RecoilUtils

A ROOT based Analysis Framework for the HERMES Recoil Detector

Andreas Mussgiller

II. Physikalisches Institut
Friedrich-Alexander-Universität
Erlangen-Nürnberg

hermes

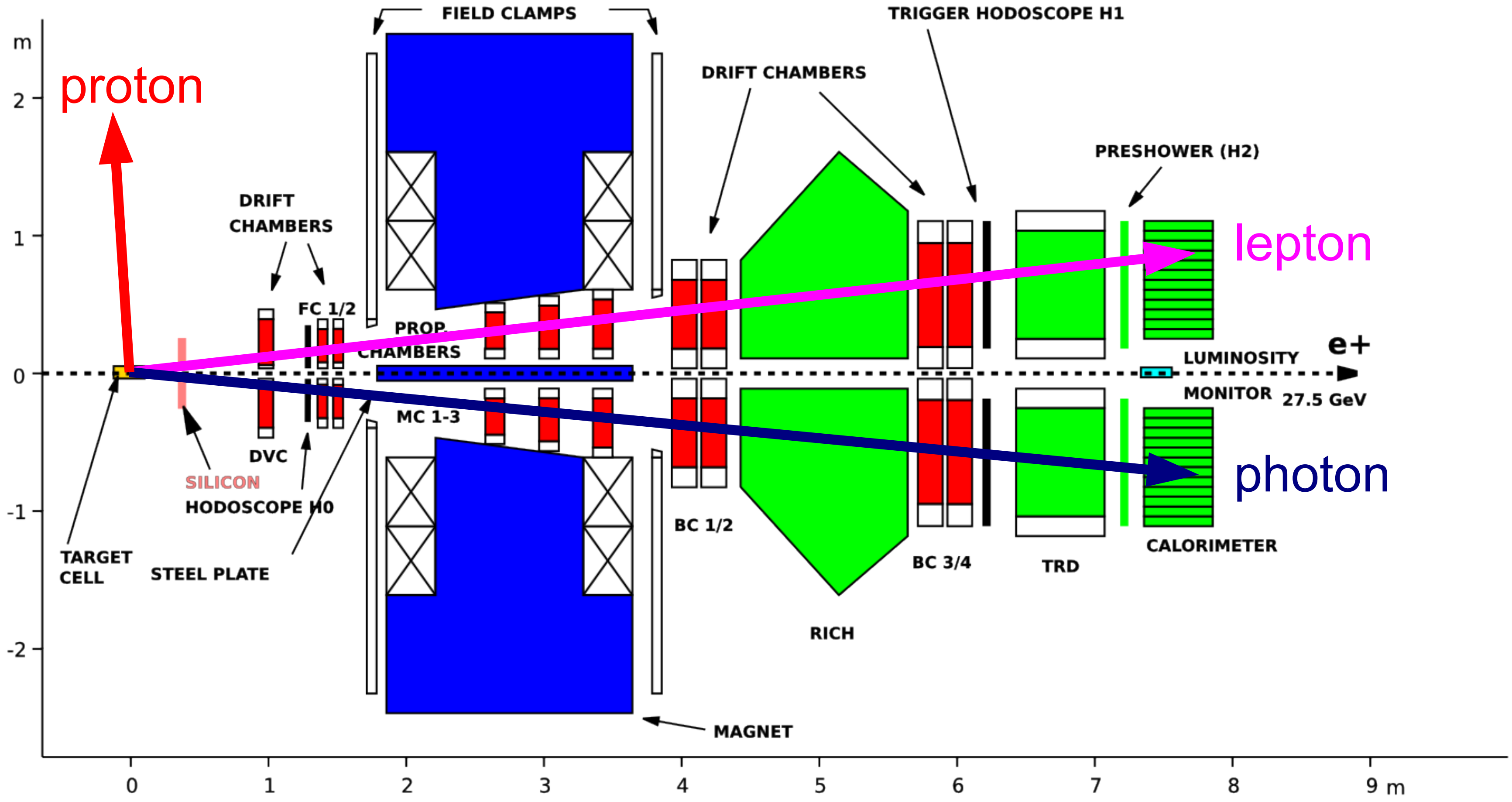
bmb+f - Förderschwerpunkt
HERMES
Großgeräte der physikalischen
Grundlagenforschung

