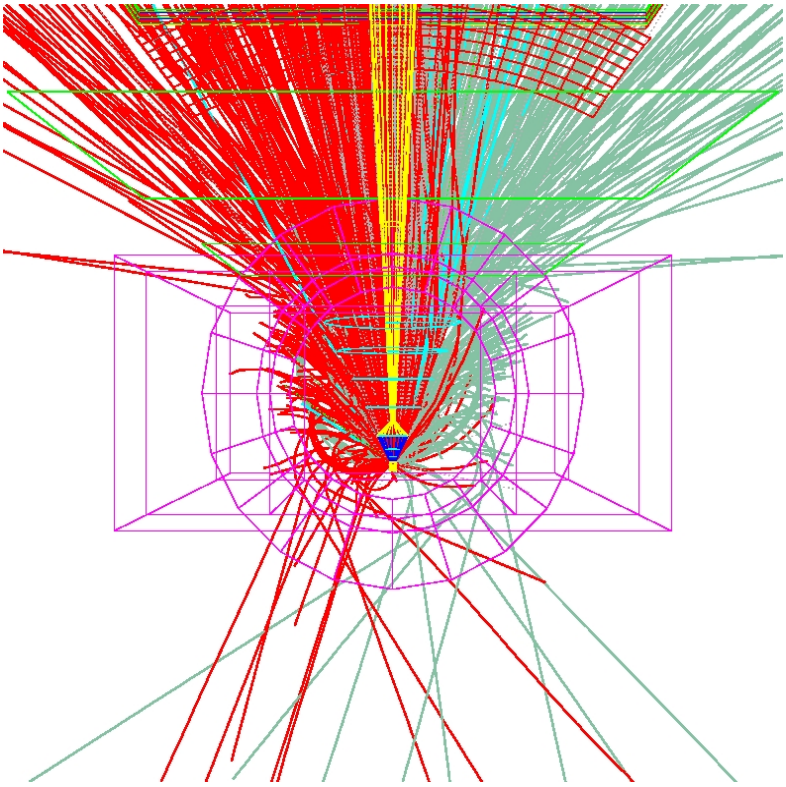
The following macro can be used to draw the trajectories from the output file.

```
gSystem->Load("libGeom.so");
TFile* f = new TFile("auaumbias.root");
TTree *t=f->Get("cbmsim") ;

TClonesArray *fT=new TClonesArray("TGeoTrack");
t->SetBranchAddress("GeoTracks",&fT) ;
TGeoManager *geoMan = (TGeoManager*) f->Get("CBMGeom");
TCanvas* c1 = new TCanvas("c1", "", 100, 100, 800, 800);

geoMan->GetVolume("magnet")->Draw("same");
TGeoTrack *tr;


for (Int_t j=0; j< t->GetEntriesFast(); j++) {
  t->GetEntry(j);
for (Int_t i=0; i<fT->GetEntriesFast(); i++) {
    tr=(TGeoTrack *)fT->At(i);
     tr->Draw("same");
}

 }
```

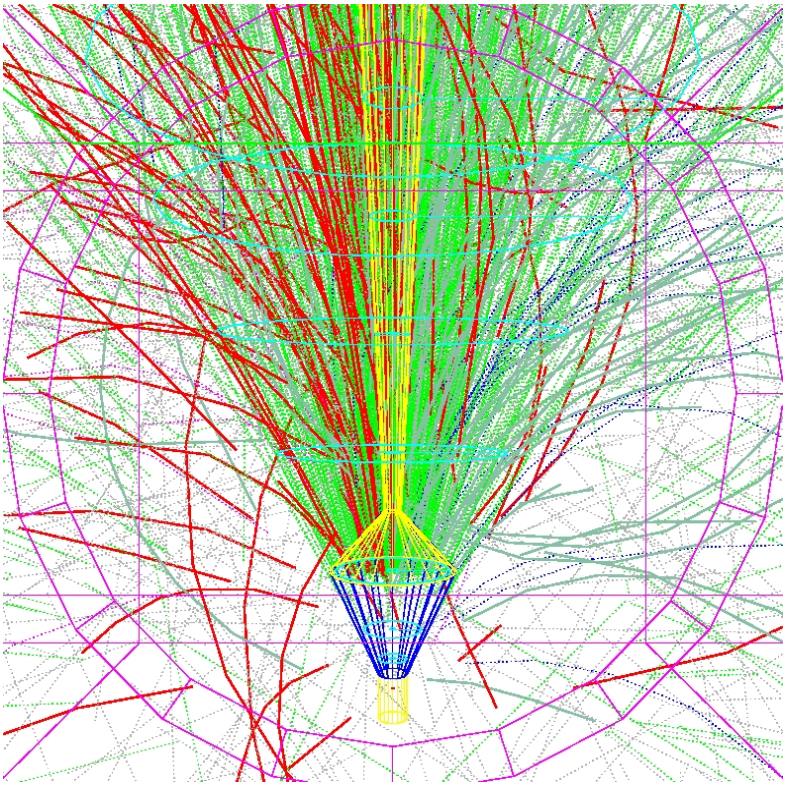
This will draw all tracks from all events. One can use the PDG code to select certain type of particles.

Examples of visualisation

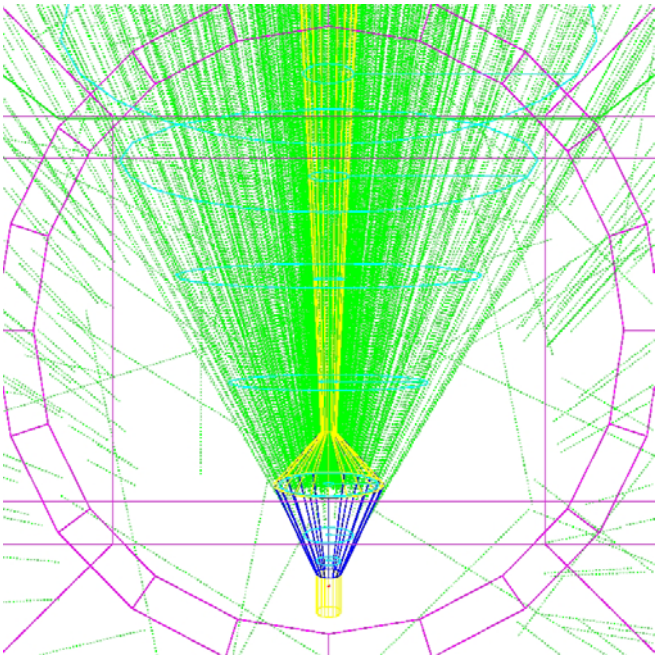
Primary particles with p > 10 MeV/c :



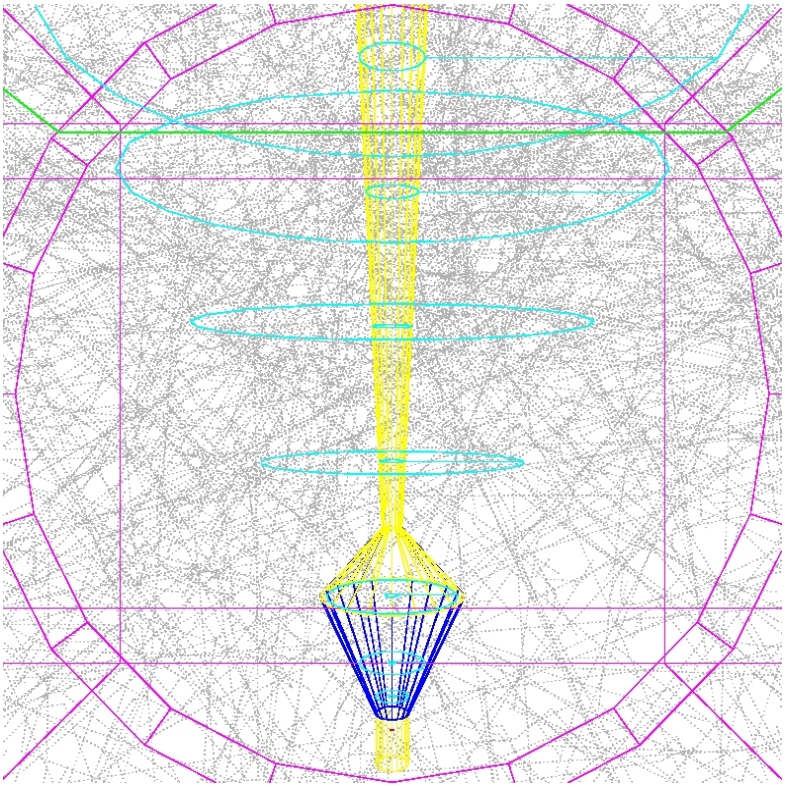
Secondary particles with p > 10 MeV/c :



