Pandora Software

The Beginner's Guide

written by
Stefano Vergani

following the slides made by
Dr John Marshall

October 23, 2017

Contents

Chapter 1

First Approach with Pandora

In this chapter it will explained how to run for the first time Pandora from a computer.

1.1 Run Pandora Blind

In this part it is explained how to run for the first time Pandora using LArSoft and enabling the visualisation of the Pandora interface.

1.1.1 UBOONECODE

As very first thing, the terminal must be opened. Digit then the following comand:

```
source /cvmfs/uboone.opensciencegrid.org/products/setup_uboone.sh
setup uboonecode v06_48_00 -q e14:prof
```

The first command tells the computer to go to the CERN Virtual Machine File System (cvmfs) and to look for the setup file. This .sh file contains a certain amount of instruction for your computer and it has to be used everytime a new terminal is opened. The second command tells the computer which version of uboonecode has to be used. v06_48_00 is the current version as this notes are written but it is better always to check which is the latest version. Note that both command must be typed everytime a new terminal is opened. For further details about source files see Section 2.1.1. Then type

```
ls $UBOONECODE_DIR/job
cp $UBOONECODE_DIR/job/reco_uboone_mcc7_driver_stage2.fcl ./myreco_uboone_mcc7_driver_stage2.fcl
```

The first command shows which files are contained in the directory $UBOONECODE_DIR/job. It is worth knowing that the dollar symbol in front of UBOONECODE_DIR is an identifier, that means the name given to a certain path to a certain folder/file. Digiting

```
echo $UBOONECODE_DIR
```

gives us the path to the folder, which in this case is

```
/cvmfs/uboone.opensciencegrid.org/products/uboonecode/v06_48_00
```

The second command copies from the folder in the cvmfs the file .fcl on the local machine, changing the name of the copied file in myreco_uboone_mcc7_driver_stage2.

1.1.2 FCL files

Fermilab Hierarchical Configuration Language (FHiCL or shortered FCL) is a language created at Fermi National Accelerator Laboratory (FNAL or Fermilab). For the pourpose of this guide, only few details will be given. For a proper introduction see [1]. Conceptually, the .fcl gives a list of instructions to the LArSoft software ([2]). Opening the .fcl file the file is this:

```
#include "reco_uboone_mcc7_driver_common.fcl"

process_name: McRecoAprStage2

services.DetectorClocksService.InheritClockConfig:  false

services.TFileService.fileName: "reco_stage_2_hist.root"
physics.reco: [ @sequence::microboone_reco_mcc7_stage2 ]
physics.trigger_paths: [ reco ]
outputs.out1.fileName: "%ifb_%tc_reco2.root"
outputs.out1.dataTier: "reconstructed"
source.inputCommands: ["keep *_*_*_*", "drop *_*_*_McRecoStage2" ]
```

At this point, rename the process name at the second line as follow:

```
process_name: PandoraWorkshop
```

After that, you can try to run LArSoft for the first time. Digit

```
lar -c myreco_uboone_mcc7_driver_stage2.fcl -n 5 /path/to/reco2/file.root
```

lar -c myreco_uboone_mcc7_driver_stage2.fcl launches the LArsoft using instructions given by the .fcl file. lar indicates that you want to run LArSoft, -c that you use the following .fcl file and -n 5 indicates to do certain processes looping on the first 5 events of the .root file and /path/to/reco2/file.root is the .root file you want to open and use. It does not indicate a specific file but rather any kind of .root file related to neutrino interaction in LAr you may have. A typical starting point is

```
/pnfs/uboone/scratch/users/uboonepro/mcc7/v05_08_00/reco2/prodgenie_bnb_nu_uboone
```

or, on Cambridge HEP systems

```
/r05/dune/mcproduction_v05_08_00/larsoft_output_reco2_bnb_nu/
```

Any of the root files contained in those folders is good. Reco2 means the reconstruction has come to a further stage (hard reconstruction) whilst reco1 means a primary stage of reconstruction (low reconstruction). At the end of this process, a new .root file reco_stage_2_hist.root has been created with all the data related to the reconstructed events. To visualize the reconstruced events with pandora software we need another piece, the .xml file.

1.1.3 XML files and Enable Visualisation

Extensible Markup Language (XML) is a language used to give a set of rules in human and machine-readable format. Digit

```
cp $UBOONECODE_DIR/scripts/PandoraSettings_MicroBooNE_Neutrino.xml ./MyPandoraSettings_MicroBooNE_Ne
```

to copy on your local machine the .xml file from the Uboonecode directory. The .xml will be the following:

```

<!-- Pandora settings xml file -->

<pandora>
    <!-- GLOBAL SETTINGS -->
    <IsMonitoringEnabled>false</IsMonitoringEnabled>
    <ShouldDisplayAlgorithmInfo>false</ShouldDisplayAlgorithmInfo>
    <SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>

    <!-- PLUGIN SETTINGS -->
    <MuonPlugin>LArMuonId</MuonPlugin>

    <!-- ALGORITHM SETTINGS -->

    <!-- NEUTRINO-INDUCED EVENT RECONSTRUCTION -->
    <algorithm type = "LArListPreparation">
        <OnlyAvailableCaloHits>true</OnlyAvailableCaloHits>
        <OutputCaloHitListNameW>CaloHitListW</OutputCaloHitListNameW>
        <OutputCaloHitListNameU>CaloHitListU</OutputCaloHitListNameU>
        <OutputCaloHitListNameV>CaloHitListV</OutputCaloHitListNameV>
        <FilteredCaloHitListName>CaloHitList2D</
    FilteredCaloHitListName>
        <CurrentCaloHitListReplacement>CaloHitList2D</
    CurrentCaloHitListReplacement>
        <OutputMCParticleListNameU>MCParticleListU</
    OutputMCParticleListNameU>
        <OutputMCParticleListNameV>MCParticleListV</
    OutputMCParticleListNameV>
        <OutputMCParticleListNameW>MCParticleListW</
    OutputMCParticleListNameW>
        <OutputMCParticleListName3D>MCParticleList3D</
    OutputMCParticleListName3D>
        <CurrentMCParticleListReplacement>MCParticleList3D</
    CurrentMCParticleListReplacement>
        <MipEquivalentCut>0.</MipEquivalentCut>
    </algorithm>

    <algorithm type = "LArVisualMonitoring">
        <CaloHitListNames>CaloHitListW CaloHitListU CaloHitListV</
    CaloHitListNames>
    </algorithm>

