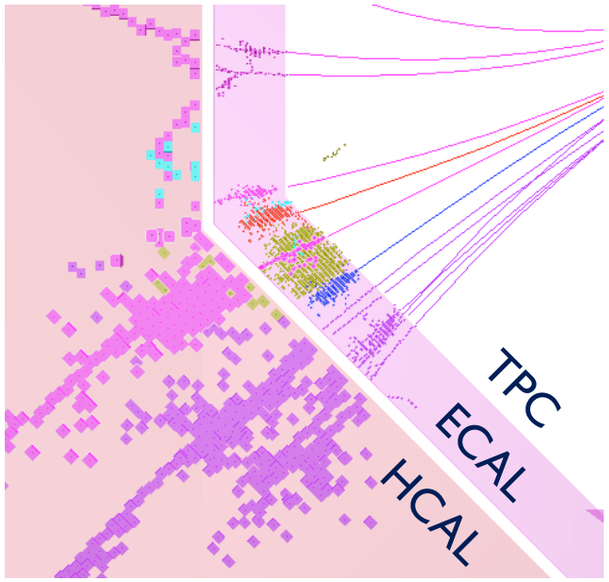


Pandora

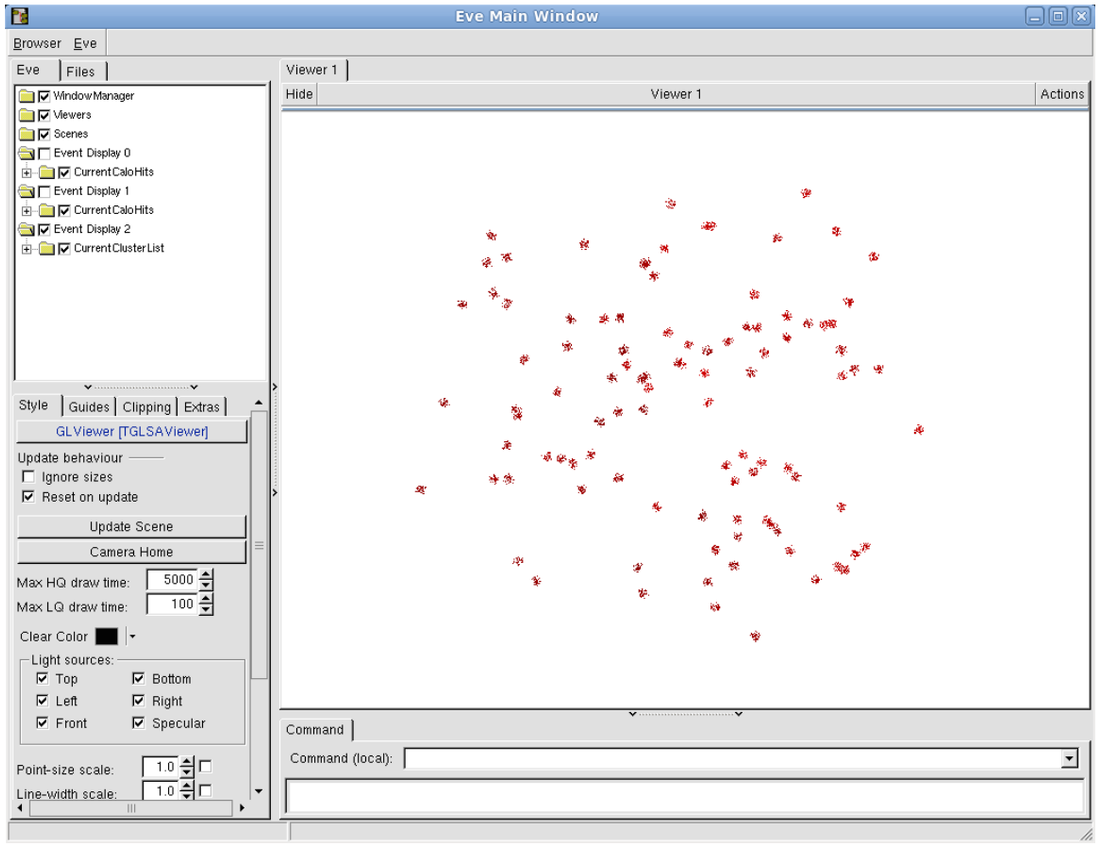
ExampleContent library and Test Application

J. S. Marshall

University of Cambridge

Pandora Example Content

- Pandora pattern-recognition algorithms aim to form clusters, merge or delete clusters, split up clusters, form particle flow objects, etc.
- However, the decision whether to proceed with particular operations can be complex and use-case specific (the decisions should be the main body of the algorithms).
- The aim of the Pandora ExampleContent library and test application is to demonstrate key Pandora functionality in a much more simple environment.



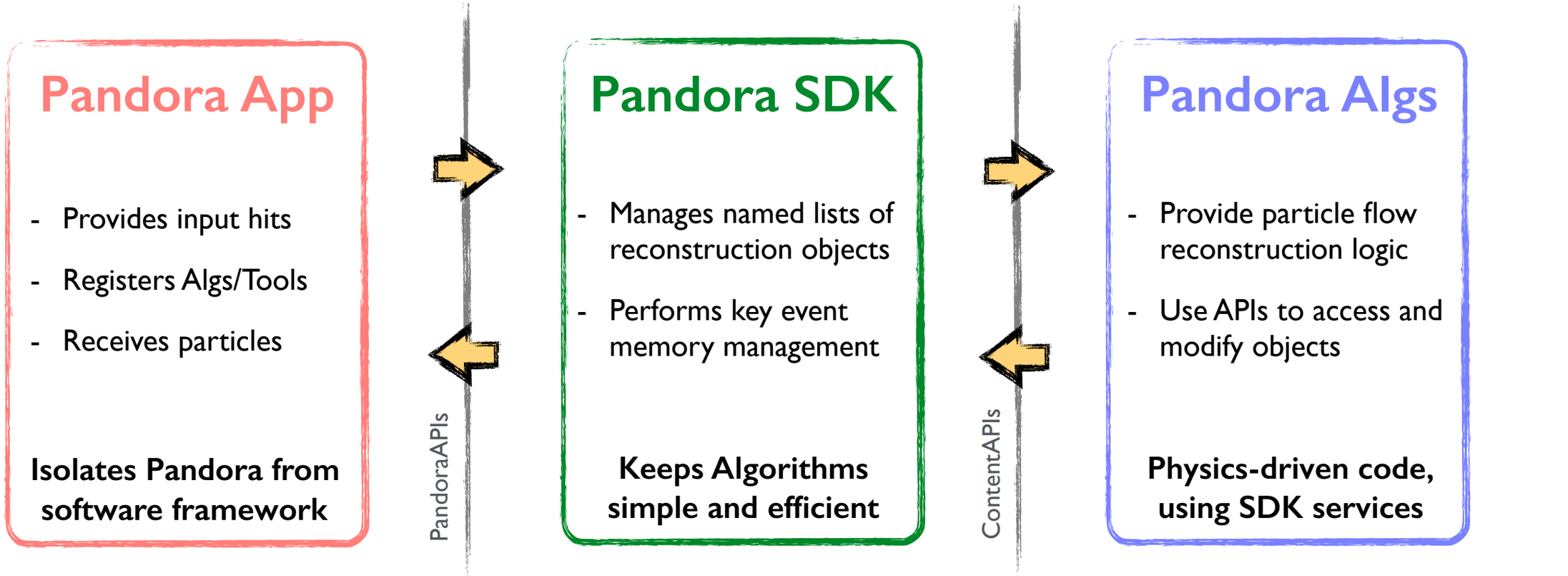
Example clusters, as displayed by the DisplayLists example algorithm (using the Pandora Monitoring APIs).

Pandora Approach

The Pandora Software Development Kit provides a software environment in which:

