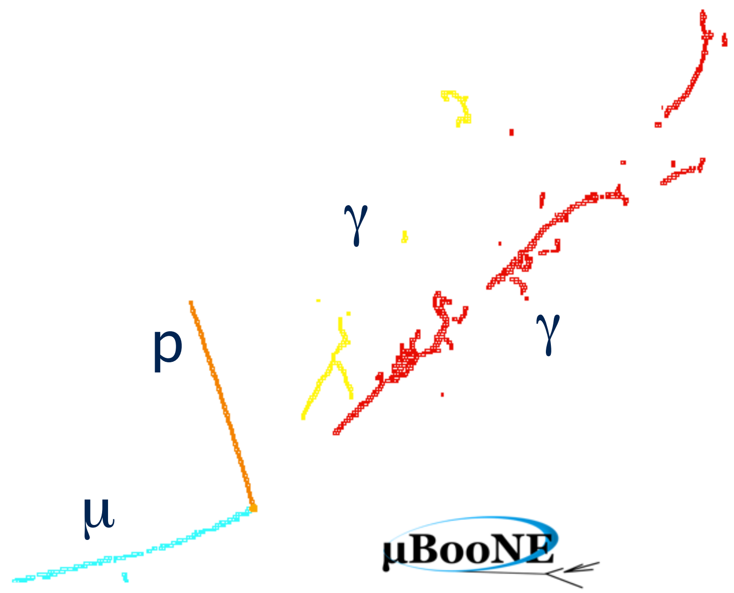


Pandora Exercise 3: Cluster Creation

J. S. Marshall for the Pandora Team

MicroBooNE Pandora Workshop

July 11-14th 2016, Cambridge

Algorithm Implementation

Pre-requisite: Exercise 2 - setup Pandora environment and add a new algorithm.

Start to add algorithm implementation:

- Add optional and mandatory configurable parameters
- Access lists of objects: Hits and MCParticles
- Draw objects and add custom markers to visualisation
- Start to form some very basic Clusters
- Begin to think about some real clustering logic and experiment with alg from LArContent

Algorithm Configuration

```
/**
 *  @file   PandoraSDK/include/Helpers/XmlHelper.h
 *
 *  @brief  Header file for the xml helper class.
 *
 *  $Log: $
 */
#ifndef PANDORA_XML_HELPER_H
#define PANDORA_XML_HELPER_H 1

#include "Objects/CartesianVector.h"
#include "Objects/TrackState.h"

#include "Pandora/PandoraInternal.h"
#include "Pandora/StatusCodes.h"

#include "Xml/tinyxml.h"

namespace pandora
{

/**
 *  @brief  XmlHelper class
 */
class XmlHelper
{
public:
    /**
     *  @brief  Read a value from an xml element
     *
     *  @param  xmlHandle the relevant xml handle
     *  @param  xmlElementName the name of the xml element to examine
     *  @param  t to receive the value
     */
    template <typename T>
    static StatusCode ReadValue(const TiXmlHandle &xmlHandle, const std::string &xmlElementName, T &t);

    /**
     *  @brief  Read a vector of values from a (space separated) list in an xml element
     *
     *  @param  xmlHandle the relevant xml handle
     *  @param  xmlElementName the name of the xml element to examine
     *  @param  vector to receive the vector of values
     */
    template <typename T>
    static StatusCode ReadVectorOfValues(const TiXmlHandle &xmlHandle, const std::string &xmlElementName, std::vector<T> &vector);
```

Use these static helper functions to read parameters from XML, remembering to check the return values

Algorithm Configuration

```
/**
 *  @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  Default 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
};

} // namespace workshop_content

#endif // #ifndef WORKSHOP_MYTEST_ALGORITHM_H
```

Configurable parameters are typically member variables. Add a default constructor to assign default values.

Convention is then for const alg member functions, don't change any of these values

Algorithm Configuration

Assign default values upon construction

```
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;
}
```

Print out values at run time

Optional and mandatory reads

Algorithm Configuration

Try to run before adding required XML configuration:

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

```
XmlHelper::ReadValue(xmlHandle, "MyMandatoryString", m_myMandatoryString) return STATUS_CODE_NOT_FOUND
    in function: ReadSettings
    in file:     /path/WorkshopContent/workshopcontent/Algorithms/MyTestAlgorithm.cc line#: 49
pLocalAlgorithm->ReadSettings(TiXmlHandle(pXmlElement)) throw STATUS_CODE_NOT_FOUND
    in function: CreateAlgorithm
    in file:     /path/PandoraPFA/PandoraSDK-v02-03-00/src/Managers/AlgorithmManager.cc line#: 117
Failure in reading pandora settings, STATUS_CODE_NOT_FOUND
PandoraApi::ReadSettings(*pPandora, parameters.m_pandoraSettingsFile) throw STATUS_CODE_FAILURE
    in function: main
    in file:     /path/WorkshopContent/workshopcontent/Test/PandoraWorkshop.cc line#: 80
Pandora Exception caught: STATUS_CODE_FAILURE
```

Algorithm Configuration

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

    <!-- ALGORITHM SETTINGS -->
    <algorithm type = "LArEventReading">
        <EventFileName>/path/to/Events_MicroBooNE.xml</EventFileName>
        <GeometryFileName>/path/to/Geometry_MicroBooNE.xml</GeometryFileName>
        <ShouldReadEvents>true</ShouldReadEvents>
        <ShouldReadGeometry>true</ShouldReadGeometry>
        <SkipToEvent>0</SkipToEvent>
    </algorithm>

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

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

    <algorithm type = "LArVisualMonitoring">
        <CaloHitListNames>CaloHitListW CaloHitListU CaloHitListV</CaloHitListNames>
        <MCParticleListNames>MCParticleList3D</MCParticleListNames>
        <SuppressMCParticles>22:0.01 2112:1.0</SuppressMCParticles>
        <ShowDetector>true</ShowDetector>
    </algorithm>
</pandora>
```

Note: haven't specified a value for the optional Boolean here, but have changed the default value for the optional unsigned int

$MY_TEST_AREA/WorkshopContent/scripts/PandoraSettings_Workshop.xml

Algorithm Configuration

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

```
> Running Algorithm: 0x7fc714da02b0, LArEventReading
> Running Algorithm: 0x7fc720d47930, LArListPreparation
ListPreparationAlgorithm: found a hit with zero energy, will remove it
> Running Algorithm: 0x7fc71f6fc020, MyTest
-m_myMandatoryString: TestString
-m_myOptionalBool: 0
-m_myOptionalUnsignedInt: 10
-m_myMandatoryString: 0 1.5 3 4.5
> Running Algorithm: 0x7fc720d47b00, LArVisualMonitoring
```

Pandora Content APIs

Operations that can be performed in algorithms will consist of:

- APIs to access or modify Pandora content or to request visualisation
- Use of services provided by objects in the Pandora Event Data Model
- Calls to static Helper functions, some provided in the SDK, more in content libraries
- Use of constructs local to the algorithm

For the first three bullet-points, the starting point for the algorithm author is the header file describing the relevant interfaces.

Pandora Content APIs

Example APIs for accessing lists of objects in the Pandora EDM. Can ask for the "current" list, specified by prior algorithm (useful for clever algorithm interplay) or just request a named list.

```
/* List-manipulation functions */

/**
 *  @brief  Get the current list
 *
 *  @param  algorithm the algorithm calling this function
 *  @param  pT to receive the address of the current list
 */
template <typename T>
static pandora::StatusCode GetCurrentList(const pandora::Algorithm &algorithm, const T *&pT);

/**
 *  @brief  Get the current list
 *
 *  @param  algorithm the algorithm calling this function
 *  @param  pT to receive the address of the current list
 *  @param  listName to receive the current list name
 */
template <typename T>
static pandora::StatusCode GetCurrentList(const pandora::Algorithm &algorithm, const T *&pT, std::string &listName);

/**
 *  @brief  Get a named list
 *
 *  @param  algorithm the algorithm calling this function
 *  @param  listName the name of the list
 *  @param  pT to receive the address of the list
 */
template <typename T>
static pandora::StatusCode GetList(const pandora::Algorithm &algorithm, const std::string &listName, const T *&pT);
```

$MY_TEST_AREA/PandoraPFA/PandoraSDK-v02-03-00/include/Api/PandoraContentApi.h

Many other APIs available - starting point for key event management operations

List Access

Example access of current CaloHit list, followed by creation of a local, sorted list of CaloHit addresses and some print statements.