```

```
...
```

Listing 1.1: Python example

and then modify the first lines as it follows:

```

<!-- Pandora settings xml file -->

<pandora>
    <!-- GLOBAL SETTINGS -->
    <IsMonitoringEnabled>true</IsMonitoringEnabled>
    <ShouldDisplayAlgorithmInfo>true</ShouldDisplayAlgorithmInfo>
    <SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>

...
```

Listing 1.2: Python example

This will enable the visualisation. Now go back to the .fcl file and modify it adding the .xml file and enabling PandoraNu:

```

#include "reco_uboone_mcc7_driver_common.fcl"

process_name: PandoraWorkshop

#services.RottGraphicsEnablingService: {}

services.DetectorClocksService.InheritClockConfig:  false

services.TFileService.fileName: "reco_stage_2_hist.root"

physics.producers.pandoraWriter: @local::microboone_pandorawriter
physics.producers.pandoraWriter.HitFinderModuleLabel: "gaushit"
physics.producers.pandoraWriter.ConfigFile: "MyPandoraSettings_Write.
    xml"
physics.producers.pandoraNu.HitFinderModuleLabel: "gaushit"
physics.producers.pandoraNu.ConfigFile: "
    MyPandoraSettings_MicroBooNE_Neutrino.xml"

physics.reco: [ pandoraNu ]
physics.trigger_paths: [ reco ]
outputs.out1.fileName: "%ifb_%tc_reco2.root"
outputs.out1.dataTier: "reconstructed"
source.inputCommands: ["keep *_*_*_*", "drop *_*_*_McRecoStage2" ]
```

Listing 1.3: Python example

We use PandoraNu because it is the part of Pandora software used for reconstruction of neutrino events. Finally, with again the command

```
lar -c myreco_uboone_mcc7_driver_stage2.fcl -n 5 /path/to/reco2/file.root
```

the pandora interacting window will appear and man should be able to visualise the reconstructed events.

1.1.4 Pandora Interface

At this stage, giving a comprehensive explanation of the interface is not worth it. Anyways, it is important to see that the user can controll manually which stage of the reconstruction she want to see. Launching the programm the first time gives the first stage of reconstruction (see Fig. 1.1), whilst after pressing for the second time enter a further stage of reconstruction will be added (see Fig. 1.2).

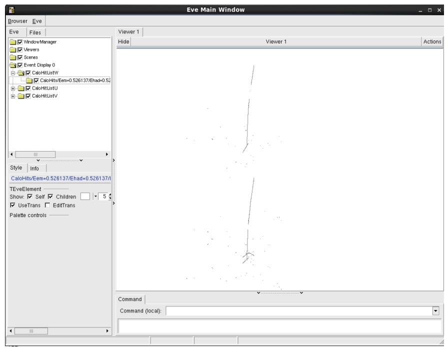

Figure 1.1: Pandora interface at first stage of reconstruction

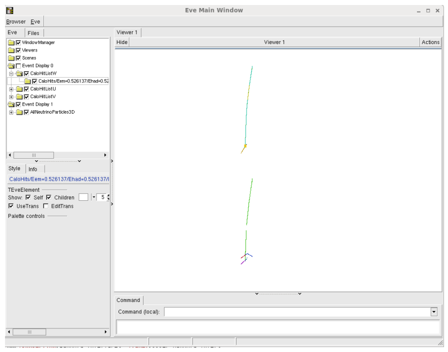

Figure 1.2: Pandora interface at second stage of reconstruction

1.2 Write Events in Pandora Format

Pandora uses special input objects (Hits, MCParticles, Gasps etc.). Those objects are already present in the .root file, but these files tends to come in a big size and for our pourposes we do not need all the information they contain. For this reason, it is possible to serialise input objects in .pndr files (which are small, but the portability is not guaranteed) or .xml files (which are large, but compressible). First thing to do is downloading a new .xml file which will give the command to write a .pndr file.

```
cp $LARPANDORA_DIR/scripts/PandoraSettings_Write.xml ./MyPandoraSettings_Write.xml
```

The file will be the following:

```

<!-- Pandora settings xml file -->

<pandora>
    <!-- GLOBAL SETTINGS -->
    <IsMonitoringEnabled>false</IsMonitoringEnabled>
    <ShouldDisplayAlgorithmInfo>false</ShouldDisplayAlgorithmInfo>
    <SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>
```

```
 9
10     <!-- PLUGIN SETTINGS -->
11     <MuonPlugin>LArMuonId</MuonPlugin>
12
13     <!-- ALGORITHM SETTINGS -->
14     <!--algorithm type = "LArEventReading">
15         <EventFileName>INPUT_XML_OR_PNDR_FILE</EventFileName>
16         <ShouldReadEvents>true</ShouldReadEvents>
17         <SkipToEvent>0</SkipToEvent>
18     </algorithm-->
19
20      <algorithm type = "LArEventWriting">
21         <EventFileName>Pandora_Events.pndr</EventFileName>
22         <ShouldWriteEvents>true</ShouldWriteEvents>
23         <ShouldOverwriteEventFile>true</ShouldOverwriteEventFile>
24         <ShouldWriteMCRelationships>true</ShouldWriteMCRelationships>
25         <ShouldWriteTrackRelationships>true</
      ShouldWriteTrackRelationships>
26     </algorithm>
27 </pandora>
```

Listing 1.4: Python example

edit line 20 as it follows, if for example you want as an output the file "MyPandoraEvents.pndr":

```
<EventFileName>MyPandoraEvents.pndr</EventFileName>
```

Then modify myreco_uboone_mcc7_driver_stage2.fcl as well:

```
physics.reco: [ pandoraNu, pandoraWriter]
```

At this point, launch

```
lar -c myreco_uboone_mcc7_driver_stage2.fcl -n 5 /path/to/reco2/file.root
```

and in your folder the file MyPandoraEvents.pndr will be created.