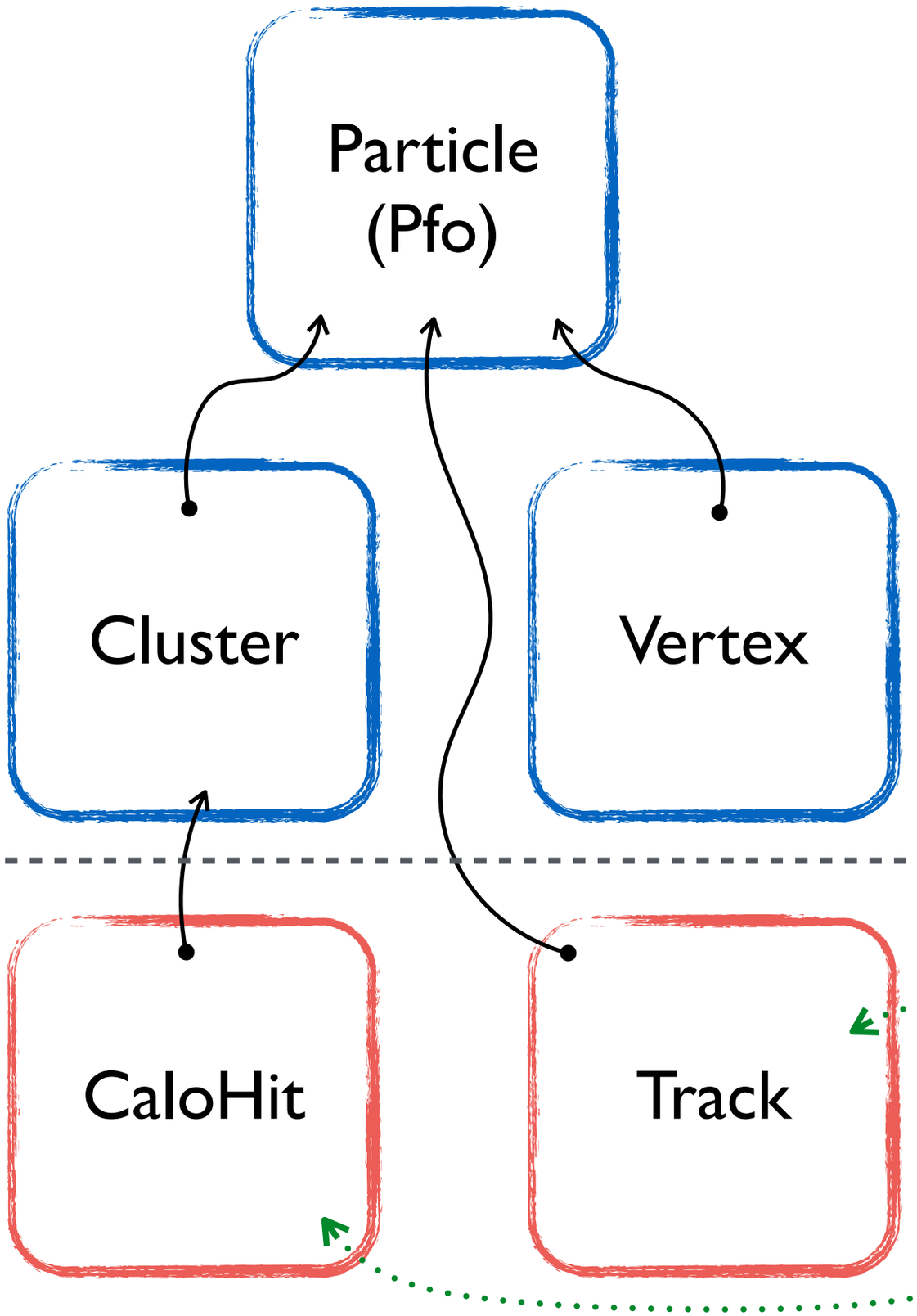
1. It is easy for users to provide the building-blocks that define a pattern recognition problem.
2. Logic required to solve pattern recognition problems is cleanly implemented in algorithms.
3. Operations to access or modify building-blocks, or to create new structures, are requested by algorithms and performed by the Pandora framework.

This promotes the use of large numbers of algorithms, each addressing specific event topologies.



https://github.com/PandoraPFA

Pandora Event Data Model



Daughter object can be added to parent object

MC Particle Link

Algorithm objects
Created by Algs

Input objects
Created by Client App

Pandora Example Content

- The simple PandoraExample application creates a Pandora instance and a configurable number of dummy CaloHits in a World volume.
- The dummy CaloHits are processed by example Algorithms, which are built as part of an ExampleContent library:
  - Example list access and display
  - Example Cluster, Vertex and Pfo creation
  - Cluster manipulation, including merging, deletion, fragmentation and reclustering
  - Creating and saving new lists of objects
  - Using Algorithm Tools and Plugins
  - Writing a tree using PandoraMonitoring.

- The ExampleContent library is structured in exactly the same manner as the LCContent and LArContent libraries, currently in use for Linear Collider and LAr TPC reconstruction.
- The Content libraries consist of Algorithms, AlgorithmTools, Plugins, Helper functions and a header file that allows a Client App to quickly register all the content with Pandora.

Pandora Build Instructions

Use of PandoraMonitoring requires a ROOT installation including TEve libraries.
To install without ROOT, please ignore all text highlighted in blue.

1. Ensure correct environment (compiler, ROOT, etc.), and set installation path: MY_PANDORA_PATH

2. There are many ways to build Pandora e.g. using git, CMake and the PandoraPFA build package:

```
git clone https://github.com/PandoraPFA/PandoraPFA.git $MY_PANDORA_PATH
cd $MY_PANDORA_PATH
git tag # you probably want to checkout the latest tag; see PandoraPFA documentation on GitHub
git checkout v03-04-02 # e.g. v03-04-02 at time of writing
mkdir build
cd build
cmake -DEXAMPLE_PANDORA_CONTENT=ON -DPANDORA_MONITORING=ON -DROOT_DIR=/path/to/root/etc/cmake/ ..
make -j4 install
```

3. You're ready to run the PandoraExample application and use the ExampleContent library!

Pandora Application

- At the terminal, you can ask the PandoraExample application to display its help menu:

```
bash-3.2$ $MY_PANDORA_PATH/bin/PandoraExample -h

./bin/PandoraExample
    -i PandoraSettings.xml   (mandatory)
    -n NEventsToProcess      (optional)
    -g NHitGroupings         (optional)
    -p NHitsPerGroup         (optional)
    -w WorldSideLength       (optional, may require algorithm retuning)
    -s GroupSideLength       (optional, may require algorithm retuning)
```

- The suggested settings file is located in the ExampleContent scripts directory: $MY_PANDORA_PATH/ExampleContent-$MY_VERSION/ExamplePandoraSettings.xml
- The application will generate dummy hits in a World volume cube of configurable size. The hits will be distributed randomly, but will be divided into groups (also of configurable size).
- Default values are provided for all but the settings argument. Note that changing the size of the World volume, or the spread of each group of hits may require tuning of example Algorithms.

Pandora Access Lists

<algorithm type="AccessListsExample"/> ⟵ Relevant xml snippet

- The idea is to read the actual header file and implementation files for each of the example Pandora Algorithms provided.
- These files will contain functioning code-snippets, with minimal surrounding logic, together with useful commenting.
- The AccessLists example demonstrates how to access the "current" lists for each of the types in the Pandora Event Data Model. Access of a named CaloHit list is also demonstrated.

ExampleContent/src/ExampleAlgorithms/AccessListsAlgorithm.cc
ExampleContent/include/ExampleAlgorithms/AccessListsAlgorithm.h

```
const ClusterList *pCurrentClusterList(NULL);
PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::GetCurrentList(*this, pCurrentClusterList));
```

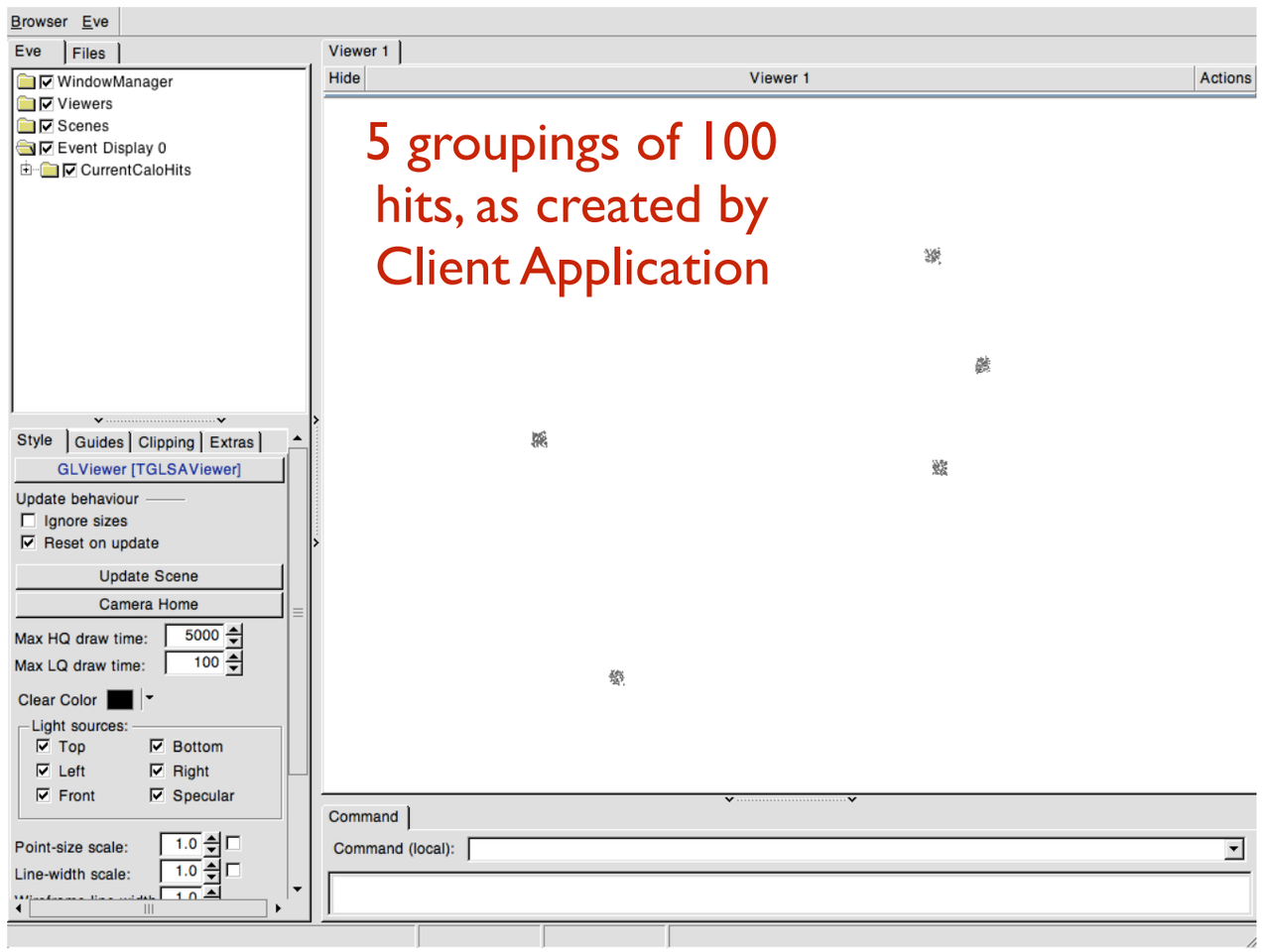
- Please note the upper-case preprocessor macro PANDORA_RETURN_RESULT_IF… which is often used to surround the API call.
- This is just a simple way to call the function and check the StatusCode returned. If the StatusCode is unexpected, a message will be printed to screen and the StatusCode is returned.