Secondary gammas with p > 10 MeV/c :



Secondary neutrons with Ekin < 100 MeV :



13. Examples r1.1

Ex1. How to start a new detector

Step 1: Create a geometry

- **Location: pandaroot/geometry**
- **Extension: .geo**

```
//
// inner cylinder copy 1
//
// name
// name of mother volume
// volume type
// medium type
// startpoint x y z
// innerradius outerradius
// endpoint x y z
// translation
// rotation matrix
```

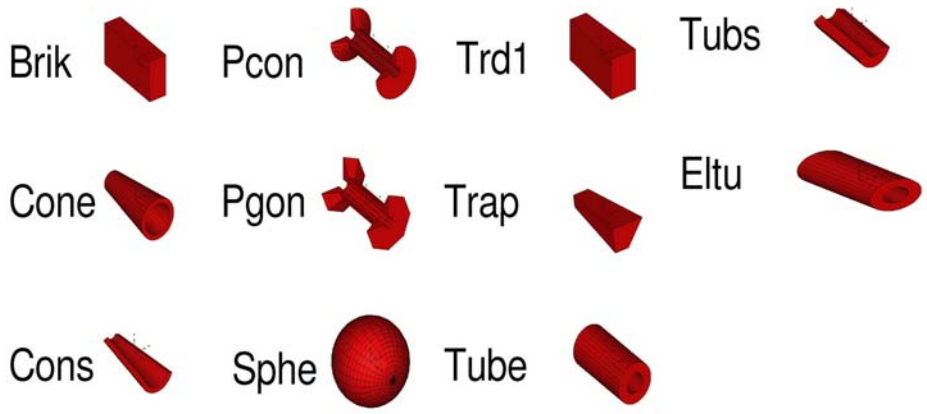
```
example1innerScintillator#1                    // name # copy
cave                                           // mother volume
TUBE                                           // shape
Silicon                                        // medium
0.000000    0.000000    -30.000000
10.000000   10.10000                           //shape params
0.000000    0.000000     30.000000
0.000000   0.000000   33.000000                 // translation
1.000000   0.000000   0.000000
0.000000   1.000000   0.000000                  // rotation
0.000000   0.000000   1.000000
```

- **Use copies when possible**
    - **Accelerate simulation**

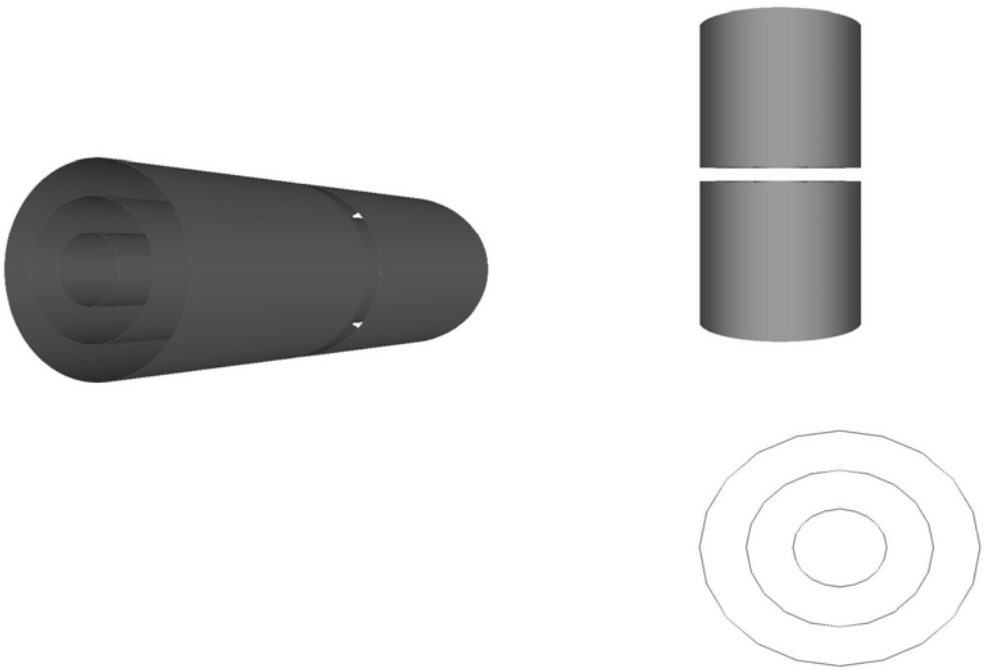
```
//
// inner cylinder copy 2
//
// name
// name of mother volume
// translation
// rotation matrix
```

```
example1innerScintillator#2                    // name # copy
cave                                           // mother volume

0.000000   0.000000   -33.000000                // translation

1.000000   0.000000   0.000000
0.000000   1.000000   0.000000                  // rotation
0.000000   0.000000   1.000000



example1middleScintillator#1
cave
TUBE
silicon
0.000000    0.000000    -30.000000
20.000000   20.10000
0.000000    0.000000     30.000000
0.000000   0.000000   33.000000
1.000000   0.000000   0.000000   0.000000   1.000000   0.000000   0.000000   0.000000   1.000000

example1middleScintillator#2
cave
0.000000   0.000000   -33.000000
1.000000   0.000000   0.000000   0.000000   1.000000   0.000000   0.000000   0.000000   1.000000

example1outerScintillator#1
cave
TUBE
silicon
0.000000    0.000000    -30.000000
30.000000   30.10000
0.000000    0.000000     30.000000
0.000000   0.000000   33.000000
1.000000   0.000000   0.000000   0.000000   1.000000   0.000000   0.000000   0.000000   1.000000

example1outerScintillator#2
cave
0.000000   0.000000   -33.000000
1.000000   0.000000   0.000000   0.000000   1.000000   0.000000   0.000000   0.000000   1.000000
```

- **Available shapes:**



- **Example Geometry:**



Step 2: Create a directory

- **Location: pandaroot/example**
- **All detectors are modular**
  - **Do not reuse other detectors code**
  - **Use framework's code**
    - **pandaroot/base**

- **pandaroot/geobase**
- **pandaroot/parbase**
- **pandaroot/mcstack**

Step 3: Adapt the autoconf system

- **Location: pandaroot/configure.ac**

```
AC_SUBST(LD_LIBRARY_PATH)

AC_CONFIG_FILES([
example/Makefile
Makefile
base/Makefile
geobase/Makefile
parbase/Makefile
mcstack/Makefile
passive/Makefile
field/Makefile
generators/Makefile
cbmg4/Makefile
stt1/Makefile
stt2/Makefile
mvd/Makefile
muo/Makefile
emc/Makefile
])

AC_OUTPUT
```

- **checks**
  - **compiler**
  - **linker**
  - **libraries**
- **creates configure script**

Step 4: Adapt the automake system

- **Location: pandaroot/Makefile.am**

```
if COND_HADES
  HADES_DIR=hades
endif

SUBDIRS = example geobase parbase base mcstack field generators passive\
                stt1 stt2 mvd muo emc $(GEANT4_DIR)
disthook:
```

```
find . -name '*Dict.cxx' | xargs rm -f
find . -name '*Dict.h' | xargs rm -f
```