```
/**
 *  @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;
}
```

Managed object lists are `unordered_sets` highly efficient, but user must be careful

Note: use of preprocessor macro to check API call return value

Note: first API argument reference to alg, redirects to relevant Pandora instance

Note first usage of functionality from the LArContent library, `lar_content::LArClusterHelper`, to perform an optional CaloHit sort

List Access

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

```
> Running Algorithm: 0x7f890dbf0d00, LArEventReading
> Running Algorithm: 0x7f8919dcbb90, LArListPreparation
ListPreparationAlgorithm: found a hit with zero energy, will remove it
> Running Algorithm: 0x7f8919dcbd40, MyTest
InputHit - HitType: 6,   x: 158.263  y: 0  z: 209.65 length: 262.679
InputHit - HitType: 6,   x: 158.316  y: 0  z: 209.95 length: 262.95
InputHit - HitType: 6,   x: 158.428  y: 0  z: 210.25 length: 263.257
InputHit - HitType: 6,   x: 158.341  y: 0  z: 210.55 length: 263.445
InputHit - HitType: 6,   x: 158.174  y: 0  z: 210.85 length: 263.584
InputHit - HitType: 6,   x: 76.3678  y: 0  z: 250.15 length: 261.547
InputHit - HitType: 6,   x: 94.3723  y: 0  z: 253.75 length: 270.731
InputHit - HitType: 6,   x: 94.2191  y: 0  z: 254.05 length: 270.959
InputHit - HitType: 6,   x: 94.8458  y: 0  z: 254.05 length: 271.177
InputHit - HitType: 6,   x: 94.1454  y: 0  z: 254.35 length: 271.214
InputHit - HitType: 6,   x: 94.8531  y: 0  z: 254.35 length: 271.461
InputHit - HitType: 6,   x: 183.866  y: 0  z: 263.65 length: 321.431
InputHit - HitType: 6,   x: 183.903  y: 0  z: 263.95 length: 321.698
    ...
```

Try also reading the named CaloHit lists defined by the `ListPreparation` algorithm: the first algorithm in the file `PandoraSettings_Workshop.xml`

For more examples, see `$MY_TEST_AREA/WorkshopContent/examplecontent/ExampleAlgorithms/AccessListsAlgorithm.cc` or `.h`

List Access

Add example access of current MCParticle list:

```
#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;
    }

    // MCParticles
    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::SortBySource);

    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;
}
```

Visualisation APIs

```
/**
 *  @brief  Add MCParticles to the Eve event-display
 *
 *  @param  pandora the calling pandora instance
 *  @param  pMCParticleList list of MC particles to be added to the event display
 *  @param  name of the MC particle list
 *  @param  color The color the track elements are drawn with
 *  @param  pParticleSuppressionMap map from pdg-codes to energy for suppression of particles types below specific energies
 */
static void VisualizeMCParticles(const pandora::Pandora &pandora, const pandora::MCParticleList *const pMCParticleList,
    const std::string &name, const Color color, const PdgCodeToEnergyMap *pParticleSuppressionMap = NULL);

/**
 *  @brief Add CaloHits to the Eve event-display
 *
 *  @param  pandora the calling pandora instance
 *  @param pCaloHitList list of calohits to be added to the event display
 *  @param name of the calohit list
 *  @param color The color the cluster elements are drawn with
 */
static void VisualizeCaloHits(const pandora::Pandora &pandora, const pandora::CaloHitList *const pCaloHitList,
    const std::string &name, const Color color);

/**
 *  @brief  Add Clusters to the Eve event-display
 *
 *  @param  pandora the calling pandora instance
 *  @param  pClusterList list of clusters to be added to the event display
 *  @param  name of the cluster list
 *  @param  color The color the cluster elements are drawn with
 *  @param  showAssociatedTracks draw the tracks associated to the cluster
 */
static void VisualizeClusters(const pandora::Pandora &pandora, const pandora::ClusterList *const pClusterList,
    const std::string &name, const Color color, bool showAssociatedTracks = false);
```

`$MY_TEST_AREA/PandoraPFA/PandoraMonitoring-v02-03-00/include/PandoraMonitoringApi.h`

Many other APIs available - starting point for all visualisation and tree-writing operations

Visualisation APIs

Request visualisation of current CaloHits and MCParticles:

```
/**
 *  @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()
{
    // 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);