Pandora Display Lists

```
<algorithm type="DisplayListsExample">
    <DisplayCurrentCaloHits>true</DisplayCurrentCaloHits>
</algorithm>
```

- This Algorithm uses Pandora Monitoring APIs to display objects in the current lists.
- Configurable parameters are provided to specify whether to display each list.
- Parameters are member variables of the Algorithm, with default values provided in the Algorithm default constructor.
- The ReadSettings method sets the xml tag names for each parameter and determines whether each is mandatory.
- Mandatory: accept STATUS_CODE_SUCCESS
  Optional: also STATUS_CODE_NOT_FOUND

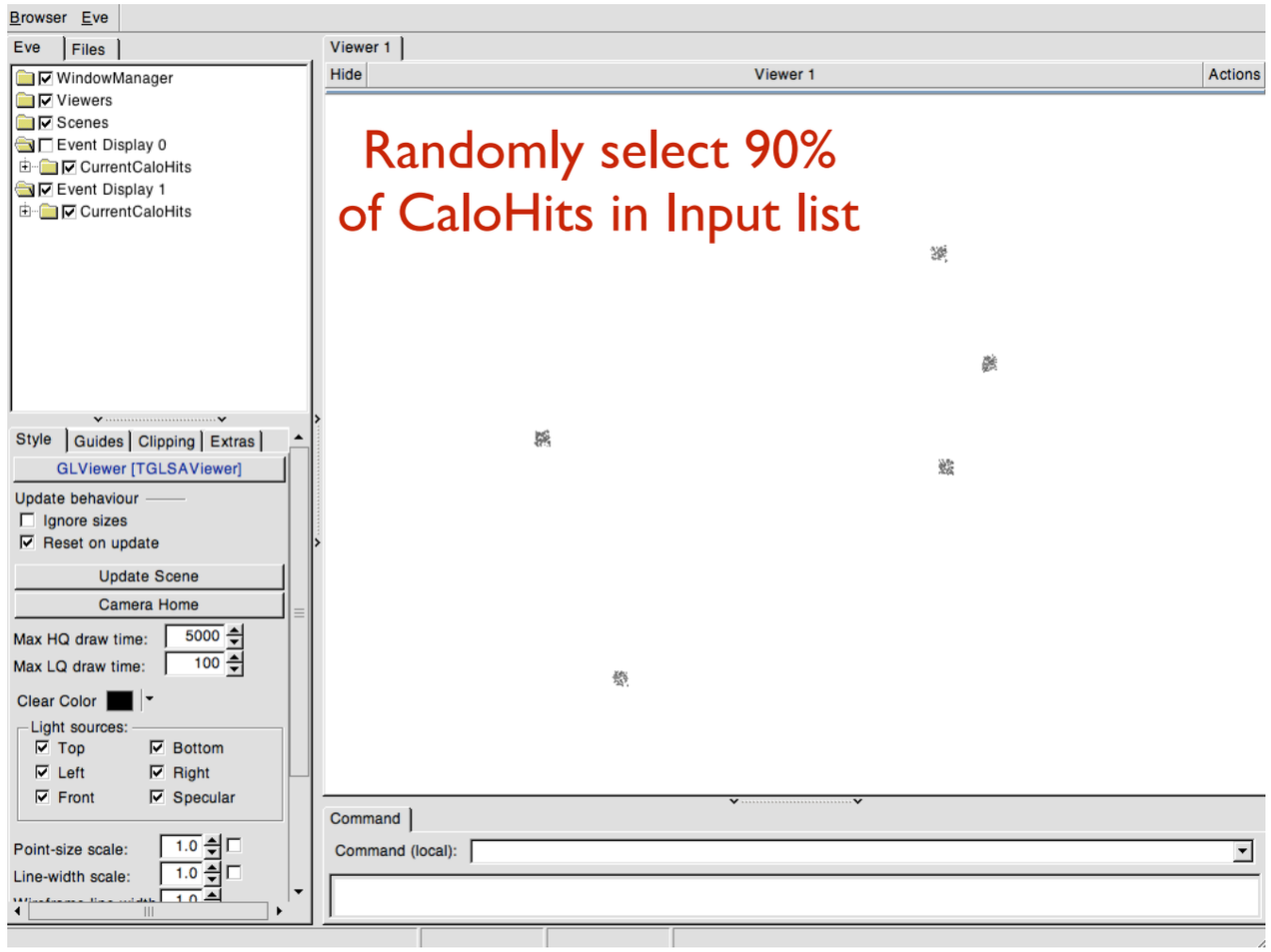
```
---500 calo hits in current list
```



Pandora Select Hit Subset

```
<algorithm type="SelectHitSubsetExample">
    <HitSelectionFraction>0.9</HitSelectionFraction>
    <OutputListName>SelectedHits</OutputListName>
</algorithm>
<algorithm type="DisplayListsExample">
    <DisplayCurrentCaloHits>true</DisplayCurrentCaloHits>
</algorithm>
```

- This Algorithm takes the current list of CaloHits and selects a specified fraction.
- A new subset CaloHit list is saved under the provided list name.
- The newly saved CaloHit list is set as the current list for subsequent Algorithms.

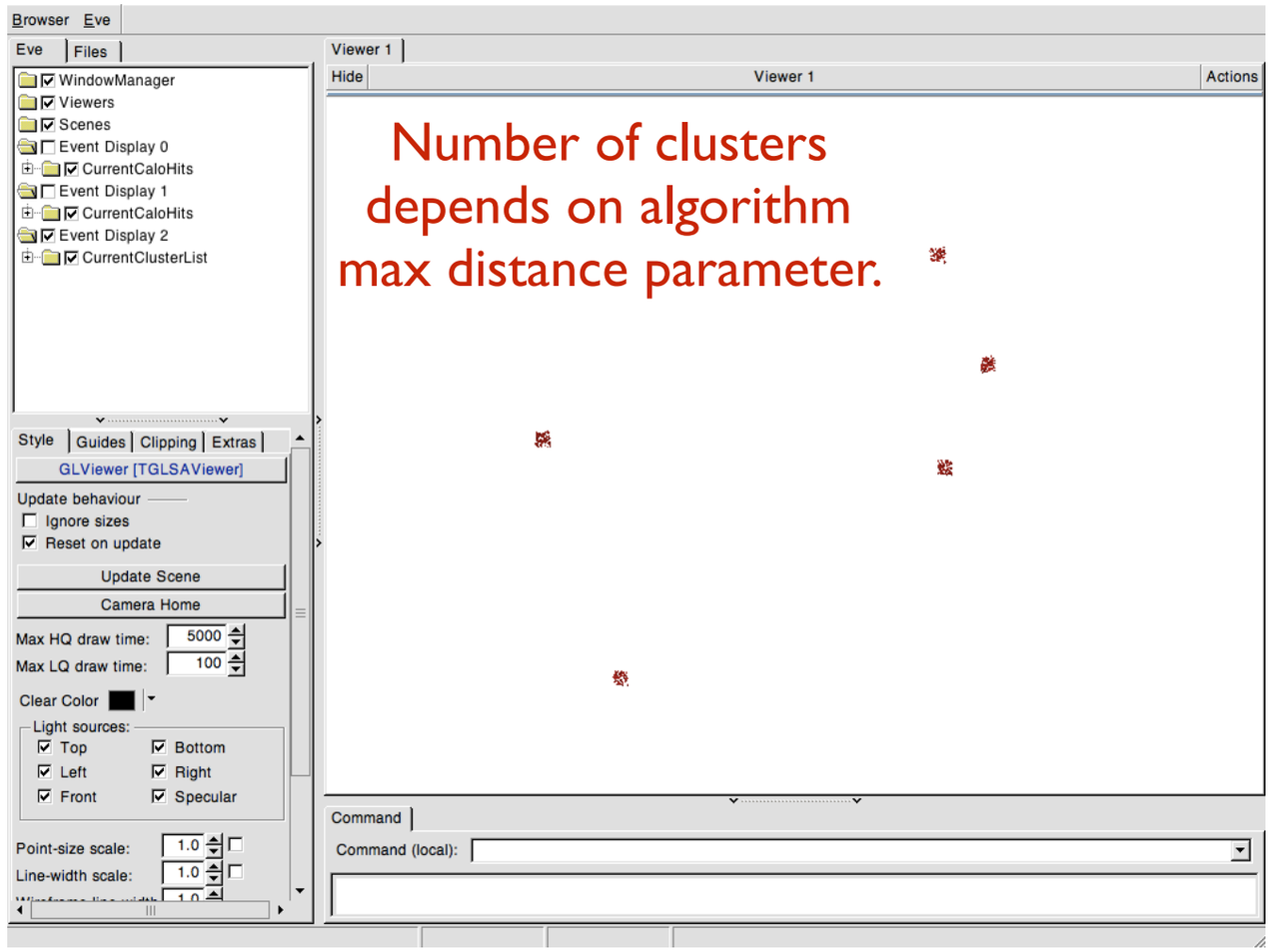


Pandora Create Clusters

```
<algorithm type="CreateClustersExample">
    <MaxClusterHitDistance>20.</MaxClusterHitDistance>
    <OutputListName>ExampleClusters</OutputListName>
</algorithm>
<algorithm type="DisplayListsExample">
    <DisplayCurrentClusters>true</DisplayCurrentClusters>
</algorithm>
```

```
---5 clusters in current list
------Cluster 0x7fd13b682420, nHits: 91
------Cluster 0x7fd13b6b98f0, nHits: 90
------Cluster 0x7fd13b6bb840, nHits: 92
------Cluster 0x7fd13b6bf030, nHits: 89
------Cluster 0x7fd13b6c0cb0, nHits: 94
```

- This Algorithm creates Clusters from the available CaloHits in the current CaloHit list.
- The first Hit encountered in the unordered CaloHit list is used to create a seed Cluster.
- Subsequent Hits are either added to an existing seed or used to form new seeds.
- The Algorithm is fully responsible for creating a temporary Cluster list, filling it with Clusters, then saving the list.