- **Generates makefiles**
  - **clean**
  - **dist**
  - **install**
- **automatic dependency generation**
  - **dep = #include files**

Step 5: Create an automake file in the new directory

- **Location: yourdir/example/Makefile.am**

```
# where to find include files
INCLUDES = $(ROOTCFLAGS)
INCLUDES += -I$(top_srcdir)/geobase
...


# the name of the static library to build
lib_LTLIBRARIES = libExample.la
ibExample_la_LDFLAGS = -versioninfo 0:0:0
libExample_la_LIBADD =-L$(ROOTLIBDIR) $(ROOTLIBS)

# list of sources and headers
sources = PndGeoExample.cxx \
...


libExample_la_SOURCES = $(sources) PndExampleDict.cxx
include_HEADERS = $(patsubst %.cxx,%.h,$(sources))


# header file which should not go in includes dir
noinst_HEADERS = ExampleLinkDef.h


# additional files generated during the make process
CLEANFILES      = PndExampleDict.cxx \
                  PndExampleDict.h \
                  libExample.la


# description of how to obtain the dictionary file
PndExampleDict.cxx: $(include_HEADERS) ExampleLinkDef.h
$(ROOTCINT) -f $@ -c -DHAVE_CONFIG_H $(INCLUDES) $^
```

- **Compiler flags**
  - **include**
- **Library name**
- **Sources to use**
- **Which files to clean**

- **How to generate additional files**

Step 6: Create a linkdef file in that new directory

- **Location: yourdir/example/ExampleLinkDef.h**
- **Tells rootcint which files to add to dictionary**
- **Should list all your classes**

```
#ifdef __CINT__

#pragma link off all globals;
#pragma link off all classes;
#pragma link off all functions;

#pragma link C++ class PndGeoExample+;
#pragma link C++ class PndExamplePoint+;
#pragma link C++ class PndExample+;
#pragma link C++ class PndExampleHit+;
#pragma link C++ class PndExampleHitProducer+;

#endif
```

Additional Remarks on Steps 1-6

- **When adding a source file to the project**
  - **Update example/Makefile.am**
  - **Update example/ExampleLinkDef.h**
  - **Run pandaroot/reconf**
  - **Run make install**
- **When adding an include file in a source file**
  - **Automake takes care of everything**
  - **Run make install**

Step 7: Create simulation classes

- **Location: yourdir/example/PndGeoExample.h/cxx**
- **Interface to geometry file**
  - **derives from CbmGeoSet**

PndGeoExample.h

```
#ifndef PNDGEOEXAMPLE_H
#define PNDGEOEXAMPLE_H
```

```
#include "CbmGeoSet.h"

class  PndGeoExample : public CbmGeoSet
{
 public:
    // constructors and destructors
    PndGeoExample();
    ~PndGeoExample();


    // implementation of virtual functions from base class CbmGeoSet
    /** return the name for a specific module number */
    const char* getModuleName(Int_t number);

    /** return the name for a specific element number */
    const char* getEleName(Int_t number);

    /** return the number for a specific module name */
    Int_t getModNumInMod(const TString& name);


    // macro (defined by ROOT) to generate streamer functions
    ClassDef(PndGeoExample,0)
};

#include <sstream>

#endif
```

PndGeoExample.cxx

```
#include "PndGeoExample.h"

// macro (defined by ROOT) to generate streamer functions
ClassImp(PndGeoExample)

PndGeoExample::PndGeoExample()
{
    // Detector name
    fName="example";

    // Sectors are not used for panda, kept for backwards compatibility
//with HADES geometry
    maxSectors=0;

    // Just a single module is used for panda, kept for backwards
//compatibility with HADES geometry
    maxModules=1;
}

PndGeoExample::~PndGeoExample()
{
}

Int_t PndGeoExample::getModNumInMod(const TString& name)
```

```
{
    return 1;
}

const char* PndGeoExample::getModuleName(Int_t m)
{
  return "example";
}

const char* PndGeoExample::getEleName(Int_t m)
{
  return "ex";
}
```

- **Class to store geant data (MC points)**
  - **derives from CbmMCPoint**

PndExamplePoint.h

```
#ifndef PNDEXAMPLEPOINT_H
#define PNDEXAMPLEPOINT_H

// ROOT includes
#include "TVector3.h"

// pandaRoot includes
#include "CbmMCPoint.h"

class PndExamplePoint : public CbmMCPoint
{
 public:
    /** Default constructor **/
    PndExamplePoint();

    /** Constructor with arguments
      \param trackID       Index of MCTrack
      \param detID         Detector ID
      \param pos           Coordinates of hit [cm]
      \param entryPos      Entry coordinates of track [cm]
      \param exitPos       Exit Coordinates of track [cm]
      \param mom           Momentum of track [GeV]
      \param tof           Time since event start [ns]
      \param length        Track length since creation [cm]
      \param eLoss         Energy deposit [GeV]
    **/
    PndExamplePoint(Int_t trackID, Int_t detID, TVector3 pos,
                TVector3 entryPos, TVector3 exitPos,
                TVector3 mom, Double_t tof, Double_t length,
                Double_t eLoss);

    /** Copy constructor **/
    PndExamplePoint(const PndExamplePoint& point) { *this = point; };

    /** Destructor **/
    virtual ~PndExamplePoint();
```

```
    /** Accessors **/
    TVector3 GetEntryPoint() const { return fEntryPoint; }
    TVector3 GetExitPoint()  const { return fExitPoint;  }

    /** Modifiers **/
    void SetEntryPoint(TVector3 newEntryPoint) {fEntryPoint =
newEntryPoint;}
    void SetExitPoint(TVector3 newExitPoint)  {fExitPoint =
newExitPoint;  }

    /** Output to screen **/
    virtual void Print(const Option_t* opt) const;

 protected:
    // exit coordinates
    TVector3 fExitPoint;
    // entry coordinates
    TVector3 fEntryPoint;

    ClassDef(PndExamplePoint,1)
};

#include <iostream>

#endif
```

PndExamplePoint.cxx

```
#include "PndExamplePoint.h"

PndExamplePoint::PndExamplePoint()
    : CbmMCPoint(),
      fExitPoint(),
      fEntryPoint()
{
}

PndExamplePoint::PndExamplePoint(Int_t trackID, Int_t detID, TVector3
pos,
                        TVector3 entryPoint, TVector3 exitPoint,
                        TVector3 mom, Double_t tof, Double_t length,
                        Double_t eLoss)
    : CbmMCPoint(trackID, detID, pos, mom, tof, length, eLoss),
      fExitPoint(exitPoint),
      fEntryPoint(entryPoint)
{
}

PndExamplePoint::~PndExamplePoint()
{
}

void PndExamplePoint::Print(const Option_t* opt) const
{
    std::cout << "PndExamplePoint: " << endl
            << " Track index                : " << fTrackID << endl
```

```
            << " Detector ID                : " << fDetectorID << endl
            << " Point coordinates          : (" << fX << ", " << fY <<
", " << fZ << ") [cm]" << endl
            << " Momentum components        : (" << fPx<< ", " << fPy<<
", " << fPz << ") [GeV]" << endl
            << " Time since event start     : " << fTime << " [ns]" <<
endl
            << " Track length since creation : " << fLength << " [cm]"
<< endl
            << " Energy loss at this point   : " << fELoss << " [GeV]"
<< endl;
}

ClassImp(PndExamplePoint)
```