    // MCParticles
    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;
}
```

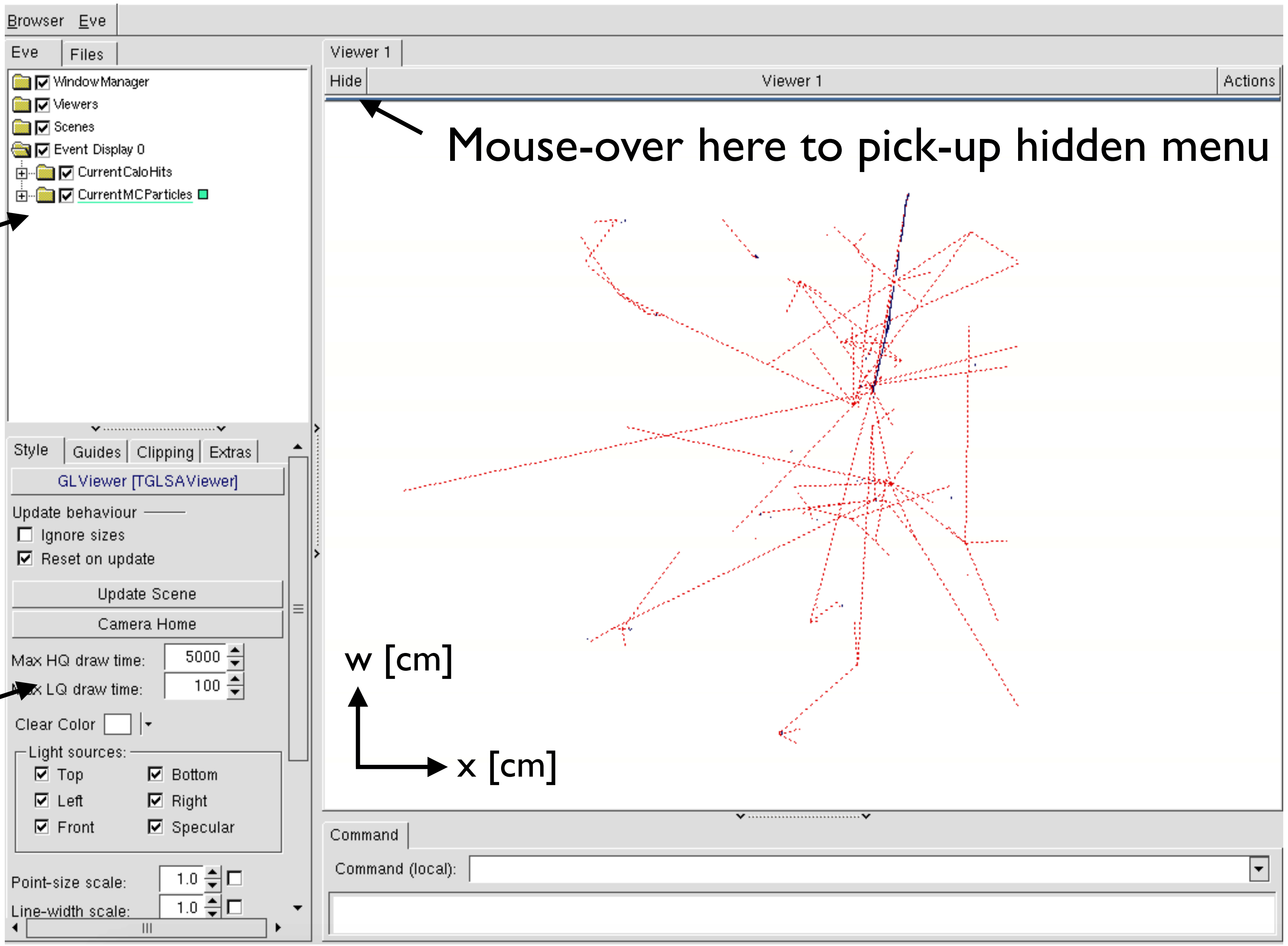
For more examples, see `$MY_TEST_AREA/WorkshopContent/examplecontent/ExampleAlgorithms/DisplayListsAlgorithm.cc` or `.h`

Visualisation APIs

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

Can explore object lists here

Lighting and guides/axes



Update behaviour
☐ Ignore sizes
☑ Reset on update

Max HQ draw time: 5000
Max LQ draw time: 100
Clear Color

Light sources:
☑ Top ☑ Bottom
☑ Left ☑ Right
☑ Front ☑ Specular

Point-size scale: 1.0 ☐
Line-width scale: 1.0 ☐

Try to make display more useful by e.g. suppressing some MCParticles or e.g. adding a custom marker to identify the neutrino interaction vertex. Try looking at 3D view too.

Cluster Creation

Create clusters, with simple logic (take next *n* Hits from sorted list for successive Clusters):

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

    const ClusterList *pTemporaryList(nullptr);
    std::string temporaryListName;
    PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::CreateTemporaryListAndSetCurrent(*this, pTemporaryList, temporaryListName));

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

    const Cluster *pCluster(nullptr);

    for (const CaloHit *const pCaloHit : sortedCaloHits)
    {
        if (!PandoraContentApi::IsAvailable(*this, pCaloHit))
            continue;

        if (!pCluster || (pCluster->GetNCaloHits() >= m_nHitsPerCluster))
        {
            PandoraContentApi::Cluster::Parameters parameters;
            parameters.m_caloHitList.insert(pCaloHit);
            PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::Cluster::Create(*this, parameters, pCluster));
        }
        else
        {
            PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::AddToCluster(*this, pCluster, pCaloHit));
        }
    }

    if (!pTemporaryList->empty())
    {
        PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::SaveList<Cluster>(*this, m_outputClusterListName));
        PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::ReplaceCurrentList<Cluster>(*this, m_outputClusterListName));
    }

    return STATUS_CODE_SUCCESS;
}
```