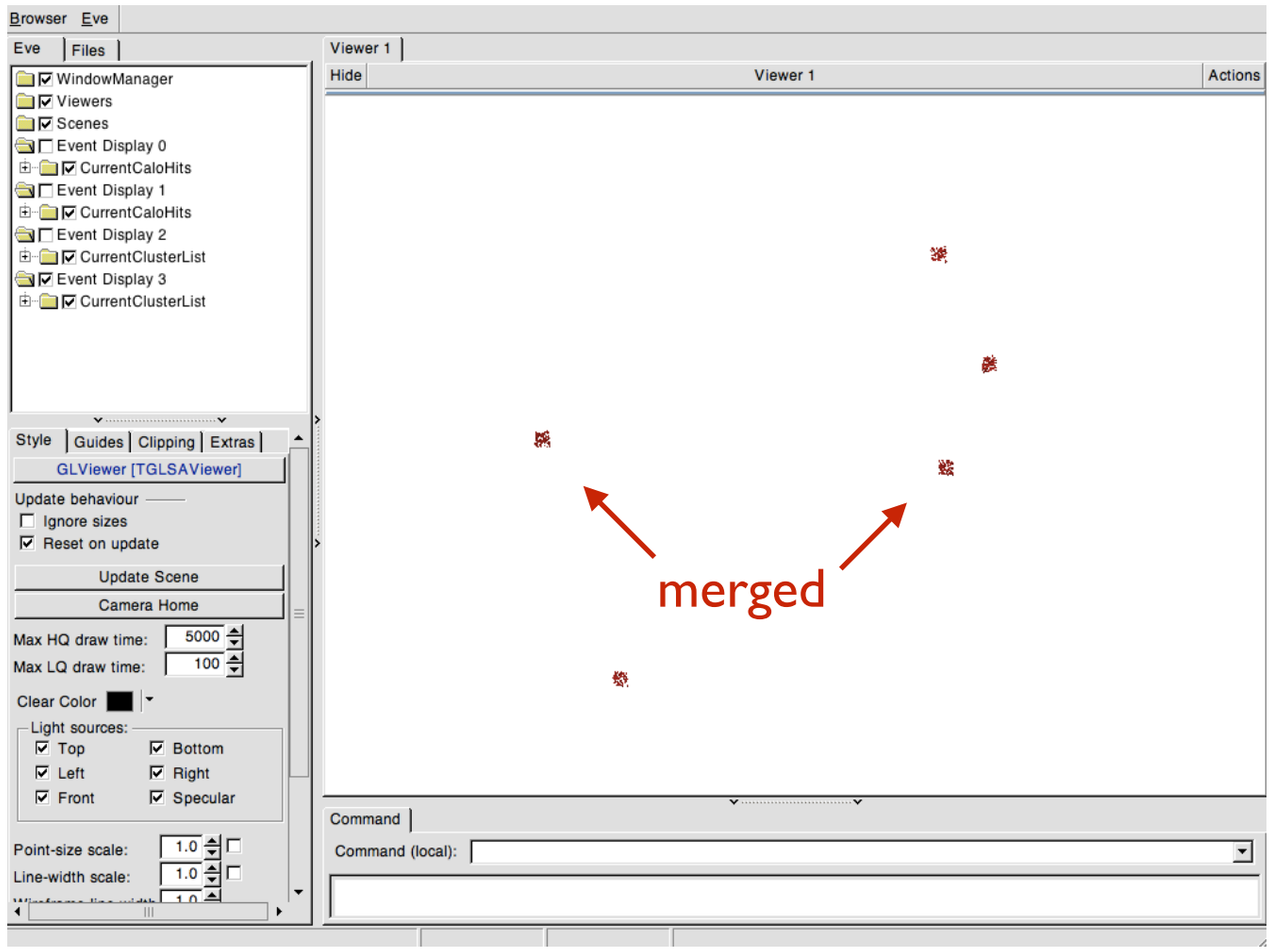


Pandora Merge Clusters

```
<algorithm type="MergeClustersExample">
    <NClusterMergesToMake>1</NClusterMergesToMake>
</algorithm>
<algorithm type="DisplayListsExample">
    <DisplayCurrentClusters>true</DisplayCurrentClusters>
</algorithm>
```

```
---4 clusters in current list
------Cluster 0x7fd13b682420, nHits: 183
------Cluster 0x7fd13b6b98f0, nHits: 90
------Cluster 0x7fd13b6bf030, nHits: 89
------Cluster 0x7fd13b6c0cb0, nHits: 94
```

- This Algorithm performs (up to) a specified number of Cluster merging operations: enlarging parent Cluster, deleting daughter.
- It takes the next Cluster in the unordered Cluster list as parent and merges it with the nearest neighbouring Cluster as daughter.
- Great care must be taken in the Algorithm: local iterators to the deleted Cluster will be invalidated and local pointers left dangling.

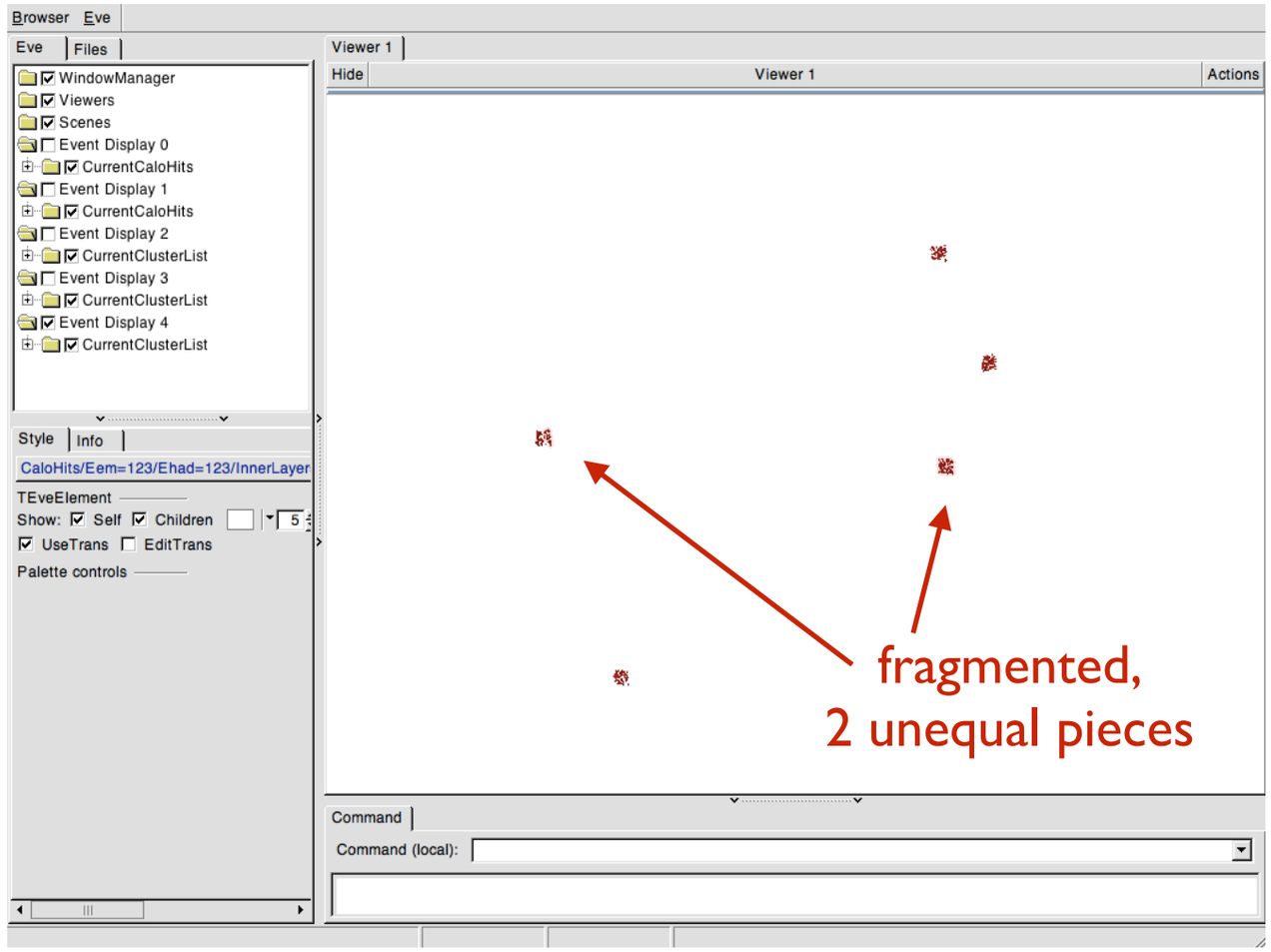


Pandora Fragment Clusters

```
<algorithm type="FragmentClustersExample">
    <NClustersToFragment>1</NClustersToFragment>
    <NFragmentsPerCluster>2</NFragmentsPerCluster>
</algorithm>
<algorithm type="DisplayListsExample">
    <DisplayCurrentClusters>true</DisplayCurrentClusters>
</algorithm>
```

```
---5 clusters in current list
------Cluster 0x7fd13b6b98f0, nHits: 90
------Cluster 0x7fd13b6bf030, nHits: 89
------Cluster 0x7fd13b6c0cb0, nHits: 94
------Cluster 0x7fd13e09dbb0, nHits: 60
------Cluster 0x7fd13e09dec0, nHits: 123
```

- This Algorithm performs a specified number of Cluster fragmentation operations.
- Each fragmentation takes a Cluster from the current Cluster list and splits its hits up into a specified number of new fragment Clusters.
- The fragmentation mechanism allows the original and new fragment Clusters to exist concurrently.
- The Algorithm makes a decision as to which configuration is best and the memory is all tidied accordingly.