- **Interface to virtual Monte-Carlo**
  - **derives from CbmDetector**

PndExample.h

```
#ifndef CBMSTT_H
#define CBMSTT_H

// ROOT includes
#include "TClonesArray.h"
#include "TVector3.h"

// pandaRoot includes
#include "CbmDetector.h"
#include "CbmRun.h"

class PndExamplePoint;
class CbmVolume;

class PndExample : public CbmDetector
{
 public:
    /** Default constructor */
    PndExample();

    /** Standard constructor.
      \param name    detector name
      \param active  sensitivity flag
    */
    PndExample(const char* name, Bool_t active = kTRUE, Int_t verbose =
0);

    /** Destructor */
    virtual ~PndExample();

    /** Defines the action to be taken when a step is inside the
      active volume. Creates PndExamplePoints and adds them to the
      collection.
```

```
      \param vol  Pointer to the active volume
    */
    virtual Bool_t ProcessHits(CbmVolume* vol = 0);

    /** If verbosity level is set, print hit collection at the
      end of the event and resets it afterwards */
    virtual void EndOfEvent();

    /** Registers the hit collection with the ROOT manager */
    virtual void Register();

    /** Accessor to the hit collection */
    virtual TClonesArray* GetCollection(Int_t iColl) const;

    /** Screen output of hit collection */
    virtual void Print() const;

    /** Clears the point collection */
    virtual void Reset();

    /** Constructs the example geometry */
    virtual void ConstructGeometry();

 private:
    /** Track information to be stored during tracking through a volume
*/
    Int_t          fTrackID;      // track index
    Int_t          fVolumeID;     // volume id
    TVector3       fPos;          // wire position in global frame
    TVector3       fEntryPos;     // entry position in global frame
    TVector3       fExitPos;      // exit position in global frame
    TVector3       fMom;          // momentum
    Double32_t     fTime;         // time
    Double32_t     fLength;       // length
    Double32_t     fEnergyLoss;   // energy loss

    TClonesArray*  fExamplePointCollection;  // Pointer to hit
collection

    /** Adds a CbmTrdPoint to the HitCollection */
    PndExamplePoint* AddPoint(Int_t trackID, Int_t detID, TVector3 pos,
                       TVector3 entryPos, TVector3 exitPos,
                       TVector3 mom, Double_t time, Double_t length,
Double_t eLoss);

    /** Resets the private members for the track parameters */
    void ResetParameters();

    ClassDef(PndExample,1)
};

#include "TVirtualMC.h"
#include "TLorentzVector.h"
#include "CbmRootManager.h"
#include "CbmGeoLoader.h"
#include "CbmGeoInterface.h"
#include "PndGeoExample.h"
```

```
#include "PndExamplePoint.h"

#endif
```

PndExample.cxx

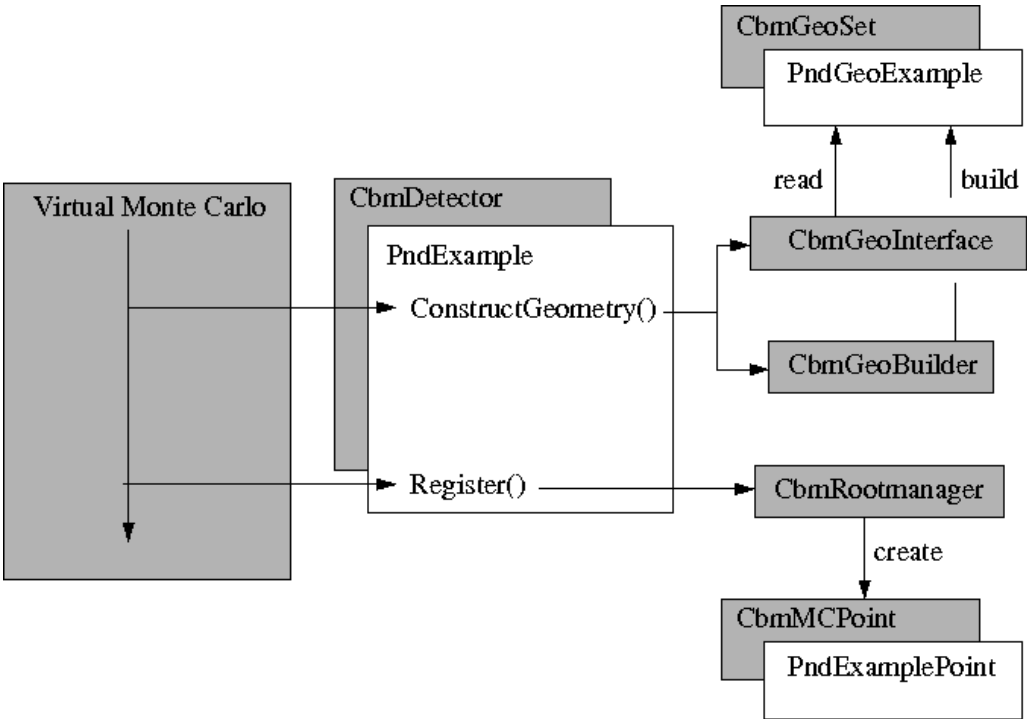
```
#include "PndExample.h"

PndExample::PndExample() : CbmDetector()
{
    fExamplePointCollection = new TClonesArray("PndExamplePoint");
    fVerboseLevel = 1;

    ResetParameters();
}

PndExample::PndExample(const char* name, Bool_t active, Int_t verbose)
    : CbmDetector(name, active)
{
    fExamplePointCollection = new TClonesArray("PndExamplePoint");
    fVerboseLevel = verbose;
}

PndExample::~PndExample()
{
    if (fExamplePointCollection)
    {
      fExamplePointCollection->Delete();
      delete fExamplePointCollection;
    }
}

Bool_t PndExample::ProcessHits(CbmVolume* vol)
{
    TLorentzVector tmp;

    if (gMC->TrackCharge() != 0.)
    {
      if (gMC->IsTrackEntering())
      {
          // Set parameters at entrance of volume. Reset ELoss.
          fEnergyLoss  = 0.;
          fTime   = gMC->TrackTime() * 1.0e09;
          fLength = gMC->TrackLength();

          gMC->TrackPosition(tmp);
          fEntryPos = tmp.Vect();

          gMC->TrackMomentum(tmp);
          fMom = tmp.Vect();
      }

      // Sum energy loss for all steps in the active volume
      fEnergyLoss += gMC->Edep();

      if (gMC->IsTrackExiting())
      {
```

```
            // Create PndExamplePoint at exit of active volume
            fTrackID  = gMC->GetStack()->GetCurrentTrackNumber();
            fVolumeID = vol->getMCid();
            gMC->TrackPosition(tmp);
            fExitPos = tmp.Vect();
            fPos.SetXYZ(0., 0., 0.);

            AddPoint(fTrackID, fVolumeID, fPos, fEntryPos, fExitPos,
                     fMom, fTime, fLength, fEnergyLoss);

            ResetParameters();
          }
        }

        return kTRUE;
}

void PndExample::EndOfEvent()
{
        if (fVerboseLevel) Print();
        fExamplePointCollection->Clear();
}

void PndExample::Register()
{
        CbmRootManager::Instance()->Register("ExamplePoint", "Example",
                                             fExamplePointCollection, kTRUE);
}

TClonesArray* PndExample::GetCollection(Int_t iColl) const
{
        if (iColl == 0)
          return fExamplePointCollection;
        else return NULL;
}

void PndExample::Print() const
{
        Int_t nPoints = fExamplePointCollection->GetEntriesFast();

        cout << "-I- CbmExample: " << nPoints
             << " points registered in this event."  << endl;

        if (fVerboseLevel > 1)
        {
          for (Int_t i = 0; i < nPoints; i++)
          {
              (*fExamplePointCollection)[i]->Print();
          }
        }
}

void PndExample::Reset()
{
        fExamplePointCollection->Clear();
        ResetParameters();
}
```