Chapter 2

A New Algorithm

In this chapter it will explained how to run for the first time Pandora from a computer locally. In the previous chapter, the exercises have been done using the LArSoft stored in the Fermilab servers. At the end of this section, the student will have her/his own version of Pandora stored locally.

2.1 Run Pandora Locally

In this chapter it will explained how to run for the first time Pandora from a computer locally.

2.1.1 Setup

As first thing, digit the following commands:

```
export MY_TEST_AREA=/path/to/your/test/area
export PANDORA_PFA_VERSION=v03-06-00
export ROOT_CMAKE_MODULE_PATH=/path/to/your/FindROOT.cmake/file
```

This is to set the identifiers MY_TEST_AREA, ROOT_CMAKE_MODULE_PATH and PANDORA_PFA_VERSION (to see what is an identifier go to Section 1.1.1). With Pandora PFA we mean Pandora Particle Flow Algorithm,which is the GitHub name for Pandora [3]. It is worth noting that whilst the first area could be any possible folder you like, the Pandora version and the path to the Root cmake must be choosen wisely. For the version of a software it is always a good idea to try the latest version (for Pandora can be found here [3]). If it gives problems, simply try the recommended version or older ones. The path to the Root cmake interely dipends on where you saved Root. If you want to use Root stored in Fermilab

```
export ROOT_CMAKE_MODULE_PATH=$ROOTSYS/etc/cmake/FindROOT.cmake
```

If you now try an echo command, it should give you back the path you indicated. One should type these three commands everytime a new terminal is opened. To avoid that, there are two possible options. The first one is to create a script, called for example setup.sh. This script will contain those three commands and typing

```
source setup.sh
```

the commands will be launched. There is also the possibility to launch automatically these commands. Go to your home directory and modify the file .bashrc. This file contain all the commands the terminal automatically run when it is opened. One can simply add the three exports, but this is not always the best thing to do because not always we want all these commands to be run automatically. So another option is to type:

```
alias start='source path/to/your/setup.sh'
```

With this command, everytime one will digit "start" the command source setup.sh will be launched.

2.1.2 Installation

Digit

```
cd $MY_TEST_AREA
git clone
git clone https://github.com/PandoraPFA/PandoraPFA
cd PandoraPFA
git checkout $PANDORA_PFA_VERSION
```

After this, you will have cloned the github repository on your machine. at this point create a folder to build Pandora

```
mkdir build
cd build
```

and create the cmake file

```
cmake -DCMAKE_MODULE_PATH=$ROOT_CMAKE_MODULE_PATH \
-DPANDORA_MONITORING=ON -DPANDORA_LAR_CONTENT=ON  \
-DCMAKE_CXX_FLAGS=-std=c++14 ..
```

the last command, -DCMAK_CXX_FLAGS=-std=c++14 orders to use the latest compiler avaible or at least version 14. The double dots at the end tells the compiler to search the file outside the directory "build".

Now it is possible to install Pandora locally

```
make -j4 install
```

The installation process should take a couple of minutes. After that, we will setup a library and applicatio explicity tailored for the next exercises.

```
cd $MY_TEST_AREA
git clone https://github.com/PandoraPFA/WorkshopContent
cd WorkshopContent
mkdir build
cd build

cmake -DCMAKE_MODULE_PATH="$ROOT_CMAKE_MODULE_PATH;$MY_TEST_AREA/PandoraPFA/cmakemodules" \
-DPandoraSDK_DIR=$MY_TEST_AREA/PandoraPFA -DPANDORA_MONITORING=ON                          \
-DPandoraMonitoring_DIR=$MY_TEST_AREA/PandoraPFA                                           \
-DLArContent_DIR=$MY_TEST_AREA/PandoraPFA -DCMAKE_CXX_FLAGS=-std=c++14 ..

make -j4 install
```

2.2 Adding a new Algorithm

In this part we will create a new algorithm. To do so we have to go in the subfolder Algorithms. There is already a set of templates which can be used. Digit

```
cd $MY_TEST_AREA/WorkshopContent/workshopcontent/Algorithms
python CreateNewAlgorithm.py --name MyTest
```

The last command will create a new algorithm called MyTestAlgorithm (both .cc and .h files) using the existing templates. Now digit

```
cd ..
cd Test
```

and modify the file PandoraWorkshop.cc adding the parts in red:

```

...

#include "Api/PandoraApi.h"

#include "larpandoracontent/LArContent.h"
#include "larpandoracontent/LArPlugins/LArPseudoLayerPlugin.h"
#include "larpandoracontent/LArPlugins/
    LArRotationalTransformationPlugin.h"

\textcolor{red}{#include "workshopcontent/Algorithms/MyTestAlgorithm.
    h"}

#ifdef MONITORING
#include "TApplication.h"
#endif

...

int main(int argc, char *argv[])
{
    try
    {
        Parameters parameters;

        if (!ParseCommandLine(argc, argv, parameters))
```

```
            return 1;

#ifdef MONITORING
        TApplication *const pTApplication = new TApplication("Workshop", &argc, argv);
        pTApplication->SetReturnFromRun(kTRUE);
#endif
        const pandora::Pandora *const pPandora = new pandora::Pandora();

        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, LArContent::RegisterAlgorithms(*pPandora));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, LArContent::RegisterBasicPlugins(*pPandora));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, PandoraApi::SetPseudoLayerPlugin(*pPandora, new lar_content::LArPseudoLayerPlugin));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, PandoraApi::SetLArTransformationPlugin(*pPandora, new lar_content::LArRotationalTransformationPlugin));

\textcolor{red}{PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, PandoraApi::RegisterAlgorithmFactory(*pPandora, "MyTestAlgorithm", new workshop_content::MyTestAlgorithm::Factory));}

        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, PandoraApi::ReadSettings(*pPandora, parameters.m_pandoraSettingsFile));

        int nEvents(0);
        while ((nEvents++ < parameters.m_nEventsToProcess) || (0 > parameters.m_nEventsToProcess))
        {
            if (parameters.m_shouldDisplayEventNumber)
                std::cout << std::endl << "   PROCESSING EVENT: " << (nEvents - 1) << std::endl << std::endl;

            PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, PandoraApi::ProcessEvent(*pPandora));
            PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, PandoraApi::Reset(*pPandora));
        }

        delete pPandora;
    }
    catch (pandora::StatusCodeException &statusCodeException)
    {
        std::cerr << "Pandora Exception caught: " << statusCodeException.ToString() << std::endl;
        return 1;
    }

    return 0;
}
```