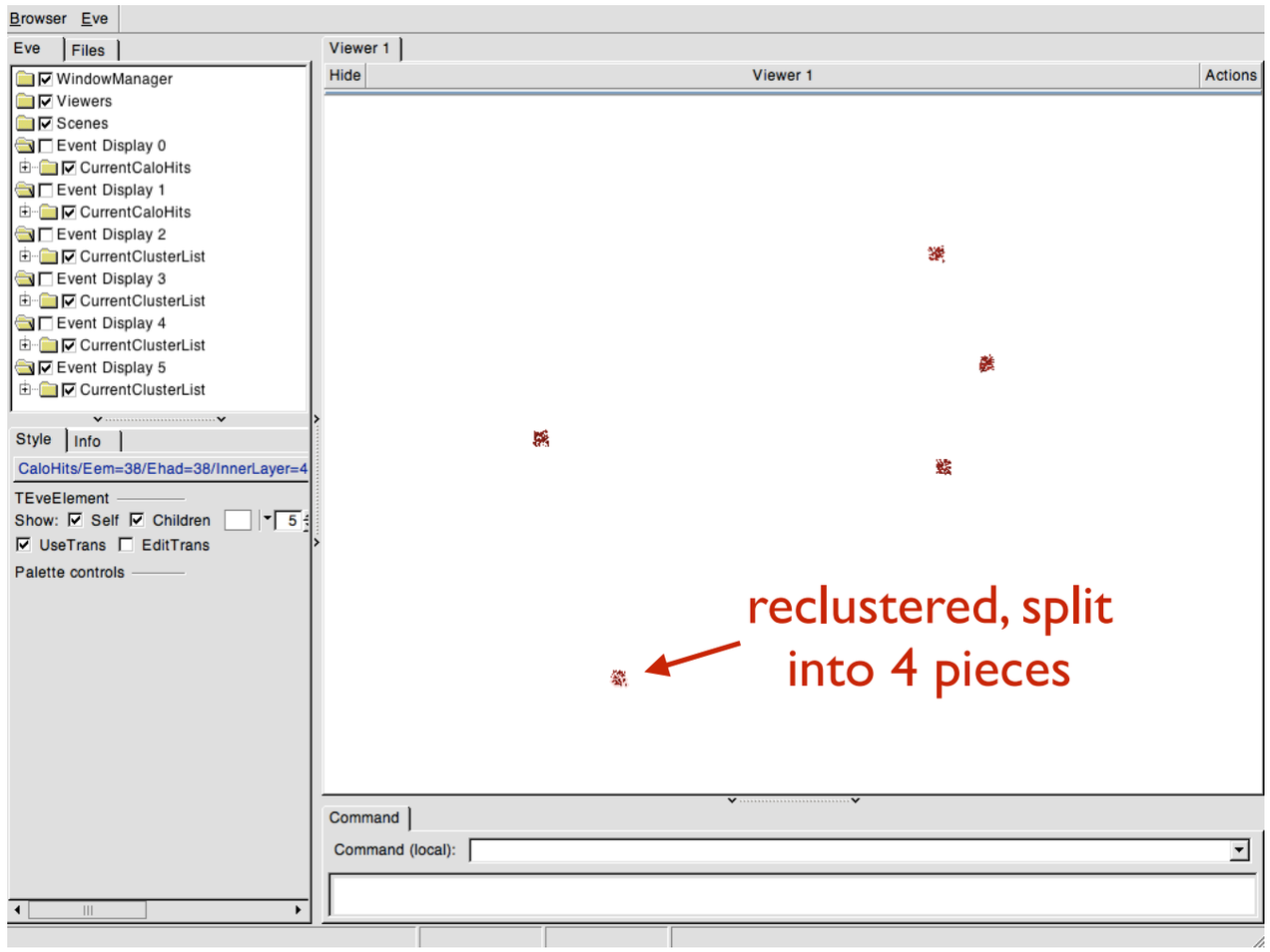


Pandora Reconfigure Clusters

```
<algorithm type="ReconfigureClustersExample">
   <NClustersToReconfigure>1</NClustersToReconfigure>
   <ReclusteringAlgorithms>
      <algorithm type="CreateClustersDaughterExample">
         <NClustersToMake>3</NClustersToMake>
       </algorithm>
      <algorithm type="CreateClustersDaughterExample">
         <NClustersToMake>4</NClustersToMake>
      </algorithm>
   </ReclusteringAlgorithms>
</algorithm>
<algorithm type="DisplayListsExample">
   <DisplayCurrentClusters>true</DisplayCurrentClusters>
</algorithm>
```

- Similar to the fragmentation example, but the new Cluster formation is performed by daughter clustering Algorithms.
- The parent Algorithm can collect N different Cluster configurations for the CaloHits in an original Cluster.
- The Algorithm can compare these many configurations, then simply select the "best", with all memory being tidied accordingly.

```
---8 clusters in current list
------Cluster 0x7fd13b6bf030, nHits: 89
------Cluster 0x7fd13b6c0cb0, nHits: 94
------Cluster 0x7fd13e09dbb0, nHits: 60
------Cluster 0x7fd13e09dec0, nHits: 123
------Cluster 0x7fd13e142140, nHits: 11
------Cluster 0x7fd13e1a51f0, nHits: 38
------Cluster 0x7fd13e1a5500, nHits: 21
------Cluster 0x7fd13e1a5810, nHits: 20
```

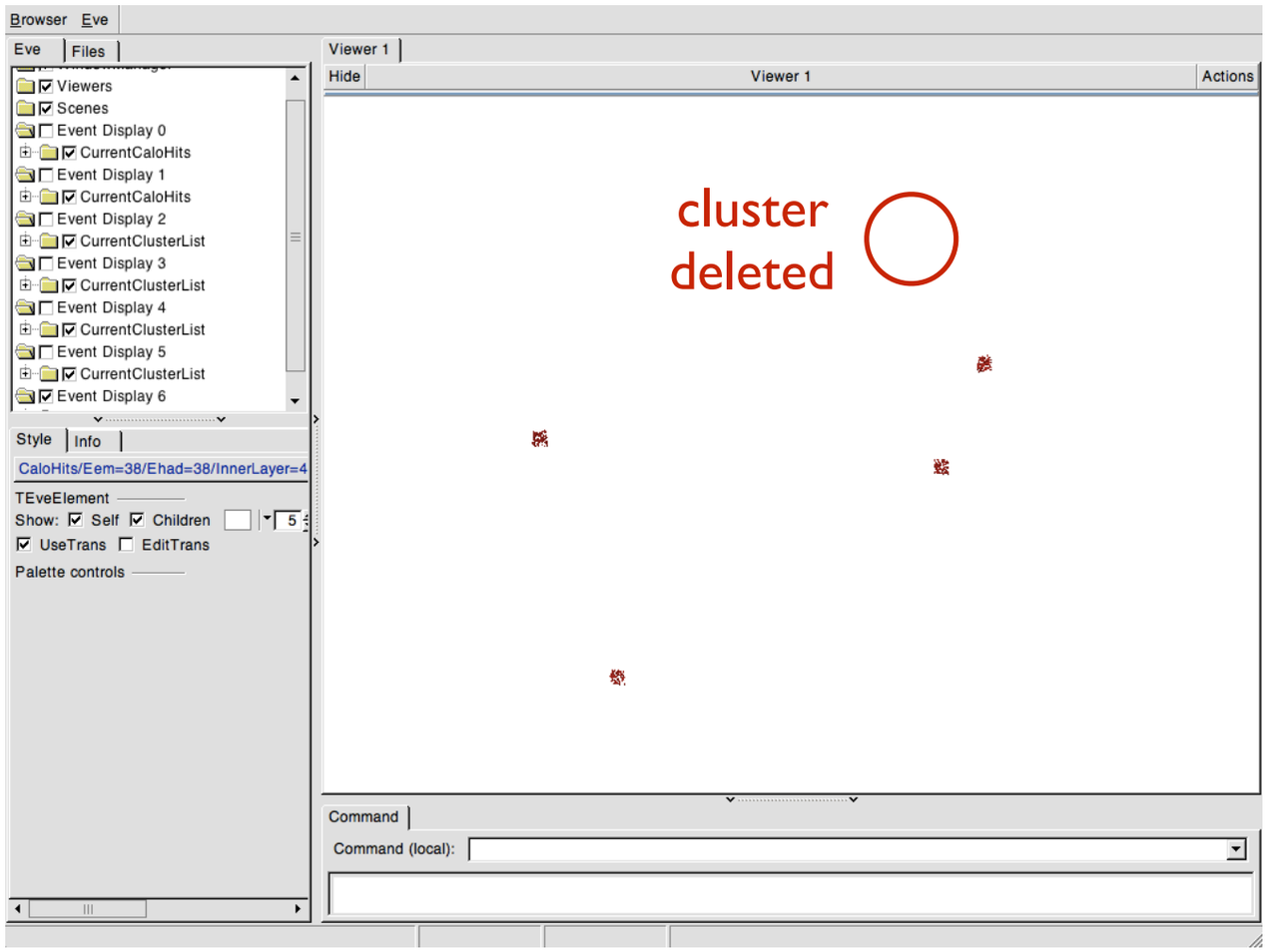


Pandora Delete Clusters

```
<algorithm type="DeleteClustersExample">
    <NClustersToDelete>1</NClustersToDelete>
</algorithm>
<algorithm type="DisplayListsExample">
    <DisplayCurrentClusters>true</DisplayCurrentClusters>
</algorithm>
```

```
---7 clusters in current list
------Cluster 0x7fd13b6c0cb0, nHits: 94
------Cluster 0x7fd13e09dbb0, nHits: 60
------Cluster 0x7fd13e09dec0, nHits: 123
------Cluster 0x7fd13e142140, nHits: 11
------Cluster 0x7fd13e1a51f0, nHits: 38
------Cluster 0x7fd13e1a5500, nHits: 21
------Cluster 0x7fd13e1a5810, nHits: 20
```

- This Algorithm simply deletes (up to) the specified number of Clusters from the current Cluster list (note: list is unordered by default).
- As with any Cluster, Vertex or Pfo deletion, care must be taken in-Algorithm with regard to the possibility of dangling pointers, etc.
- Once a Cluster has been deleted, the constituent CaloHits are flagged as available.