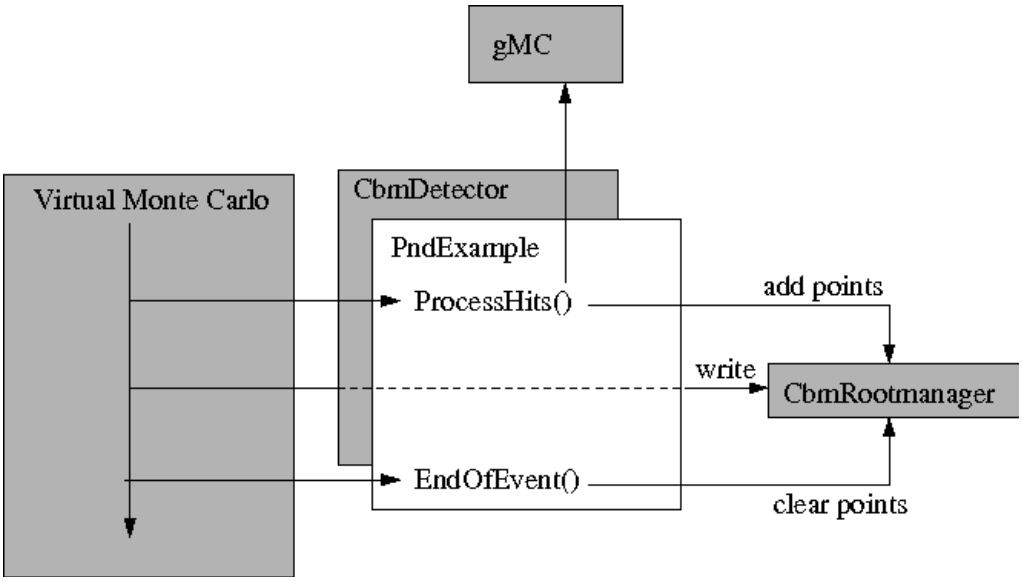
```
void PndExample::ConstructGeometry()
{
    CbmGeoLoader *geoLoad = CbmGeoLoader::Instance();

    // geo interface is the class that reads our .geo file
    CbmGeoInterface *geoFace = geoLoad->getGeoInterface();

    // geo builder is the class that constructs the geometry
    CbmGeoBuilder *geoBuild = geoLoad ->getGeoBuilder();

    PndGeoExample *exampleGeo  = new PndGeoExample();

    exampleGeo->setGeomFile(GetGeometryFileName());

   // read in example.geo
    geoFace->addGeoModule(exampleGeo);

    Bool_t rc = geoFace->readSet(exampleGeo);

    // create the geometry
    if (rc) exampleGeo->create(geoBuild);

    TList* volList = exampleGeo->getListOfVolumes();

    // Set active/inactive
    ProcessNodes(volList);
}


PndExamplePoint* PndExample::AddPoint(Int_t trackID, Int_t detID,
                                      TVector3 pos,
                                      TVector3 entryPos,
                                      TVector3 exitPos,
                                      TVector3 mom, Double_t time,
                                      Double_t length, Double_t eLoss)
{
    // Get a reference to the hit collection
    TClonesArray& points = *fExamplePointCollection;

    // get the number of hits already present
    Int_t size = points.GetEntriesFast();

    // create a new hit after the last one present in the collection
    return new(points[size]) PndExamplePoint(trackID, detID, pos,
                                             entryPos, exitPos,
                                             mom, time, length, eLoss);
}

void PndExample::ResetParameters()
{
    fTrackID = fVolumeID = 0;
    fTime = fLength = fEnergyLoss = 0;

    fPos.SetXYZ(0., 0., 0.);
    fEntryPos.SetXYZ(0., 0., 0.);
    fExitPos.SetXYZ(0., 0., 0.);
```

```
    fMom.SetXYZ(0., 0., 0.);
};

ClassImp(PndExample)
```

- **Summary**
  - **Initialization:**



  - **Running:**



- **Run the simulation:**
  - **Location: –> pandaroot/macro/run/simExample.C**

```
{
  // Load basic libraries
  gROOT->LoadMacro("$VMCWORKDIR/gconfig/basiclibs.C");
  basiclibs();

  // Load this example libraries
  gSystem->Load("libGeoBase");
  gSystem->Load("libParBase");
  gSystem->Load("libBase");
  gSystem->Load("libMCStack");
  gSystem->Load("libGen");
  gSystem->Load("libField");
  gSystem->Load("libPassive");
  gSystem->Load("libExample");

  // Create a monte carlo
  // --------------------
  CbmRunSim *fRun = new CbmRunSim();

  // Set the MC version used
  // -----------------------
  fRun->SetName("TGeant3");

  // Choose an output file name
  fRun->SetOutputFile("exampleSimOutput.root");

  // Set material file name
  fRun->SetMaterials("media_pnd.geo");
  fRun->SetStoreTraj(kTRUE);

  // Create and add detectors
  // ------------------------
  CbmModule *Cave= new CbmCave("CAVE");
  Cave->SetGeometryFileName("pndcave.geo");
  fRun->AddModule(Cave);

  // kTRUE : Active detector
  // kFALSE: The process hit of this detector will not be called
  CbmDetector *Example = new PndExample("Example", kTRUE);
  Example->SetGeometryFileName("example.geo");
  //add the example
  fRun->AddModule(Example);

  // Create and set event generator
  // ------------------------------
  CbmPrimaryGenerator* primGen = new CbmPrimaryGenerator();
  fRun->SetGenerator(primGen);

  CbmParticleGenerator* partGen = new CbmParticleGenerator(2212, 1,
                                                           0.3, 0.3,
                                                           0.3);
  primGen->AddGenerator(partGen);
```

```
  // Create and set magnetic field
  // ---------constant in this case--------------------
  CbmFieldConst *magField=new CbmFieldConst();
  magField->SetField(0.,0.,20.); // values are in kG

  // MinX, MinY, MinZ, MaxX, MaxY ,MaxZ
  magField->SetFieldRegion(-50,50,-50,50,-100,100);//values are in cm
  fRun->SetField(magField);

  // Initalize the monte carlo
  // -------------------------
  fRun->Init();

  // Transport nEvents
  // -----------------
  Int_t nEvents = 30;
  fRun->Run(nEvents);
   }
```

Step 8: Create reconstruction classes

- **Class to store reconstructed hits**
  - **Derives from CbmHit**

PndExampleHit.h

```
#ifndef PNDEXAMPLEHIT_H
#define PNDEXAMPLEHIT_H 1

#include "TVector3.h"
#include "CbmHit.h"

class PndExampleHit : public CbmHit
{

 public:
  /** Default constructor */
  PndExampleHit();

  /** Standard constructor
      \param detID     Detector unique volume ID
      \param pos       Position coordinates [cm]
      \param dpos      Errors in position coordinates [cm]
      \param index     Index of corresponding MCPoint
      \param hitLost   Flag is true whenever hit was lost by
inefficiency
  */
  PndExampleHit(Int_t detID, TVector3& pos, TVector3& dpos,
                Int_t index, Int_t hitLost);
  PndExampleHit(Int_t detID, TVector3& pos, TVector3& dpos, Int_t
                index);
```

```
  /** Destructor */
  virtual ~PndExampleHit();

  /** Output to screen */
  virtual void Print(const Option_t* opt = 0) const;

  /** Accessors */
  Bool_t HitWasLost() const {return fWasLost; }

  /** Modifiers */
  void LoseHit(Bool_t newState = kTRUE) {fWasLost = newState;}

 private:
  Bool_t fWasLost;

  ClassDef(PndExampleHit,1);
};

#include <iostream>

#endif
```