Listing 2.1: Python example

This will ???????????????????????. At this point, go to the build folder, re-run the CMake and do make install.

```
cd $MY_TEST_AREA/WorkshopContent/build

cmake -DCMAKE_MODULE_PATH="$ROOT_CMAKE_MODULE_PATH;$MY_TEST_AREA/PandoraPFA/cmakemodules" \
-DPandoraSDK_DIR=$MY_TEST_AREA/PandoraPFA -DPANDORA_MONITORING=ON                         \
-DPandoraMonitoring_DIR=$MY_TEST_AREA/PandoraPFA                                          \
-DLArContent_DIR=$MY_TEST_AREA/PandoraPFA -DCMAKE_CXX_FLAGS=-std=c++14 ..

make install
```

At this point, the last thing to do is to run the new algorithm.

```
cd $MY_TEST_AREA/WorkshopContent/settings
```

and modify the file PandoraSettings_Workshop.xml at line 30 adding

```
  <algorithm type = "MyTestAlgorithm"/>
```

because we have to declare the new algorithm.
Now digit

```
$MY_TEST_AREA/WorkshopContent/bin/PandoraWorkshop -?
```

This command tells you which arguments you have to give to run the programm. In this case, the outcome will be

```
PandoraWorkshop
    -i PandoraSettings.xml (required)
    -n NEventsToProcess    (optional)
    -N                     (optional, display event numbers)
```

So, launch the programm with

```
$MY_TEST_AREA/WorkshopContent/bin/PandoraWorkshop                      \
-i $MY_TEST_AREA\WorkshopContent/settings/PandoraSettings_Workshop.xml \
-n 10
```

This programm will give you this error

```
Failure in reading pandora settings, STATUS_CODE_FAILURE
PandoraApi::ReadSettings(*pPandora, parameters.m_pandoraSettingsFile) throw STATUS_CODE_FAILURE
    in function: main
    in file:     /usera/sv408/WorkshopContent/workshopcontent/Test/PandoraWorkshop.cc line#: 88
Pandora Exception caught: STATUS_CODE_FAILURE
```

2.2.1 Geometry Files

The previous error is due to the fact we did not specify a Geometry file. In fact, if we have a look to the PandoraSettings_Workshop.xml

```

    <!-- ALGORITHM SETTINGS -->
    <algorithm type = "LArEventReading">
        <EventFileNameList>/path/to/geometry/.pndr/or/.xml</
    EventFileNameList>
        <GeometryFileName>/path/to/geometry/.pndr/or/.xml</
    GeometryFileName>
        <SkipToEvent>0</SkipToEvent>
    </algorithm>
```

Listing 2.2: Python example

we see we have to specify a geometry file. Pandora is quite a flexible software that can be adapted for a variety of different detectors. Every different detector needs a specific geometry file, and in this case we want the geometry file related to MicroBooNE. So change the PandoraSettings_Workshop.xml in this way

```

    <!-- ALGORITHM SETTINGS -->
    <algorithm type = "LArEventReading">
        <EventFileNameList>/r05/uboone/jjd49/cincinatti_sample/
    Pandora_Events_Cincinatti_BNB_NuMu_1714.pndr</EventFileNameList>
        <GeometryFileName>/usera/sv408/WorkshopContent/settings/
    uboone/Geometry_MicroBooNE.xml</GeometryFileName>
        <SkipToEvent>0</SkipToEvent>
    </algorithm>
```

Listing 2.3: Python example

and this time the program will work.

Chapter 3

Cluster Creation

Blablablabla

3.1 Algorithm Configuration

First of all, in the following section we will modify many times many files. As a rule, every time we modify a .cc or .h file, before launching the modified progroamm we will run

```
cd $MY_TEST_AREA/WorkshopContent/build/
make install
```

If instead we modify a .xml, there is no need to run make install. In the following we will use many times files from the XmlHelper.h. For the pourpose of this guide, you do not need neither to modify nor to know this files, but in case you were curious you can find it in the Pandora Software Development Kit (PandoraSDK) located in:

```
$MY_TEST_AREA/PandoraPFA/PandoraSDK-v03-01-00/include/Helpers
```

"The Pandora SDK aims to provide a robust, reliable and easy-to-use environment for developing and running pattern recognition algorithms" [5]. The entire PandoraSDK can be found here [3].

At this point go in the folder

```
$MY_TEST_AREA/WorkshopContent/workshopcontent/Algorithms
```

and open the files MyTestAlgorithm.h and MyTestAlgorithm.cc. Modify the .h file as it follows:

```

/**
 *  @file   WorkshopContent/workshopcontent/Algorithms/
    MyTestAlgorithm.h
 *
 *  @brief  Header file for the mytest algorithm class.
 *
 *  $Log: $
 */
#ifndef WORKSHOP_MYTEST_ALGORITHM_H
#define WORKSHOP_MYTEST_ALGORITHM_H 1

#include "Pandora/Algorithm.h"

namespace workshop_content
{

/**
 *  @brief  MyTestAlgorithm class
 */
class MyTestAlgorithm : public pandora::Algorithm
{
public:


    /**
     *  @brief  Factory class for instantiating algorithm
     */

 class Factory : public pandora::AlgorithmFactory
    {
    public:
        pandora::Algorithm *CreateAlgorithm() const;
    };

  /**
  *  @brief Defaul constructor
  */
  MyTestAlgorithm();
private:
    pandora::StatusCode Run();
    pandora::StatusCode ReadSettings(const pandora::TiXmlHandle
    xmlHandle);

    // Member variables here

    std::string      m_myMandatoryString;     ///< A mandatory string
    bool       m_myOptionalBool;    ///< An optional Boolean
    unsigned int     m_myOptionalUnsignedInt;  ///< An optional
    unsigned int
    pandora::FloatVector   m_myMandatoryFloatVector; ///< A mandatory
    vector of floats


};


inline pandora::Algorithm *MyTestAlgorithm::Factory::CreateAlgorithm
    () const
{
    return new MyTestAlgorithm();
}


} // namespace workshop_content

#endif // #ifndef WORKSHOP_MYTEST_ALGORITHM_H
```