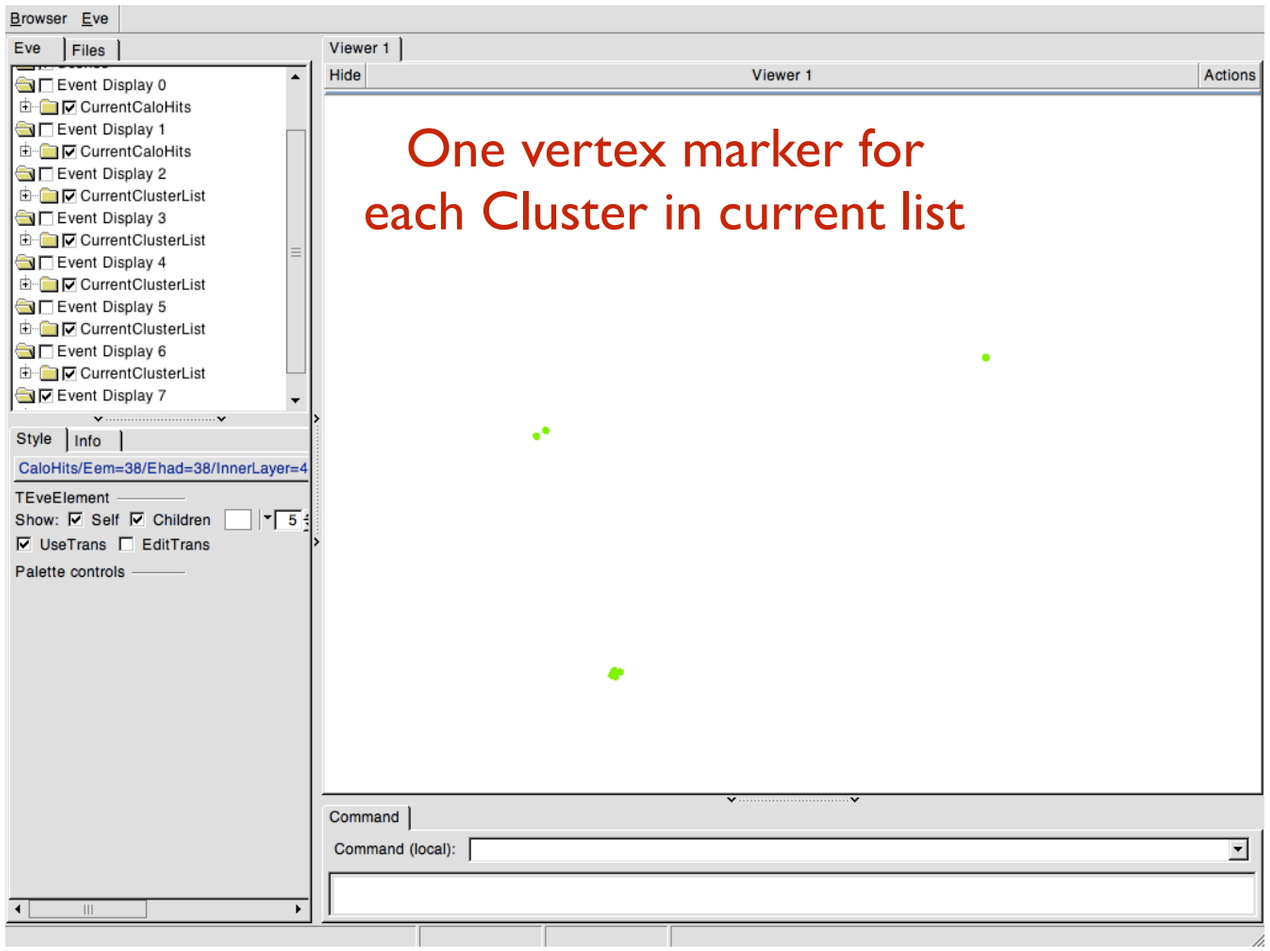


Pandora Create Vertices

```
<algorithm type="CreateVerticesExample">
    <OutputListName>ExampleVertices</OutputListName>
</algorithm>
<algorithm type="DisplayListsExample">
    <DisplayCurrentVertices>true</DisplayCurrentVertices>
</algorithm>
```

```
---7 vertices in current list
```

- This Algorithm creates Vertices corresponding to the inner layer centroid of each of the Clusters in the current Cluster list.
- The Vertex list management is performed solely in the algorithm, with all procedures similar to those required for Cluster creation.

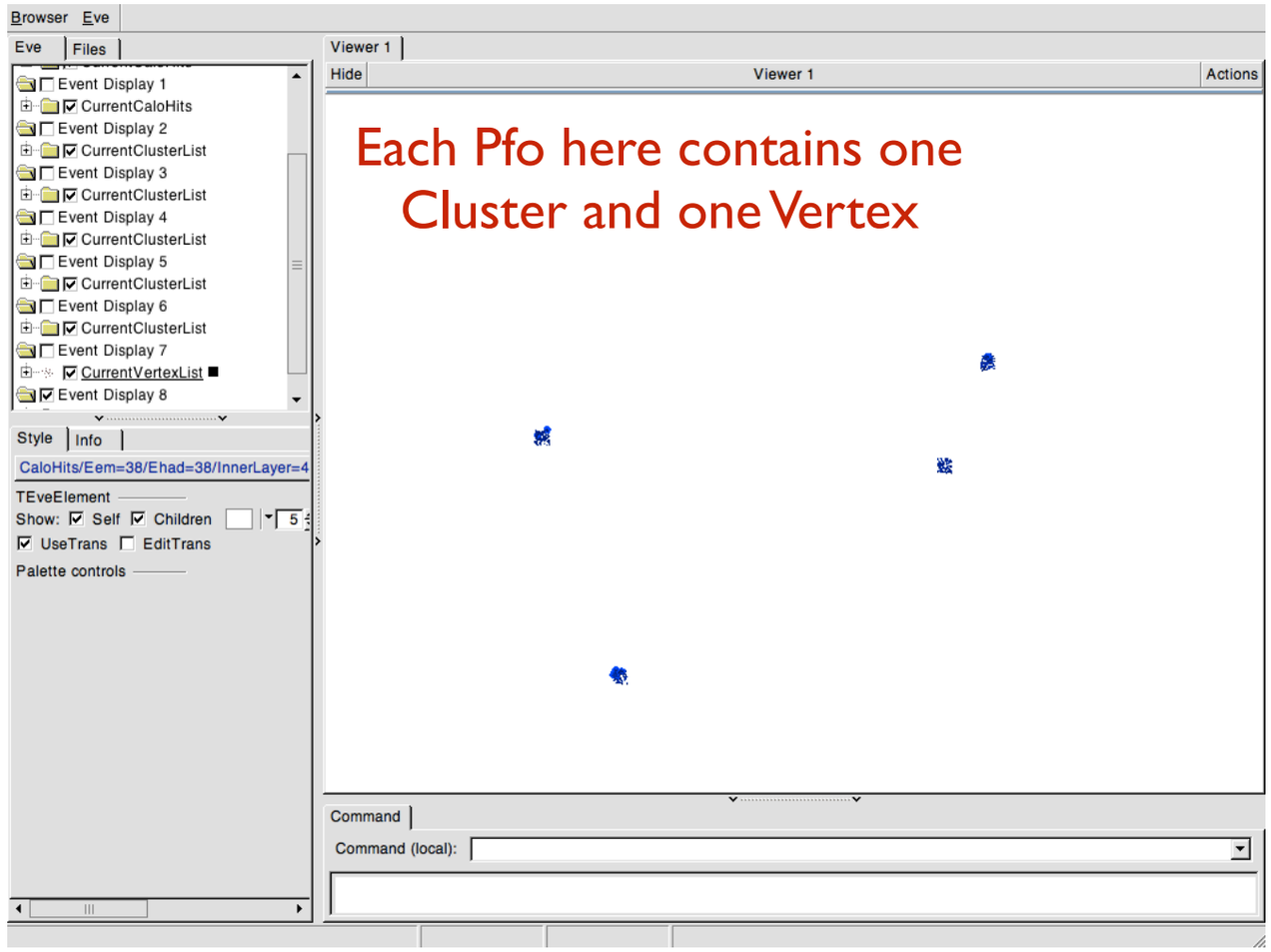


Pandora Create Pfos

```
<algorithm type="CreatePfosExample">
    <OutputListName>ExamplePfos</OutputListName>
</algorithm>
<algorithm type="DisplayListsExample">
    <DisplayCurrentPfos>true</DisplayCurrentPfos>
</algorithm>
```

```
---7 pfos in current list
------Pfo 0x7fd13e137e70, nClusters: 1, nVertices: 1
------Pfo 0x7fd13e13bbe0, nClusters: 1, nVertices: 1
------Pfo 0x7fd13e13bdd0, nClusters: 1, nVertices: 1
------Pfo 0x7fd13e17d730, nClusters: 1, nVertices: 1
------Pfo 0x7fd13e17ec60, nClusters: 1, nVertices: 1
------Pfo 0x7fd13e193580, nClusters: 1, nVertices: 1
------Pfo 0x7fd13e193ba0, nClusters: 1, nVertices: 1
```