PndExampleHit.cxx

```
#include "PndExampleHit.h"

PndExampleHit::PndExampleHit()
    : CbmHit()
{
    fWasLost = false;
}

PndExampleHit::PndExampleHit(Int_t detID, TVector3& pos,
                             TVector3&  dpos, Int_t index,
                             Int_t wasLost)
  : CbmHit(detID, pos, dpos, index)
{
    fWasLost = wasLost;
}

PndExampleHit::PndExampleHit(Int_t detID, TVector3& pos,
                             TVector3& dpos, Int_t index)
  : CbmHit(detID, pos, dpos, index)
{
    fWasLost = false;
}

PndExampleHit::~PndExampleHit()
{
}

void PndExampleHit::Print(const Option_t* opt) const
{
    std::cout << "PndExampleHit: " << std::endl
```

```
            << " Detector unique identifier       : " << fDetectorID <<
std::endl
            << " Position of hit                  : (" << fX << ", " <<
fY << ", " << fZ << ") [cm]" << std::endl
            << " Errors of position               : (" << fDx << ", " <<
fDy << ", " << fDz << ") [cm]" << std::endl
            << " Index of CbmMCPoint for this hit : " << fRefIndex <<
std::endl
            << " Hit lost due to ineff.           : " << fWasLost <<
std::endl;
}

ClassImp(PndExampleHit)
```

- **Task to reconstruct hits = to covert MC point to "real" hit**
  - **Derives from TTask**
    - **Init()**
    - **Exec()**
    - **Finish()**

PndExampleHitProducer.h

```
#include "TRandom.h"

class TClonesArray;

class PndExampleHitProducer : public CbmTask
{
 public:
    /** Default constructor */
    PndExampleHitProducer();

    /** Standard constructor */
    PndExampleHitProducer(Double_t positionResolution, Double_t
efficiency = 1., Int_t verbose = 0);

    /** Destructor */
    ~PndExampleHitProducer();

    /** Virtual method Init */
    virtual InitStatus Init();

    /** Virtual method Exec */
    virtual void Exec(Option_t* opt);

    /** Output to screen */
    void Print() const;

 private:
    /** Pointer to input array of PndExamplePoints */
    TClonesArray* fExamplePointCollection;
```

```
    /** Pointer to output array of PndExampleHits */
    TClonesArray* fExampleHitCollection;

    Double_t fPositionResolution;
    Double_t fEfficiency;
    Int_t fVerboseLevel;

    ClassDef(PndExampleHitProducer,1);
};

#include "TRandom.h"

#include "CbmRootManager.h"

#include "PndExamplePoint.h"
#include "PndExampleHit.h"
#endif
```

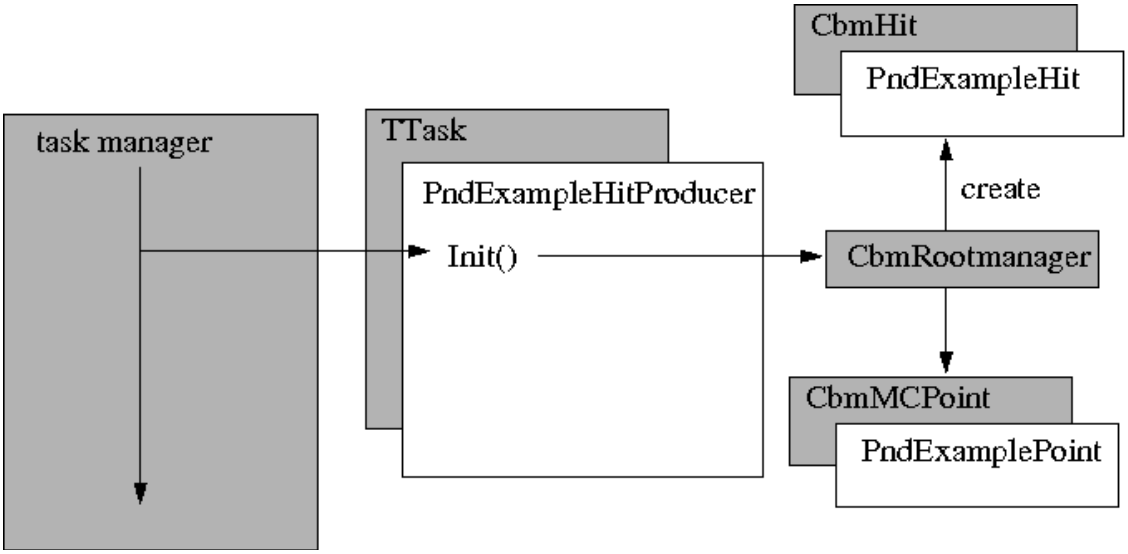
PndExampleHitProducer.cxx

```
#include "PndExampleHitProducer.h"

PndExampleHitProducer::PndExampleHitProducer()
    :  CbmTask("Example Hit Producer")
{
    fPositionResolution = 0;
    fEfficiency = 1.;
    fVerboseLevel = 0;
}

PndExampleHitProducer::PndExampleHitProducer(Double_t
positionResolution, Double_t efficiency, Int_t verbose)
    :  CbmTask("Example Hit Producer")
{
    fPositionResolution = positionResolution;
    fEfficiency = efficiency;
    fVerboseLevel = verbose;
}

PndExampleHitProducer::~PndExampleHitProducer()
{
}

InitStatus PndExampleHitProducer::Init()
{
    // Get RootManager
    CbmRootManager* ioman = CbmRootManager::Instance();
    if ( ! ioman )
    {
      cout << "-E- PndExampleHitProducer::Init: "
           << "RootManager not instantiated!" << endl;
      return kFATAL;
    }

    // Get input array
```

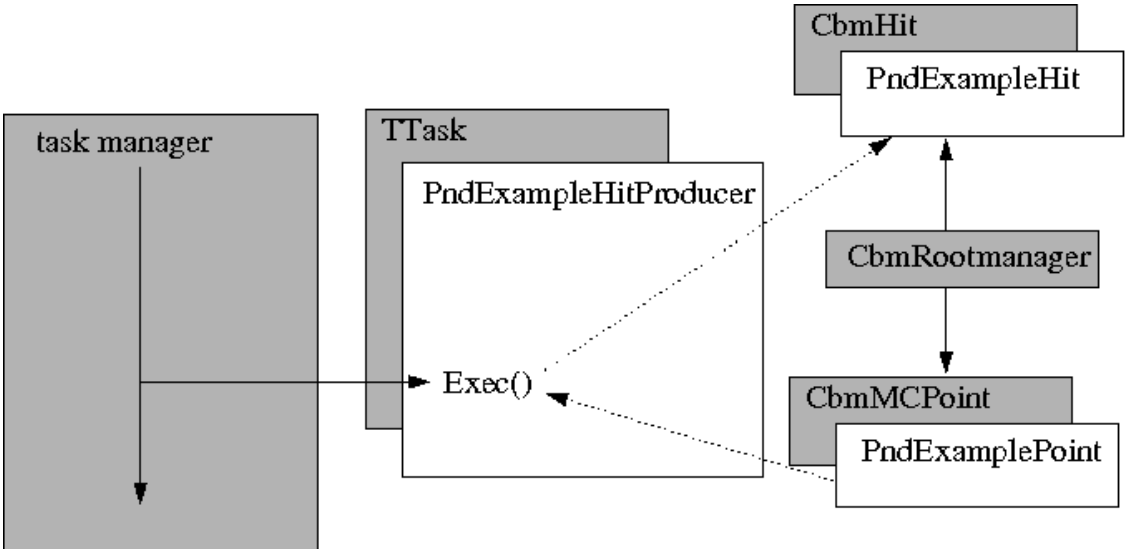
```
    fExamplePointCollection =
                      (TClonesArray*) ioman->GetObject("ExamplePoint");

    if (!fExamplePointCollection)
    {
      cout << "-W- PndExampleHitProducer::Init: "
           << "No EXAMPLEPoint collection!" << endl;
      return kERROR;
    }

    // Create and register output array
    fExampleHitCollection = new TClonesArray("PndExampleHit");
    ioman->Register("ExampleHit", "Example",
                    fExampleHitCollection, kTRUE);

    return kSUCCESS;
}

void PndExampleHitProducer::Exec(Option_t* opt)
{
    // Reset output array
    if (!fExampleHitCollection)
      Fatal("Exec", "No hit collection");

    fExampleHitCollection->Clear();

    // Declare some variables
    PndExamplePoint *point = NULL;

    Int_t detID = 0,       // Detector ID
          trackID = 0;     // Track index

    TVector3 pos, dpos;       // Position and error vectors

    // Loop over ExamplePoints
    Int_t nPoints = fExamplePointCollection->GetEntriesFast();

    for (Int_t iPoint = 0; iPoint < nPoints; iPoint++)
    {
      point = (PndExamplePoint*) fExamplePointCollection->At(iPoint);

      if ( ! point)
          continue;

      // Detector ID
      detID = point->GetDetectorID();

      // MCTrack ID
      trackID = point->GetTrackID();

      // Determine hit position
      TVector3 entryPos = point->GetEntryPoint(),
               exitPos = point->GetExitPoint(),
               dpos(fPositionResolution, fPositionResolution,
fPositionResolution),
          pos(((entryPos.X() + exitPos.X()) / 2.) + gRandom->Gaus(0.,
fPositionResolution),
```

```
            ((entryPos.Y() + exitPos.Y()) / 2.) + gRandom->Gaus(0.,
fPositionResolution),
            ((entryPos.Z() + exitPos.Z()) / 2.) + gRandom->Gaus(0.,
fPositionResolution));

      if (gRandom->Rndm() < fEfficiency)
      {
          new ((*fExampleHitCollection)[iPoint]) PndExampleHit(detID,
pos, dpos, iPoint, 0);
      }
      else
      {
          new ((*fExampleHitCollection)[iPoint]) PndExampleHit(detID,
pos, dpos, iPoint, 1);
      }
    }   // Loop over MCPoints

    if (fVerboseLevel)
    {
      Print();
    }
}

void PndExampleHitProducer::Print() const
{
    Int_t nHits = fExampleHitCollection->GetEntriesFast();

    cout << "-I- CbmExampleHitProducer: " << nHits
         << " hits registered in this event."  << endl;

    if (fVerboseLevel > 1)
    {
      for (Int_t i=0; i < nHits; i++)
      {
          (*fExampleHitCollection)[i]->Print();
      }
    }
}

ClassImp(PndExampleHitProducer)
```