Listing 3.1: Python example

Then modify the .cc file from this

```
/**
 *  @file   WorkshopContent/workshopcontent/Algorithms/
    MyTestAlgorithm.cc
 *
 *  @brief  Implementation of the mytest algorithm class.
 *
 *  $Log: $
 */

#include "Pandora/AlgorithmHeaders.h"

#include "workshopcontent/Algorithms/MyTestAlgorithm.h"

using namespace pandora;

namespace workshop_content
{

StatusCode MyTestAlgorithm::Run()
{
    // Algorithm code here

    return STATUS_CODE_SUCCESS;
}

//------------------------------------------------------------------------------------------------------

StatusCode MyTestAlgorithm::ReadSettings(const TiXmlHandle /*
    xmlHandle*/)
{
    // Read settings from xml file here

    return STATUS_CODE_SUCCESS;
}

} // namespace workshop_content
```

Listing 3.2: Python example

To this

```

/**
 *  @file   WorkshopContent/workshopcontent/Algorithms/
    MyTestAlgorithm.cc
 *
 *  @brief  Implementation of the mytest algorithm class.
 *
 *  $Log: $
 */

#include "Pandora/AlgorithmHeaders.h"

#include "workshopcontent/Algorithms/MyTestAlgorithm.h"

using namespace pandora;



namespace workshop_content
{

MyTestAlgorithm::MyTestAlgorithm() :
    m_myMandatoryString(),
    m_myOptionalBool(false),
    m_myOptionalUnsignedInt(5),
    m_myMandatoryFloatVector()
{
}
//----------------------------------------------------------------------------
StatusCode MyTestAlgorithm::Run()
{
    std::cout << "-m_myMandatoryString: " << m_myMandatoryString << std::endl
    <<      "-m_myOptionalBool: "   << m_myOptionalBool       << std::endl
    << "-m_myOptionalUnsignedInt: " << m_myOptionalUnsignedInt << std::endl
    << "-m_myMandatoryString: ";

    for (const auto value: m_myMandatoryFloatVector)
        std::cout << value << " ";

    std::cout << std::endl;
    return STATUS_CODE_SUCCESS;
}

//------------------------------------------------------------------

StatusCode MyTestAlgorithm::ReadSettings(const TiXmlHandle xmlHandle)
{
  PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, XmlHelper::ReadValue(xmlHandle,"MyMandatoryString", m_myMandatoryString));
  PANDORA_RETURN_RESULT_IF_AND_IF(STATUS_CODE_SUCCESS, STATUS_CODE_NOT_FOUND, !=, XmlHelper::ReadValue(xmlHandle,"MyOptionalBool", m_myOptionalBool));
  PANDORA_RETURN_RESULT_IF_AND_IF(STATUS_CODE_SUCCESS, STATUS_CODE_NOT_FOUND, !=, XmlHelper::ReadValue(xmlHandle,"MyOptionalUnsignedInt", m_myOptionalUnsignedInt));
  PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, XmlHelper::ReadVectorOfValues(xmlHandle,"MyMandatoryFloatVector", m_myMandatoryFloatVector));

        return STATUS_CODE_SUCCESS;
}

} // namespace workshop_content
```

Listing 3.3: Python example

These are actually basic modifications, but instructive to understand how the .cc file works. We will analyse now what these modifications mean and do.

```

MyTestAlgorithm::MyTestAlgorithm() :
    m_myMandatoryString(),
    m_myOptionalBool(false),
    m_myOptionalUnsignedInt(5),
    m_myMandatoryFloatVector()
{
}
```

Listing 3.4: Python example

simply assignes default values upon construction.

```
StatusCode MyTestAlgorithm::Run()
{
    std::cout << "-m_myMandatoryString: " << m_myMandatoryString << std::endl
    <<      "-m_myOptionalBool: "   << m_myOptionalBool       << std::endl
    << "-m_myOptionalUnsignedInt: " << m_myOptionalUnsignedInt << std::endl
    << "-m_myMandatoryString: ";

    for (const auto value: m_myMandatoryFloatVector)
        std::cout << value << " ";

    std::cout << std::endl;
    return STATUS_CODE_SUCCESS;
}
```

Listing 3.5: Python example

prints out the values at run time and

```
StatusCode MyTestAlgorithm::ReadSettings(const TiXmlHandle xmlHandle)
{
  PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, XmlHelper::
     ReadValue(xmlHandle,"MyMandatoryString", m_myMandatoryString));
  PANDORA_RETURN_RESULT_IF_AND_IF(STATUS_CODE_SUCCESS,
     STATUS_CODE_NOT_FOUND, !=, XmlHelper::ReadValue(xmlHandle,"
     MyOptionalBool", m_myOptionalBool));
  PANDORA_RETURN_RESULT_IF_AND_IF(STATUS_CODE_SUCCESS,
     STATUS_CODE_NOT_FOUND, !=, XmlHelper::ReadValue(xmlHandle,"
     MyOptionalUnsignedInt", m_myOptionalUnsignedInt));
  PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, XmlHelper::
     ReadVectorOfValues(xmlHandle,"MyMandatoryFloatVector",
     m_myMandatoryFloatVector));

        return STATUS_CODE_SUCCESS;
}
```

Listing 3.6: Python example

adds the optional and mandatory reads.