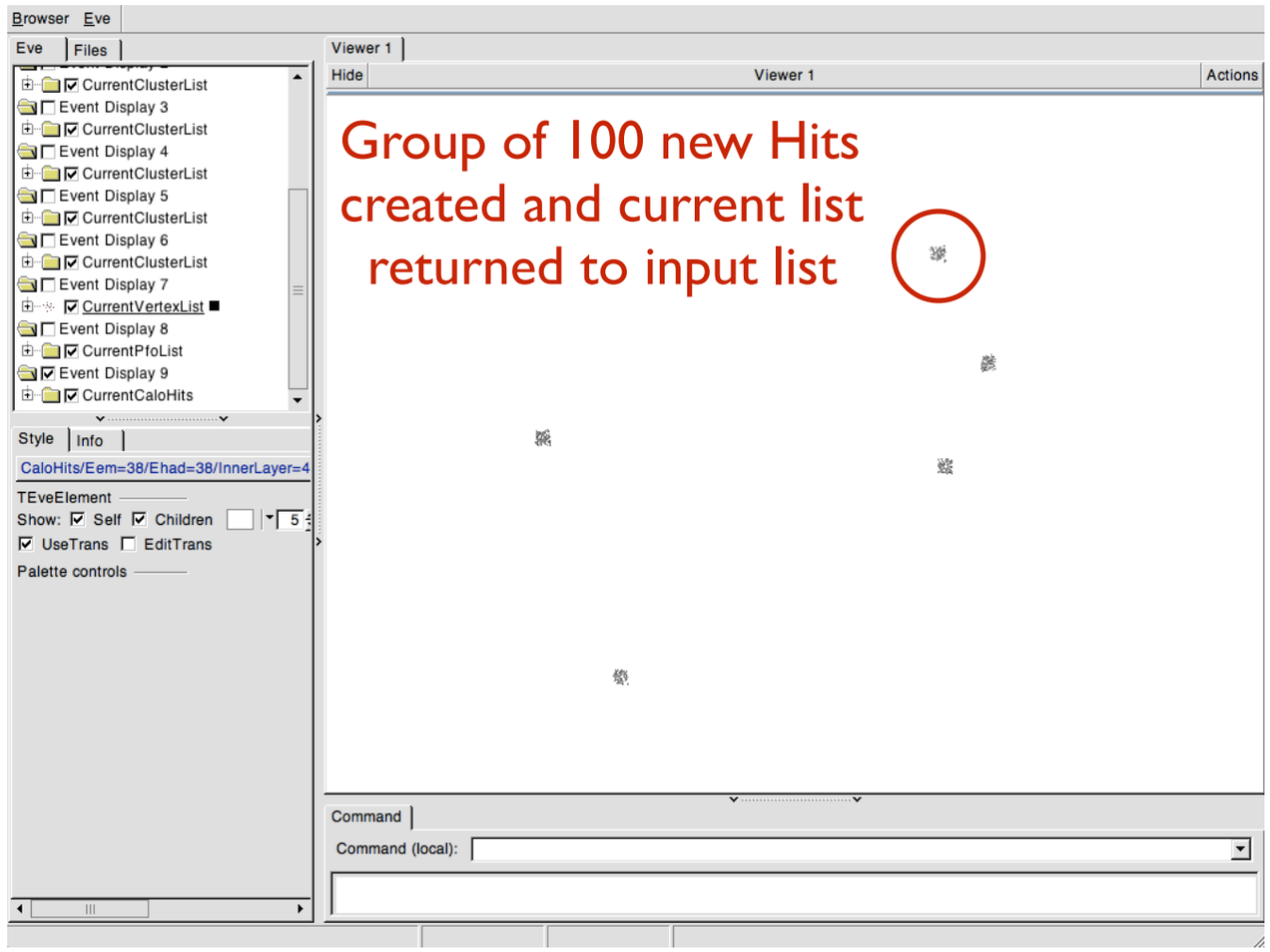
- This Algorithm creates one Pfo for each of the Clusters in the current Cluster list.
- The Algorithm also looks to find the closest Vertex to each Cluster and add it to the Pfo.
- The Pfo energy mass, etc. metadata all takes dummy values.
- The list management is performed solely in the Algorithm, with all procedures similar to those required for Cluster/Vertex creation.



Pandora Create Additional Hits

```
<algorithm type="CreateAdditionalCaloHitsExample">
    <NCaloHitsToMake>100</NCaloHitsToMake>
    <SetCurrentListToInputList>true</SetCurrentListToInputList>
</algorithm>
<algorithm type="DisplayListsExample">
    <DisplayCurrentCaloHits>true</DisplayCurrentCaloHits>
</algorithm>
```

- This Algorithm creates an additional group of a specified number of ExampleCaloHits (demonstrating an extension of the default CaloHits, with an additional string property).
- The newly created CaloHits appear in a special "Input" list, owned by the Pandora CaloHitManager.
- The Algorithm can set the current list back to the Input list, switching away from our earlier list of (randomly) selected CaloHits.



Pandora Use Algorithm Tool

```
<algorithm type="UseAlgorithmToolExample">
    <ExampleTools>
        <tool type="AlgorithmToolExample"/>
    </ExampleTools>
</algorithm>
```

- This demonstrates the mechanics of creating/running AlgorithmTools, with configurable interfaces.
- AlgorithmTools inheriting from the IExampleAlgorithmTool interface class (defined in the UseAlgorithmToolAlgorithm header file) can be registered via the Pandora Client Application.
- These AlgorithmTools can then be instantiated when parsing the xml settings of a parent algorithm, using e.g. the XmlHelper function ProcessAlgorithmTool(List).
- The AlgorithmTools are owned by the Pandora AlgorithmManager, but the parent Algorithm can store the addresses of the Tool instances and use the functionality that the Tools provide.
- AlgorithmTools thus allow a user to drop-in, via xml, multiple methods of querying/processing a data structure. This will typically be a large/expensive object created by the Algorithm.
- The parent Algorithm can fully define the interface for its AlgorithmTools (unlike using daughter Algorithms) and there is no change in the list management when an Algorithm uses its Tools.

Pandora Use Plugins

```
<!-- PLUGIN SETTINGS -->
<MuonPlugin>ParticleIdExample</MuonPlugin>
<HadronicEnergyCorrectionPlugins>EnergyCorrectionExample</HadronicEnergyCorrectionPlugins>

<ParticleIdExample>
    <ExampleParameter>1</ExampleParameter>
</ParticleIdExample>

<EnergyCorrectionExample>
    <EnergyMultiplier>1.1</EnergyMultiplier>
</EnergyCorrectionExample>

<!-- ALGORITHM SETTINGS —>
<algorithm type="UsePluginsExample"/>
```

- This Algorithm demonstrates the mechanics of using Pandora Plugins objects, which are created by the Client Application and registered with the Pandora PluginManager.
- The interfaces for the Plugins are defined by Pandora, but the implementation can be provided freely, allowing for very use-case-specific solutions, maybe with external dependencies.
- Plugins can be provided for cluster energy corrections, particle identification, BField maps and the division of space in the detector into “pseudo-layers”.
- The ExampleContent library provides a simple pseudo-layer calculator (binning of Z coordinates), a simple scaling-factor energy correction and a dummy particle id assigned to the muon id role.

Pandora Write Tree

```
<algorithm type="WriteTreeExample">
    <FileName>ExampleFile.root</FileName>
    <TreeName>ExampleTree</TreeName>
</algorithm>
```

- This Algorithm uses Pandora Monitoring in order to create a simple ROOT TTree.
- The tree contains, for each event:
  - An integer, recording the number of clusters in the current list.
  - A vector of floats, recording the hadronic energy estimator for each cluster.
- Values are assigned to local variables each event, then FillTree is called.
- The tree is written via the Algorithm destructor, called when the parent Pandora instance is deleted.