- **Summary of reconstruction**
  - **Initialization:**



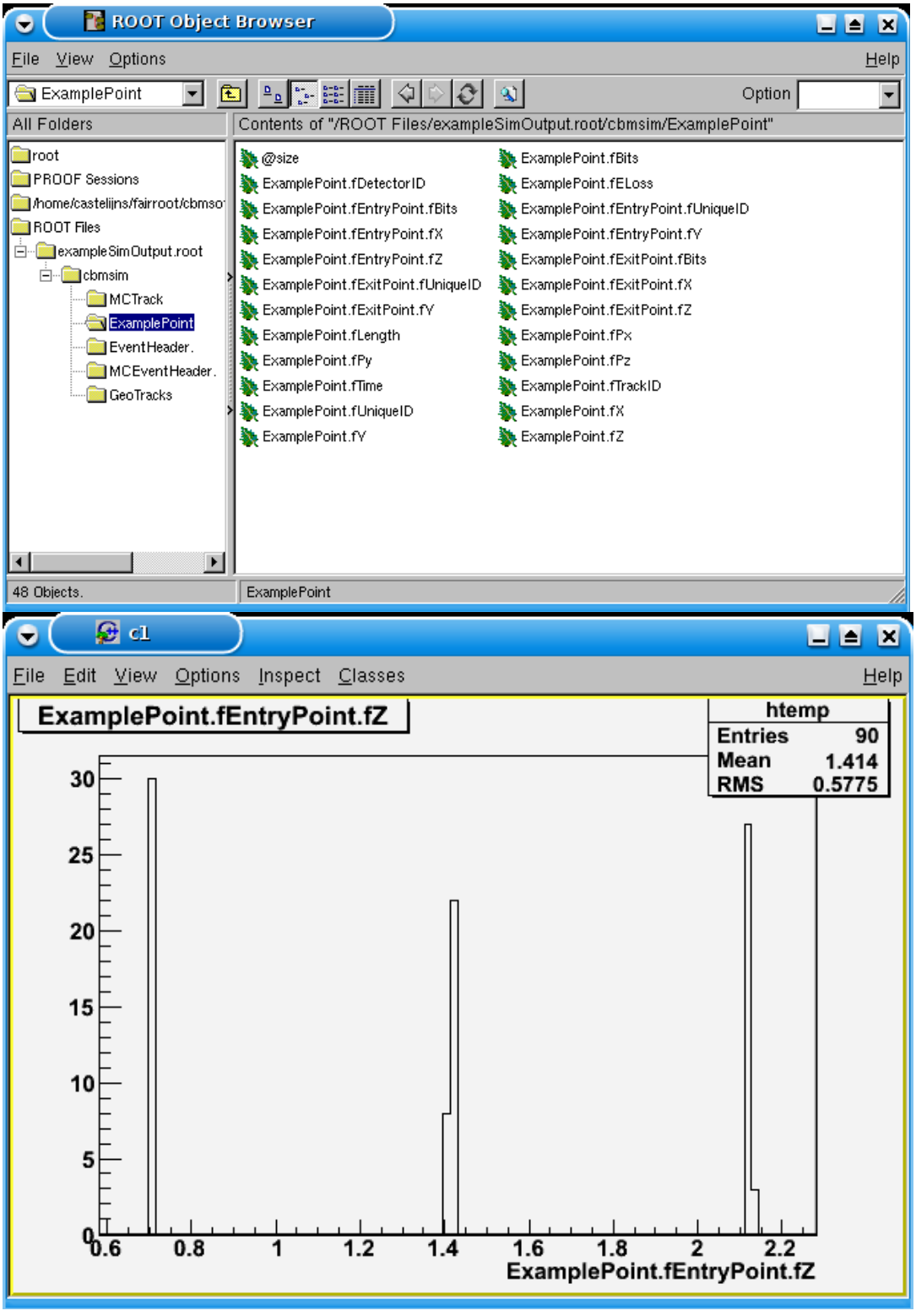
  - **Running:**



- **Run the reconstruction:**
  - **Location: –> pandaroot/macro/run/recoExample.C**

```
{
  // Load basic libraries
  gROOT->LoadMacro("$VMCWORKDIR/gconfig/basiclibs.C");
  basiclibs();

  // Load this example libraries
  gSystem->Load("libGeoBase");
  gSystem->Load("libParBase");
  gSystem->Load("libBase");
  gSystem->Load("libMCStack");
  gSystem->Load("libGen");
  gSystem->Load("libField");
  gSystem->Load("libPassive");
  gSystem->Load("libExample");

  Int_t iVerbose = 1;
  Int_t nEvents = 30;

  TString inFile = "exampleSimOutput.root";
  TString outFile = "exampleRecoOutput.root";

  // Reconstruction run
  // ------------------
  CbmRunAna *fRun = new CbmRunAna();
  fRun->SetInputFile(inFile);
  fRun->SetOutputFile(outFile);

  // Produce hits
  // ------------
  PndExampleHitProducer* exampleHitProducer =
                                  new PndExampleHitProducer(0., 1., 1);
  fRun->AddTask(exampleHitProducer);

  // Intialise and run
  fRun->Init();
  fRun->Run(0, nEvents);

  cout << "finished okay" << endl;
}
```