Now, try to run the program. As explained above, before running the program compile everything (in this guide, we will use the verb to compile and to build as synonyms):

```
cd $MY_TEST_AREA/WorkshopContent/build
make install
```

At this point, you can try to run the program with:

```
$MY_TEST_AREA/WorkshopContent/bin/PandoraWorkshop                      \
-i $MY_TEST_AREA\WorkshopContent/settings/PandoraSettings_Workshop.xml \
-n 10
```

and you will get this error:

```
XmlHelper::ReadValue(xmlHandle,"MyMandatoryString",
     m_myMandatoryString) return STATUS_CODE_NOT_FOUND
     in function: ReadSettings
     in file:     /usera/sv408/WorkshopContent/workshopcontent/
     Algorithms/MyTestAlgorithm.cc line#: 46
pLocalAlgorithm->ReadSettings(TiXmlHandle(pXmlElement)) throw
     STATUS_CODE_NOT_FOUND
     in function: CreateAlgorithm
     in file:     /usera/sv408/PandoraPFA/PandoraSDK-v03-01-00/src/
     Managers/AlgorithmManager.cc line#: 135
Failure in reading pandora settings, STATUS_CODE_NOT_FOUND
PandoraApi::ReadSettings(*pPandora, parameters.m_pandoraSettingsFile)
      throw STATUS_CODE_FAILURE
     in function: main
     in file:     /usera/sv408/WorkshopContent/workshopcontent/Test/
     PandoraWorkshop.cc line#: 88
Pandora Exception caught: STATUS_CODE_FAILURE
```

Listing 3.7: Python example

This is because we modified in the .cc file parts which needed the .xml without then modifiyng the .xml file. Therefore, open the .xml file

`$MY_TEST_AREA/WorkshopContent/settings /PandoraSettings_Workshop.xml`

and where where you added (see Section 2.2):

```
 <algorithm type = "MyTestAlgorithm"/>
```

change and add the following lines:

```
  <algorithm type = "MyTestAlgorithm">
        <MyMandatoryString>TestString</MyMandatoryString>
  <MyOptionalUnsignedInt>10</MyOptionalUnsignedInt>
        <MyMandatoryFloatVector>0. 1.5 3.0 4.5</
    MyMandatoryFloatVector>
   </algorithm>
```

Listing 3.8: XML example

At this point, run again the program with

```
$MY_TEST_AREA/WorkshopContent/bin/PandoraWorkshop                        \
-i $MY_TEST_AREA\WorkshopContent/settings/PandoraSettings_Workshop.xml \
-n 10
```

and you will see that in the output will appear also

```
> Running Algorithm: Alg0001, LArEventReading
> Running Algorithm: Alg0002, LArListPreparation
> Running Algorithm: Alg0003, MyTestAlgorithm
-m_myMandatoryString: TestString
-m_myOptionalBool: 0
-m_myOptionalUnsignedInt: 10
-m_myMandatoryString: 0 1.5 3 4.5
...
```

Listing 3.9: Python example

Therefore, even if very small, we managed to write and make it work a first simple Pandora algorithm.

3.1.1 Application Programming Interfaces

"An API (application programming interface) is a term meaning the functions/methods in a library that you can call to ask it to do things for you - the interface to the library." [4]. An API is basically something we do not need to modify that is an intermediate between us and libraries. We can use an API to call functions in the library without actually knowing exactly how this libriray works or it is made. If interested in seeing of an API is made, have a look to:

`$MY_TEST_AREA/workshop/PandoraPFA/PandoraSDK-v03-01-00/include/Api/PandoraContentApi.h`

3.2 Adding a Sorting Algorithm

Now we want to change MyTestAlgorithm.cc and to start doing something a bit more elaborated. We want to take the fist n hits and sort them according to the z distance, which in 3D is given by the following formula:

$$D = \sqrt{x^2 + y^2 + z^2} \tag{3.1}$$

Modify the code MyTestAlgorithm.cc as it follows:

```
/**
 *  @file   WorkshopContent/workshopcontent/Algorithms/MyTestAlgorithm.cc
 *
 *  @brief  Implementation of the mytest algorithm class.
 *
 *  $Log: $
 */

#include "Pandora/AlgorithmHeaders.h"

#include "larpandoracontent/LArHelpers/LArClusterHelper.h"

#include "workshopcontent/Algorithms/MyTestAlgorithm.h"

using namespace pandora;
using namespace lar_content;

namespace workshop_content
{
StatusCode MyTestAlgorithm::Run()
{

  const CaloHitList *pCaloHitList(nullptr);
  PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi
     ::GetCurrentList(*this, pCaloHitList));

  CaloHitVector sortedCaloHits(pCaloHitList->begin(), pCaloHitList->
     end());
  std::sort(sortedCaloHits.begin(), sortedCaloHits.end(),
     LArClusterHelper::SortHitsByPosition);

  for(const CaloHit *const pCaloHit : sortedCaloHits)
  {
    std::cout << "InputHit - HitType: " << pCaloHit->GetHitType() <<
     ", " << pCaloHit->GetPositionVector() << std::endl;
  }

return STATUS_CODE_SUCCESS;



}



StatusCode MyTestAlgorithm::ReadSettings(const TiXmlHandle) //
     xmlHandle)
{


        return STATUS_CODE_SUCCESS;
}
}
```

Listing 3.10: Version of MyTestAlgorithms.cc to sort hits regarding z-coordinate

and MyTestAlgorithm.h as well:

```
/**
 *  @file   WorkshopContent/workshopcontent/Algorithms/
     MyTestAlgorithm.h
 *
 *  @brief  Header file for the mytest algorithm class.
 *
 *  $Log: $
 */
#ifndef WORKSHOP_MYTEST_ALGORITHM_H
```

```
#define WORKSHOP_MYTEST_ALGORITHM_H 1

#include "Pandora/Algorithm.h"

namespace workshop_content
{

/**
 *  @brief  MyTestAlgorithm class
 */
class MyTestAlgorithm : public pandora::Algorithm
{
public:
    /**
     *  @brief  Factory class for instantiating algorithm
     */

 class Factory : public pandora::AlgorithmFactory
    {
    public:
        pandora::Algorithm *CreateAlgorithm() const;
    };
private:
    pandora::StatusCode Run();
    pandora::StatusCode ReadSettings(const pandora::TiXmlHandle
    xmlHandle);

    // Member variables here
};

inline pandora::Algorithm *MyTestAlgorithm::Factory::CreateAlgorithm
    () const
{
    return new MyTestAlgorithm();
}

} // namespace workshop_content

#endif // #ifndef WORKSHOP_MYTEST_ALGORITHM_H
```