Discussed in more detail over next few slides

Cluster Creation

Supporting implementation, for reference

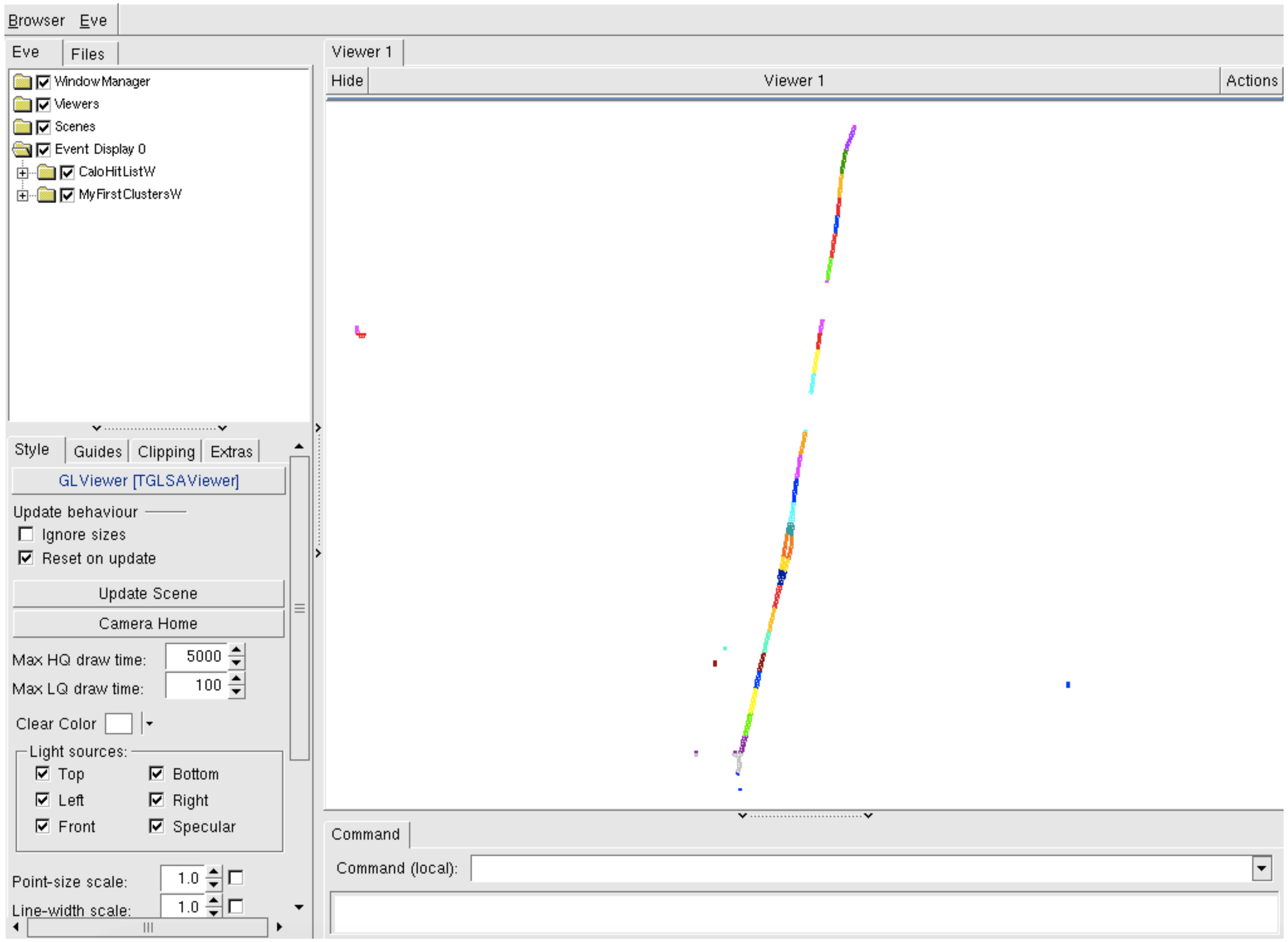
```
/**
 *  @file   WorkshopContent/workshopcontent/Algorithms/MyTestAlgorithm.cc
 *
 *  @brief  Implementation of the mytest algorithm class.
 *
 *  $Log: $
 */

#include "Pandora/AlgorithmHeaders.h"

#include "larpandoracontent/LArHelpers/LArClusterHelper.h"

#include "workshopcontent/Algorithms/MyTestAlgorithm.h"

using namespace pandora;
using namespace lar_content;

namespace workshop_content
{

MyTestAlgorithm::MyTestAlgorithm() :
    m_outputClusterListName(),
    m_nHitsPerCluster(10)
{
}

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

StatusCode MyTestAlgorithm::ReadSettings(const TiXmlHandle xmlHandle)
{
    PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, XmlHelper::ReadValue(xmlHandle,
        "OutputClusterListName", m_outputClusterListName));

    PANDORA_RETURN_RESULT_IF_AND_IF(STATUS_CODE_SUCCESS, STATUS_CODE_NOT_FOUND, !=, XmlHelper::ReadValue(xmlHandle,
        "NHitsPerCluster", m_nHitsPerCluster));

    return STATUS_CODE_SUCCESS;
}
```