View the simulated/reconstructed data



Useful script to compile, install all the files

- **./createDetector.sh** ***path to your pandaroot installation***

```
#!/bin/bash

CBM_BASE_DIR="$1"
TUTORIAL_DIR="$PWD"

echo This script will create an example detector

echo -n "Checking your pandaRoot installation ...."
if [ ! -d $CBM_BASE_DIR ]
then
    echo " failed."
    echo "couldn't find the $CBM_BASE_DIR directory"
    exit
else
    echo " success."
fi
echo -n "Still checking your pandaRoot installation ...."
if [ ! -d $CBM_BASE_DIR/../build ]
then
    echo " failed."
    echo "couldn't find the build directory, the script expects at
least the following structure:"
    echo "cbmsoft"
    echo "cbmsoft/pandaroot"
    echo "cbmsoft/build"
    exit
else
    echo " success."
fi


# check if project exists
echo -n Creating project directory ....
if [ ! -d $CBM_BASE_DIR/example ]
then
    mkdir $CBM_BASE_DIR/example

    # check if directory can be created
    if [ ! -d $CBM_BASE_DIR/example ]
    then
        echo " failed."
      echo "Could not create a directory $CBM_BASE_DIR/example"
      exit
    else
      echo " success."
      NAME_CORRECT="kTRUE"
    fi
else
    echo " failed."
    echo "The example project directory already exists, exiting"
    exit
```

```
fi

echo -n Updating main Makefile.am ...
# check for existance main makefile.am
if [ -f $CBM_BASE_DIR/Makefile.am ]
then
    # check if example is already on the makefiles line
    grep "example" $CBM_BASE_DIR/Makefile.am > $TUTORIAL_DIR/log 2>
$TUTORIAL_DIR/err
    if [[ "$?" == "0" ]]
    then
      echo " warning."
      echo "Makefile.am already setup for example"
    else
        # find the sub dir line and add project subdir
      sed "/SUBDIRS/s/=/= example/g" $CBM_BASE_DIR/Makefile.am >
/tmp/Makefile.tmp

        # check if it worked
      grep example /tmp/Makefile.tmp >> $TUTORIAL_DIR/log 2>>
$TUTORIAL_DIR/err

        if [[ "$?" == "1" ]]
        then
          echo " failed."
            rm -rf $CBM_BASE_DIR/example
          exit;
        else
          echo " success."
          cp $CBM_BASE_DIR/Makefile.am $CBM_BASE_DIR/Makefile.am.bak
          cp /tmp/Makefile.tmp $CBM_BASE_DIR/Makefile.am
        fi
    fi
else
    echo " failed."
    echo "$CBM_BASE_DIR/Makefile.am not found, is $CBM_BASE_DIR really
the base directory of your PandaRoot installation?"
    exit
fi

echo -n " Updating main configure.ac ..."
# check for existance main configure.ac
if [ -f $CBM_BASE_DIR/configure.ac ]
then
    # check if example is already on the makefiles line
    grep "example/Makefile" $CBM_BASE_DIR/configure.ac >>
$TUTORIAL_DIR/log 2>> $TUTORIAL_DIR/err
    if [[ "$?" == "0" ]]
    then
      echo " warning"
      echo "configure.ac already setup for example"
    else
        # find the sub dir line and add project subdir
      sed "/AC_CONFIG_FILES(\[/a\   example\/Makefile"
$CBM_BASE_DIR/configure.ac > /tmp/configure.tmp

        # check if it worked
```

```
      grep "example/Makefile" /tmp/configure.tmp >> $TUTORIAL_DIR/log
2>> $TUTORIAL_DIR/err

      if [[ "$?" == "1" ]]
      then
          echo " failed."
          rm -rf $CBM_BASE_DIR/example
          mv $CBM_BASE_DIR/configure.ac.bak $CBM_BASE_DIR/configure.ac
          touch $CBM_BASE_DIR/configure.ac
          exit;
        else
          echo " success."
          cp $CBM_BASE_DIR/configure.ac $CBM_BASE_DIR/configure.ac.bak
          cp /tmp/configure.tmp $CBM_BASE_DIR/configure.ac
        fi
    fi
else
    echo " failed."
    echo "$CBM_BASE_DIR/configure.ac not found, is $CBM_BASE_DIR really
the base directory of your PandaRoot installation?"
    exit
fi


echo -n Installing example project files ...
cp $TUTORIAL_DIR/example/* $CBM_BASE_DIR/example >> $TUTORIAL_DIR/log
2>> $TUTORIAL_DIR/err
cp $TUTORIAL_DIR/geometry/* $CBM_BASE_DIR/geometry >> $TUTORIAL_DIR/log
2>> $TUTORIAL_DIR/err
if [ -f $CBM_BASE_DIR/example/PndExample.cxx ]
then
    echo " success."
else
    echo " failed."
    rm -rf $CBM_BASE_DIR/example
    mv $CBM_BASE_DIR/Makefile.am.bak $CBM_BASE_DIR/Makefile.am
    mv $CBM_BASE_DIR/configure.ac.bak $CBM_BASE_DIR/configure.ac
    touch $CBM_BASE_DIR/configure.ac
    exit;
fi

echo -n "Reconfiguring source tree ..."
cd $CBM_BASE_DIR
./reconf >> $TUTORIAL_DIR/log 2>> $TUTORIAL_DIR/err
cd $CBM_BASE_DIR/../build
../pandaroot/configure --enable-geant3 --prefix=$PWD >>
$TUTORIAL_DIR/log 2>> $TUTORIAL_DIR/err
if [ -f $CBM_BASE_DIR/../build/example/Makefile ]
then
    echo " success."
else
    echo " failed."
    rm -rf $CBM_BASE_DIR/example
    mv $CBM_BASE_DIR/Makefile.am.bak $CBM_BASE_DIR/Makefile.am
    mv $CBM_BASE_DIR/configure.ac.bak $CBM_BASE_DIR/configure.ac
    touch $CBM_BASE_DIR/configure.ac
    exit;
```

```
fi

echo -n "Compiling example project ..."
make >> $TUTORIAL_DIR/log 2>> $TUTORIAL_DIR/err
if [[ "$?" == "0" ]]
then
    echo " success."
else
    echo " failed."
    rm -rf $CBM_BASE_DIR/example
    mv $CBM_BASE_DIR/Makefile.am.bak $CBM_BASE_DIR/Makefile.am
    mv $CBM_BASE_DIR/configure.ac.bak $CBM_BASE_DIR/configure.ac
    touch $CBM_BASE_DIR/configure.ac
    exit;
fi

echo -n "Installing example project libraries ..."
make install >> $TUTORIAL_DIR/log 2>> $TUTORIAL_DIR/err
if [[ "$?" == "0" ]]
then
    echo " success."
else
    echo " failed."
    rm -rf $CBM_BASE_DIR/example
    mv $CBM_BASE_DIR/Makefile.am.bak $CBM_BASE_DIR/Makefile.am
    mv $CBM_BASE_DIR/configure.ac.bak $CBM_BASE_DIR/configure.ac
    touch $CBM_BASE_DIR/configure.ac
    exit;
fi

echo -n "Installing example macros ..."
cp $TUTORIAL_DIR/macro/run/* $CBM_BASE_DIR/macro/run >>
$TUTORIAL_DIR/log 2>> $TUTORIAL_DIR/err
if [ -f $CBM_BASE_DIR/macro/run/simExample.C ]
then
    echo " success."
else
    echo " failed."
    rm -rf $CBM_BASE_DIR/example
    mv $CBM_BASE_DIR/Makefile.am.bak $CBM_BASE_DIR/Makefile.am
    mv $CBM_BASE_DIR/configure.ac.bak $CBM_BASE_DIR/configure.ac
    touch $CBM_BASE_DIR/configure.ac
    exit;
fi

echo "Done."

echo "To run the project, change into the $CBM_BASE_DIR/macro/run
directory and start root by typing 'root -l'"
echo "At the root command prompt:"
echo "   - type .X ./simExample.C to run the simulation"
echo "   - type .X ./drawExample.C to view the detector (run the
simulation first)"
echo "   - type .X ./recoExample.C to run the reconstruction code"
```