Listing 3.11: Version of MyTestAlgorithms.h to sort hits regarding z-coordinate

Now, after building it as in code 3.1, run it using 3.1 and this is the outcome:

```
> Running Algorithm: Alg0001, LArEventReading
> Running Algorithm: Alg0002, LArListPreparation
> Running Algorithm: Alg0003, MyTestAlgorithm
InputHit - HitType: 6,   x: 220.64  y: 0  z: 300.85 length: 373.086
InputHit - HitType: 6,   x: 220.659  y: 0  z: 301.15 length: 373.339
InputHit - HitType: 6,   x: 220.645  y: 0  z: 301.45 length: 373.572
InputHit - HitType: 6,   x: 220.642  y: 0  z: 301.75 length: 373.813
InputHit - HitType: 6,   x: 220.638  y: 0  z: 302.05 length: 374.052
InputHit - HitType: 6,   x: 220.645  y: 0  z: 302.35 length: 374.299
InputHit - HitType: 6,   x: 220.634  y: 0  z: 302.65 length: 374.535
InputHit - HitType: 6,   x: 220.581  y: 0  z: 302.95 length: 374.746
InputHit - HitType: 6,   x: 220.557  y: 0  z: 303.25 length: 374.974
InputHit - HitType: 6,   x: 220.556  y: 0  z: 303.55 length: 375.217
InputHit - HitType: 6,   x: 220.527  y: 0  z: 303.85 length: 375.442
InputHit - HitType: 6,   x: 220.189  y: 0  z: 304.15 length: 375.487
InputHit - HitType: 6,   x: 220.76  y: 0  z: 304.15 length: 375.822
InputHit - HitType: 6,   x: 220.13  y: 0  z: 304.45 length: 375.695
InputHit - HitType: 6,   x: 220.754  y: 0  z: 304.45 length: 376.061
...
```

Listing 3.12: Python example

Having a look to the outcome, we see InputHit - HitType. If we want to know what it means, we can do an useful exercise. Go to the PandoraSDK [3], click on include and then objects. We are working with CaloHit, therefore click on CaloHit and search for HitType. You will find:

```
const HitType m_hitType; ///< The type of calorimeter hit
```

Now if you want to know what to what the number 6 is related, go to include, Pandora and PandoraEnumeratedTypes.h. Search again for HitType and you will find:

```

/**
 *  @brief  Calorimeter hit type enum
 */
enum HitType
{
    TRACKER,
    ECAL,
    HCAL,
    MUON,
    TPC_VIEW_U,
    TPC_VIEW_V,
    TPC_VIEW_W,
    TPC_3D,
    HIT_CUSTOM
};
```

Keep in mind it starts counting from 0.

3.3 Adding MCParticle List

Monte Carlo Particle list (MCParticle list) provides details of true pattern-recognition solition. This means that we start with simulated Monte Carlo (MC) events, than we reconstruct them and eventually we can check with MCParticle how good we have reconstruced those events. We will not us MCParticle for real data but it is really useful during the development process. We can now modify the MyTestAlgorithm.cc to add MCParticle as an output:

```
/**
 *  @file   WorkshopContent/workshopcontent/Algorithms/
    MyTestAlgorithm.cc
 *  @brief  Implementation of the mytest algorithm class.
 */

#include "Pandora/AlgorithmHeaders.h"
#include "larpandoracontent/LArHelpers/LArClusterHelper.h"
#include "larpandoracontent/LArHelpers/LArMCParticleHelper.h"
#include "workshopcontent/Algorithms/MyTestAlgorithm.h"

using namespace pandora;
using namespace lar_content;

namespace workshop_content
{
StatusCode MyTestAlgorithm::Run()
{
  //CaloHits
  const CaloHitList *pCaloHitList(nullptr);
  PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi
    ::GetCurrentList(*this, pCaloHitList));

  CaloHitVector sortedCaloHits(pCaloHitList->begin(), pCaloHitList->
    end());
  std::sort(sortedCaloHits.begin(), sortedCaloHits.end(),
    LArClusterHelper::SortHitsByPosition);

  for(const CaloHit *const pCaloHit : sortedCaloHits)
  {
    std::cout << "InputHit - HitType: " << pCaloHit->GetHitType() <<
    ", " << pCaloHit->GetPositionVector() << std::endl;
  }
  //MCParticle
  const MCParticleList *pMCParticleList(nullptr);
  PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi
    ::GetCurrentList(*this, pMCParticleList));

  MCParticleVector sortedMCParticles(pMCParticleList->begin(),
    pMCParticleList->end());
  std::sort(sortedMCParticles.begin(), sortedMCParticles.end(),
    LArMCParticleHelper::SortByMomentum);

  for (const MCParticle *const pMCParticle : sortedMCParticles)
  {
    std::cout << "InputMCParticle - PDG: " << pMCParticle->
    GetParticleId() << ", nParents " << pMCParticle->GetParentList()
    .size()
        << ", nDaughters " << pMCParticle->GetDaughterList().size()
    << std::endl;
  }
return STATUS_CODE_SUCCESS;

}

StatusCode MyTestAlgorithm::ReadSettings(const TiXmlHandle) //
    xmlHandle)
{


        return STATUS_CODE_SUCCESS;
}
}
```

Listing 3.13: MyTestAlgorithm.cc including for the first time MCParticles

and the output will be:

```
...
InputMCParticle - PDG: 2212, nParents 0, nDaughters 0
InputMCParticle - PDG: 13, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 11, nParents 0, nDaughters 0
InputMCParticle - PDG: 22, nParents 0, nDaughters 0
InputMCParticle - PDG: 22, nParents 0, nDaughters 0
...
```