In "workshopcontent/Algorithms/MyTestAlgorithm.h"

```
// Member variables here
std::string     m_outputClusterListName;    ///< The output cluster list name
unsigned int    m_nHitsPerCluster;          ///< The number of hits to add to each dummy cluster
```

Cluster Visualisation

```
<algorithm type = "MyTest">
    <OutputClusterListName>MyFirstClustersW</OutputClusterListName>
</algorithm>

<algorithm type = "LArVisualMonitoring">
    <CaloHitListNames>CaloHitListW</CaloHitListNames>
    <ClusterListNames>MyFirstClustersW</ClusterListNames>
</algorithm>
```



Cluster Creation

// Algorithms must either create a temporary list for newly created clusters, or ask to run a daughter clustering algorithm
// (temporary list, owned by parent algorithm is then created automatically for you). Any Clusters remaining in a temporary
// list at the end of the algorithm will be deleted, so all desired clusters must be saved before the algorithm ends.

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

    const ClusterList *pTemporaryList(nullptr);
    std::string temporaryListName;
    PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::CreateTemporaryListAndSetCurrent(*this, pTemporaryList, temporaryListName));

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

    const Cluster *pCluster(nullptr);

    for (const CaloHit *const pCaloHit : sortedCaloHits)
    {
        if (!PandoraContentApi::IsAvailable(*this, pCaloHit))
            continue;

        if (!pCluster || (pCluster->GetNCaloHits() >= m_nHitsPerCluster))
        {
            PandoraContentApi::Cluster::Parameters parameters;
            parameters.m_caloHitList.insert(pCaloHit);
            PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::Cluster::Create(*this, parameters, pCluster));
        }
        else
        {
            PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::AddToCluster(*this, pCluster, pCaloHit));
        }
    }

    if (!pTemporaryList->empty())
    {
        PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::SaveList<Cluster>(*this, m_outputClusterListName));
        PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::ReplaceCurrentList<Cluster>(*this, m_outputClusterListName));
    }

    return STATUS_CODE_SUCCESS;
}
```

Cluster Creation

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

    const ClusterList *pTemporaryList(nullptr);
    std::string temporaryListName;
    PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::CreateTemporaryListAndSetCurrent(*this, pTemporaryList, temporaryListName));

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

    const Cluster *pCluster(nullptr);

    for (const CaloHit *const pCaloHit : sortedCaloHits)
    {
        if (!PandoraContentApi::IsAvailable(*this, pCaloHit))
            continue;

        if (!pCluster || (pCluster->GetNCaloHits() >= m_nHitsPerCluster))
        {
            PandoraContentApi::Cluster::Parameters parameters;
            parameters.m_caloHitList.insert(pCaloHit);
            PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::Cluster::Create(*this, parameters, pCluster));
        }
        else
        {
            PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::AddToCluster(*this, pCluster, pCaloHit));
        }
    }

    if (!pTemporaryList->empty())
    {
        PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::SaveList<Cluster>(*this, m_outputClusterListName));
        PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::ReplaceCurrentList<Cluster>(*this, m_outputClusterListName));
    }

    return STATUS_CODE_SUCCESS;
}
```