Example tree contents:

```
bash-3.2$ root -l ExampleFile.root
root [0]
Attaching file ExampleFile.root as _file0...
root [1] .ls
TFile**                 ExampleFile.root
 TFile*                 ExampleFile.root
  KEY: TTree    ExampleTree;1   ExampleTree
root [2] ExampleTree->Scan()
************************************************
*    Row   * Instance * nClusters * clusterEn *
************************************************
*        0 *        0 *         7 *         94 *
*        0 *        1 *         7 *         60 *
*        0 *        2 *         7 *        123 *
*        0 *        3 *         7 *         11 *
*        0 *        4 *         7 *         38 *
*        0 *        5 *         7 *         21 *
*        0 *        6 *         7 *         20 *
************************************************
```

Additional Information

http://arxiv.org/abs/1506.05348

The Pandora Software Development Kit for Pattern Recognition

J. S. Marshall[a,*], M. A. Thomson[a]

[a] *Cavendish Laboratory, University of Cambridge, Cambridge, United Kingdom*

Abstract

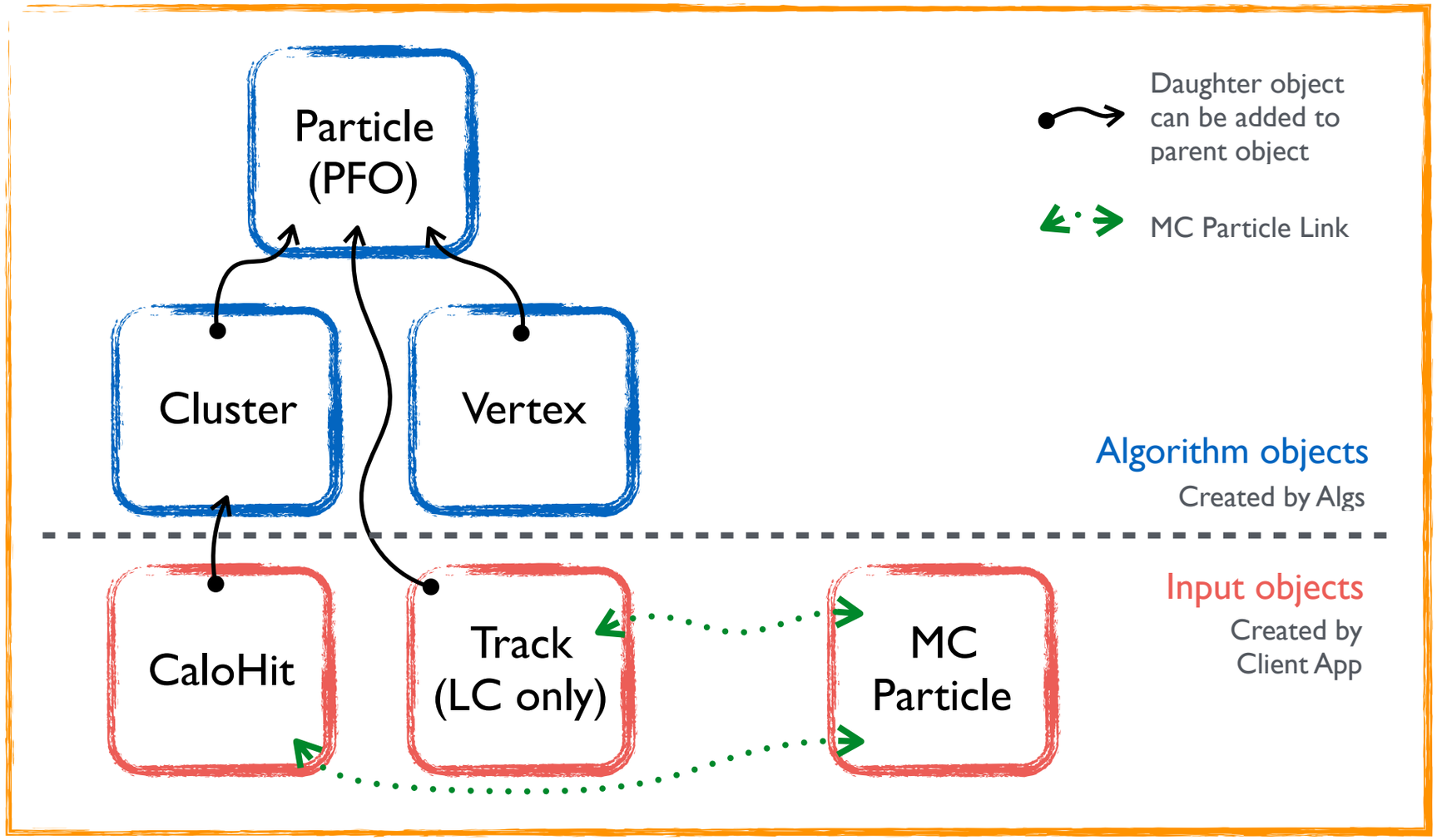
The development of automated solutions to pattern recognition problems is important in many areas of scientific research and human endeavour. This paper describes the implementation of the Pandora Software Development Kit, which aids the process of designing, implementing and running pattern recognition algorithms. The Pandora Application Programming Interfaces ensure simple specification of the building-blocks defining a pattern recognition problem. The logic required to solve the problem is implemented in algorithms. The algorithms request operations to create or modify data structures and the operations are performed by the Pandora framework. This design promotes an approach using many decoupled algorithms, each addressing specific topologies. Details of algorithms addressing two pattern recognition problems in High Energy Physics are presented: reconstruction of events at a high-energy e^+e^- linear collider and reconstruction of cosmic ray or neutrino events in a liquid argon time projection chamber.

Keywords: Software Development Kit, Pattern recognition, High Energy Physics

Pandora SDK

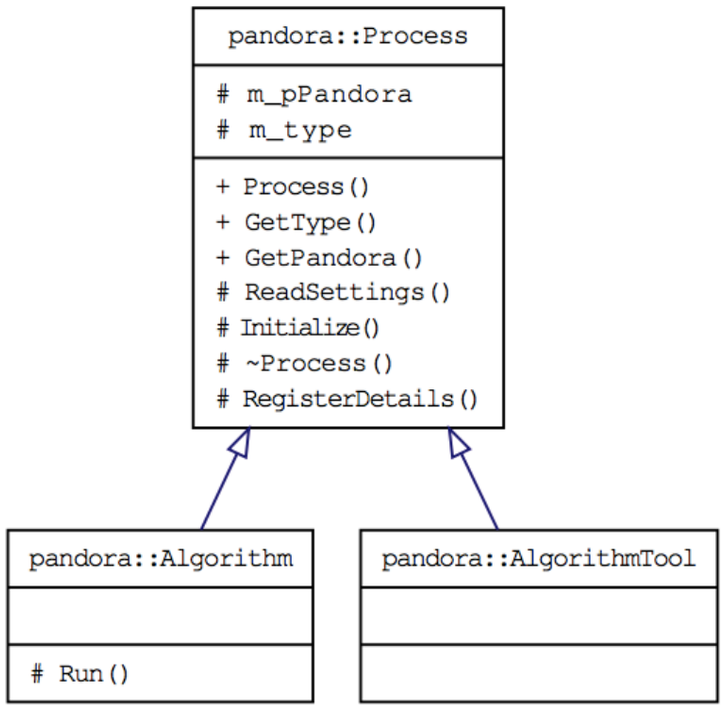
- The Pandora SDK provides a comprehensive **Event Data Model** (EDM) for managing pattern recognition problems. Instances of objects in the EDM are owned by **Pandora Managers**.
- The object instances are stored in named lists and the Managers are able to create new objects, delete objects, create and save new lists and move objects between lists.
- The Managers provide a complete set of low-level operations that allow the high-level operations requested by pattern recognition algorithms to be satisfied.

pandora::Pandora
- m_pAlgorithmManager - m_pCaloHitManager - m_pClusterManager - m_pGeometryManager - m_pMCManager - m_pPfoManager - m_pPluginManager - m_pTrackManager - m_pVertexManager - m_pPandoraSettings - m_pPandoraApiImpl - m_pPandoraContentApiImpl - m_pPandoraImpl
+ Pandora() + ~Pandora() + GetPandoraApiImpl() + GetPandoraContentApiImpl() + GetSettings() + GetGeometry() + GetPlugins() - PrepareEvent() - ProcessEvent() - ResetEvent() - ReadSettings()



Pandora Algorithms

- Pandora algorithms contain the step-by-step instructions for finding patterns in the provided data.
- They use the APIs to access objects and to request the Managers to make new objects or modify existing objects.
- They inherit from the **Process** class, which provides functionality for handshaking with Pandora, XML config and function callbacks.



Algorithm 1 Cluster creation pseudocode. The logic determining when to create new Clusters and when to extend existing Clusters will vary between algorithms.

```
procedure CLUSTER CREATION
    Create temporary Cluster list
    Get current CaloHit list
    for all CaloHits do
        if CaloHit available then
            for all newly-created Clusters do
                Find best host Cluster
            if Suitable host Cluster found then
                Add CaloHit to host Cluster
            else
                Add CaloHit to a new Cluster
    Save new Clusters in a named list
```

Algorithm 2 Cluster merging pseudocode. The logic governing the identification of suitable parent Clusters and daughter Clusters will vary between algorithms.

```
procedure CLUSTER MERGING
    Get current Cluster list
    for all Clusters do
        if Cluster is suitable parent then
            for all Clusters do
                Find best daughter Cluster
            if Suitable daughter Cluster found then
                Merge daughter Cluster into Parent
```

Pandora Client App

- To use the Pandora SDK, a user must create a Pandora client application. This provides the input building-blocks to describe the pattern recognition problem and receives the final output.
- The client application is responsible for controlling the pattern recognition reconstruction. It creates the Pandora instance(s) and uses Pandora APIs to send requests to these instances.

- E.g. For LArSoft, the Pandora client application is **LArPandoraInterface**
- All Algorithms and Plugins are built as part of the LArContent library.
- Pandora Hits (and, optionally, MCParticles), are extracted from named art producers.
- The output consists of collections of PFParticles, Clusters, Tracks, Showers, Vertices

Algorithm 3 Pseudocode description of a client application for LAr TPC event reconstruction in a single drift volume

```
1:  procedure MAIN
2:      Create a Pandora instance
3:      Register Algorithms and Plugins
4:      Ask Pandora to parse XML settings file
5:      for all Events do
6:          Create CaloHit instances
7:          Create MCParticle instances
8:          Specify MCParticle-CaloHit relationships
9:          Ask Pandora to process the event
10:         Get output PFOs and write to file
11:         Reset Pandora before next event
```