Listing 3.14: Output of the programm

Particle Data Group (PDG) is a number used to identify particles according to this table:

```
// Specify (name, pdg code, mass in GeV, width in GeV, charge)
#define PARTICLE_DATA_TABLE(d)                                          \
    d(PHOTON,              22,            0.E+00f,            0.E+00f,        0)      \
    d(E_MINUS,             11,     5.10998902E-04f,            0.E+00f,       -1)      \
    d(E_PLUS,             -11,     5.10998902E-04f,            0.E+00f,       +1)      \
    d(MU_MINUS,            13,     1.05658357E-01f,        2.99591E-19f,      -1)      \
    d(MU_PLUS,            -13,     1.05658357E-01f,        2.99591E-19f,      +1)      \
    d(TAU_MINUS,           15,        1.77699E+00f,          2.265E-12f,      -1)      \
    d(TAU_PLUS,           -15,        1.77699E+00f,          2.265E-12f,      +1)      \
    d(NU_E,                12,            0.E+00f,            0.E+00f,        0)      \
    d(NU_E_BAR,           -12,            0.E+00f,            0.E+00f,        0)      \
    d(NU_MU,               14,            0.E+00f,            0.E+00f,        0)      \
    d(NU_MU_BAR,          -14,            0.E+00f,            0.E+00f,        0)      \
    d(NU_TAU,              16,            0.E+00f,            0.E+00f,        0)      \
    d(NU_TAU_BAR,         -16,            0.E+00f,            0.E+00f,        0)      \
    d(PI_PLUS,            211,      1.3957018E-01f,         2.5284E-17f,      +1)      \
    d(PI_MINUS,          -211,      1.3957018E-01f,         2.5284E-17f,      -1)      \
    d(PI_ZERO,            111,       1.349766E-01f,            7.8E-09f,       0)      \
    d(LAMBDA,            3122,       1.115683E+00f,          2.501E-15f,       0)      \
    d(LAMBDA_BAR,       -3122,       1.115683E+00f,          2.501E-15f,       0)      \
    d(K_PLUS,             321,        4.93677E-01f,          5.315E-17f,      +1)      \
    d(K_MINUS,           -321,        4.93677E-01f,          5.315E-17f,      -1)      \
    d(K_SHORT,            310,        4.97672E-01f,          7.367E-15f,       0)      \
    d(K_LONG,             130,        4.97672E-01f,          1.272E-17f,       0)      \
    d(SIGMA_MINUS,       3112,         1.1975E+00f,           8.28E-15f,      -1)      \
    d(SIGMA_PLUS,        3222,         1.1975E+00f,           8.28E-15f,      +1)      \
    d(SIGMA_MINUS_BAR,  -3112,         1.1975E+00f,           8.28E-15f,      +1)      \
    d(SIGMA_PLUS_BAR,   -3222,         1.1975E+00f,           8.28E-15f,      -1)      \
    d(HYPERON_ZERO ,     3322,        1.31483E+00f,           2.28E-15f,       0)      \
    d(HYPERON_ZERO_BAR, -3322,        1.31483E+00f,           2.28E-15f,       0)      \
    d(HYPERON_MINUS,     3312,        1.32131E+00f,           4.04E-15f,      -1)      \
    d(HYPERON_MINUS_BAR, -3312,       1.32131E+00f,           4.04E-15f,      +1)      \
    d(PROTON,            2212,      9.3827200E-01f,            0.E+00f,       +1)      \
    d(PROTON_BAR,       -2212,      9.3827200E-01f,            0.E+00f,       -1)      \
    d(NEUTRON,           2112,      9.3956533E-01f,          7.432E-28f,       0)      \
d(NEUTRON_BAR, -2112, 9.3956533E-01f, 7.432E-28f, 0)
```

According to this table, in that particular event there were generated one protons, one muon, several electrons and photons.

3.3.1 Enable Visualisation

At this point, we want to modify Listing 3.13 in order to enable the visualisation. Modify as so MyTestAlgorithm.cc:

```cpp
include "Pandora/AlgorithmHeaders.h"
#include "larpandoracontent/LArHelpers/LArClusterHelper.h"
#include "larpandoracontent/LArHelpers/LArMCParticleHelper.h"
#include "workshopcontent/Algorithms/MyTestAlgorithm.h"

using namespace pandora;
using namespace lar_content;

namespace workshop_content
{
StatusCode MyTestAlgorithm::Run()
{
  //CaloHits
  const CaloHitList *pCaloHitList(nullptr);
  PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi
     ::GetCurrentList(*this, pCaloHitList));

  const bool showDetectorGaps(true);
  PandoraMonitoringApi::SetEveDisplayParameters(this->GetPandora(),
     showDetectorGaps, DETECTOR_VIEW_XZ, -1.f, -1.f, 1.f);
  PandoraMonitoringApi::VisualizeCaloHits(this->GetPandora(),
     pCaloHitList, "CurrentCaloHits", BLUE);

  //MCParticle
  const MCParticleList *pMCParticleList(nullptr);
  PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi
     ::GetCurrentList(*this, pMCParticleList));
  PandoraMonitoringApi::VisualizeMCParticles(this->GetPandora(),
     pMCParticleList, "CurrentMCParticles", RED);

  PandoraMonitoringApi::ViewEvent(this->GetPandora());
return STATUS_CODE_SUCCESS;

}

StatusCode MyTestAlgorithm::ReadSettings(const TiXmlHandle) //
     xmlHandle)
{


        return STATUS_CODE_SUCCESS;
}
}
```

Listing 3.15: MyTestAlgorithm.cc now enables visualisation of both CaloHits (blue) and MCParticles (red)

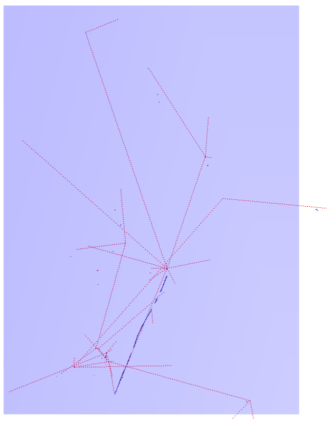

Figure 3.1: Pandora interface with CaloHits (blue) and MCParticles (red)

From Figure 3.1 we can see the CaloHits and the MCParticles. For more examples, see

`$MY_TEST_AREA/WorkshopContent/examplecontent/ExampleAlgorithms/DisplayListsAlgorithm.cc or .h`

Bibliography

[1] FHiCL 3 Quick Start Guide https://cdcvs.fnal.gov/redmine/projects/uboonecode/wiki/Guide_to_Using_FCL_fi

[2] Introduction to LArSoft Software https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/Introduction_to_LArSoft

[3] GitHub page containing Pandora documentation https://github.com/PandoraPFA

[4] Difference between framework vs Library vs IDE vs API vs SDK vs Toolkits? https://stackoverflow.com/questions/8772746/difference-between-framework-vs-library-vs-ide-vs-api-vs-sdk-vs-toolkits

[5] J.S. Marshall, M.A. Thomson *The Pandora Software Development Kit for Pattern Recognition* arXiv:1506.05348