// Once a calo hit has been added to a cluster, it is flagged as unavailable.
// Important example of Pandora book-keeping, preventing double-counting

Cluster Creation

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

    const ClusterList *pTemporaryList(nullptr);
    std::string temporaryListName;
    PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::CreateTemporaryListAndSetCurrent(*this, pTemporaryList, temporaryListName));

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

    const Cluster *pCluster(nullptr);

    for (const CaloHit *const pCaloHit : sortedCaloHits)
    {
        if (!PandoraContentApi::IsAvailable(*this, pCaloHit))
            continue;

        if (!pCluster || (pCluster->GetNCaloHits() >= m_nHitsPerCluster))
        {
            PandoraContentApi::Cluster::Parameters parameters;
            parameters.m_caloHitList.insert(pCaloHit);
            PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::Cluster::Create(*this, parameters, pCluster));
        }
        else
        {
            PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::AddToCluster(*this, pCluster, pCaloHit));
        }
    }

    if (!pTemporaryList->empty())
    {
        PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::SaveList<Cluster>(*this, m_outputClusterListName));
        PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::ReplaceCurrentList<Cluster>(*this, m_outputClusterListName));
    }

    return STATUS_CODE_SUCCESS;
}
```

// Create new Cluster if not yet done so, or current Cluster has reached
// configurable target number of Hits, else add Hit to current Cluster.

Cluster Creation

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

    const ClusterList *pTemporaryList(nullptr);
    std::string temporaryListName;
    PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::CreateTemporaryListAndSetCurrent(*this, pTemporaryList, temporaryListName));

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

    const Cluster *pCluster(nullptr);

    for (const CaloHit *const pCaloHit : sortedCaloHits)
    {
        if (!PandoraContentApi::IsAvailable(*this, pCaloHit))
            continue;

        if (!pCluster || (pCluster->GetNCaloHits() >= m_nHitsPerCluster))
        {
            PandoraContentApi::Cluster::Parameters parameters;
            parameters.m_caloHitList.insert(pCaloHit);
            PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::Cluster::Create(*this, parameters, pCluster));
        }
        else
        {
            PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::AddToCluster(*this, pCluster, pCaloHit));
        }
    }

    if (!pTemporaryList->empty())
    {
        PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::SaveList<Cluster>(*this, m_outputClusterListName));
        PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::ReplaceCurrentList<Cluster>(*this, m_outputClusterListName));
    }

    return STATUS_CODE_SUCCESS;
}
```

// Choose to save all the temporary clusters under a specified name and to set as the current list.
// All Clusters left in temporary list at end of parent algorithm's operations are deleted (alongside list)

LArContent - TrackClusterCreation

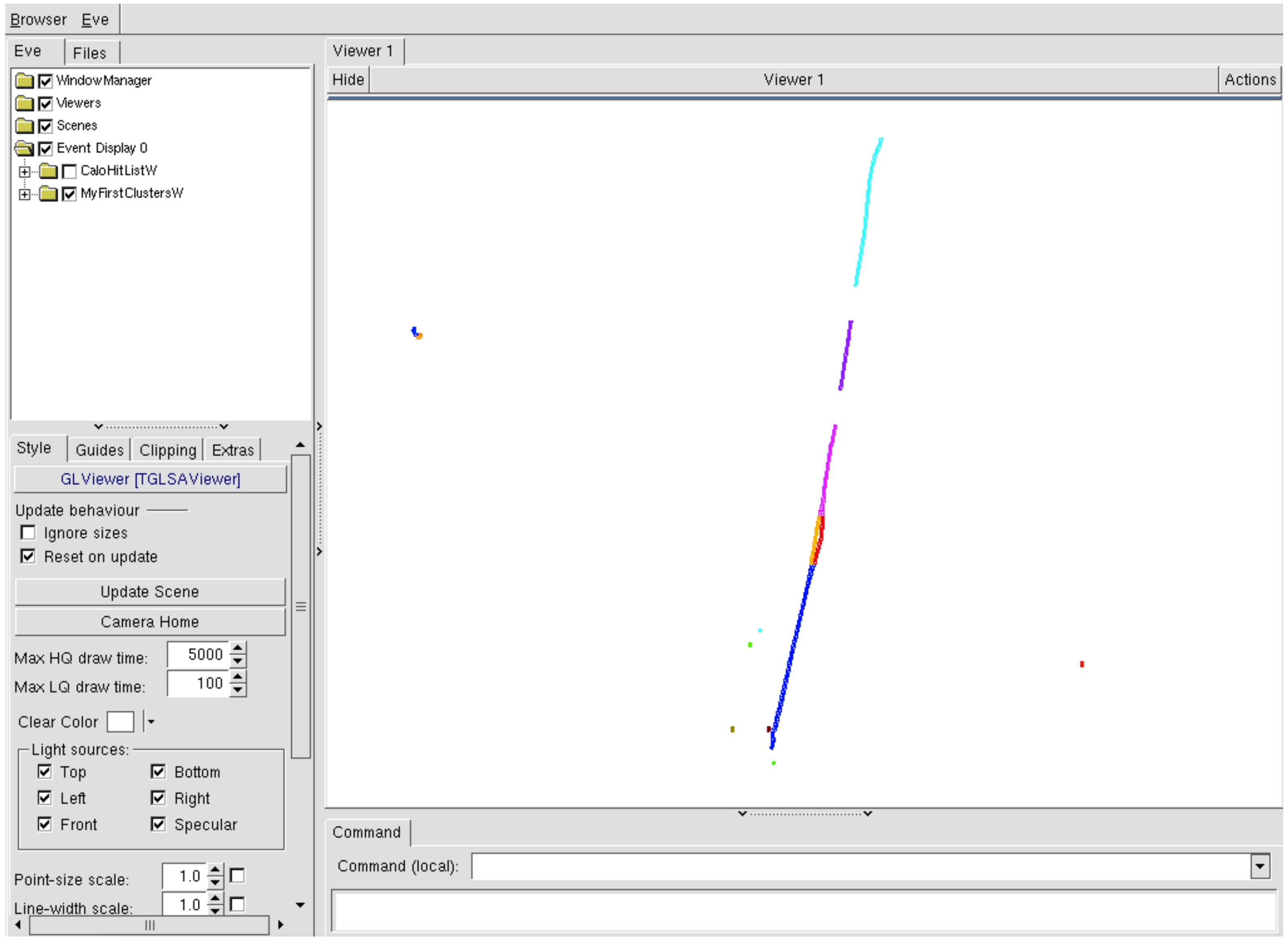
```
<pandora>
    <!-- GLOBAL SETTINGS -->
    <IsMonitoringEnabled>true</IsMonitoringEnabled>
    <ShouldDisplayAlgorithmInfo>true</ShouldDisplayAlgorithmInfo>
    <SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>

    <!-- ALGORITHM SETTINGS -->
    <algorithm type = "LArEventReading">
        <EventFileName>/path/to/Events_MicroBooNE.xml</EventFileName>
        <GeometryFileName>/path/to/Geometry_MicroBooNE.xml</GeometryFileName>
        <ShouldReadEvents>true</ShouldReadEvents>
        <ShouldReadGeometry>true</ShouldReadGeometry>
        <SkipToEvent>0</SkipToEvent>
    </algorithm>

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

    <algorithm type = "LArClusteringParent">
        <algorithm type = "LArTrackClusterCreation" description = "ClusterFormation"/>
        <InputCaloHitListName>CaloHitListW</InputCaloHitListName>
        <ClusterListName>MyFirstClustersW</ClusterListName>
        <ReplaceCurrentCaloHitList>false</ReplaceCurrentCaloHitList>
        <ReplaceCurrentClusterList>true</ReplaceCurrentClusterList>
    </algorithm>

    <algorithm type = "LArVisualMonitoring">
        <CaloHitListNames>CaloHitListW</CaloHitListNames>
        <ClusterListNames>MyFirstClustersW</ClusterListNames>
    </algorithm>
</pandora>
```

Get a feel for how default Pandora LAr TPC clustering performs by using it replace MyTest algorithm.

For reasons of alg re-use, TrackClusterCreation alg runs via a parent alg.

Completely separates Cluster creation logic (daughter) from list management (parent)

LArContent - TrackClusterCreation



`$MY_TEST_AREA/PandoraPFA/LArContent-v02_07_04/larpandoracontent/LArTwoDReco/LArClusterCreation/`
`TrackClusterCreationAlgorithm.cc or .h`

Try to get idea of logic flow in a "real" algorithm and start to make test clustering algorithm do something a bit more substantial.

Next Exercise: Write a Cluster Merging Algorithm