DESY IT Seminar, 26/11/07

Outline

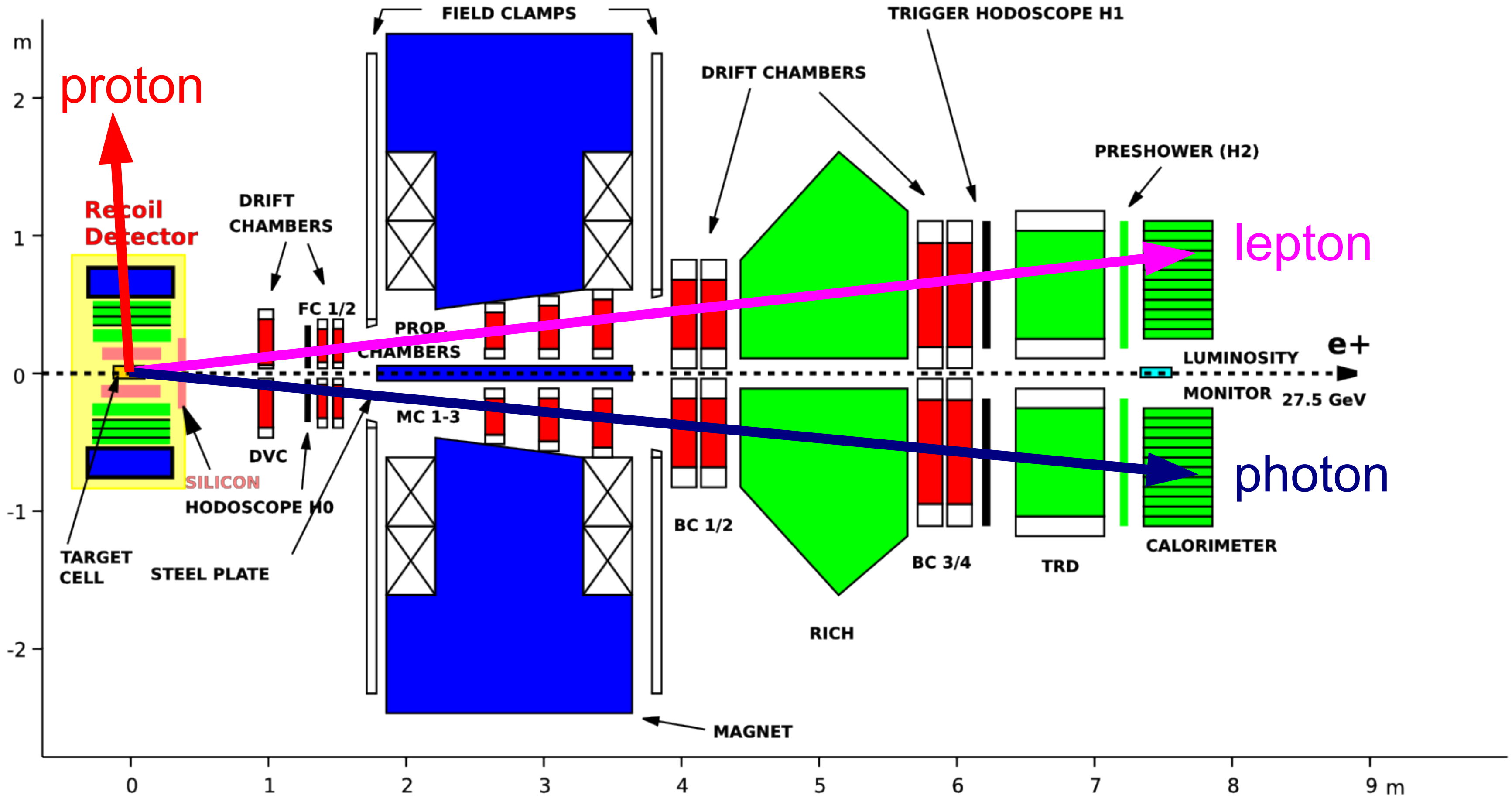
- The HERMES Recoil Detector
- General Idea
- A brief Project Overview
- Selected Features
  - uDST Framework
  - Cut Management
  - Event Display
- Summary and Future Plans

DVCS at HERMES



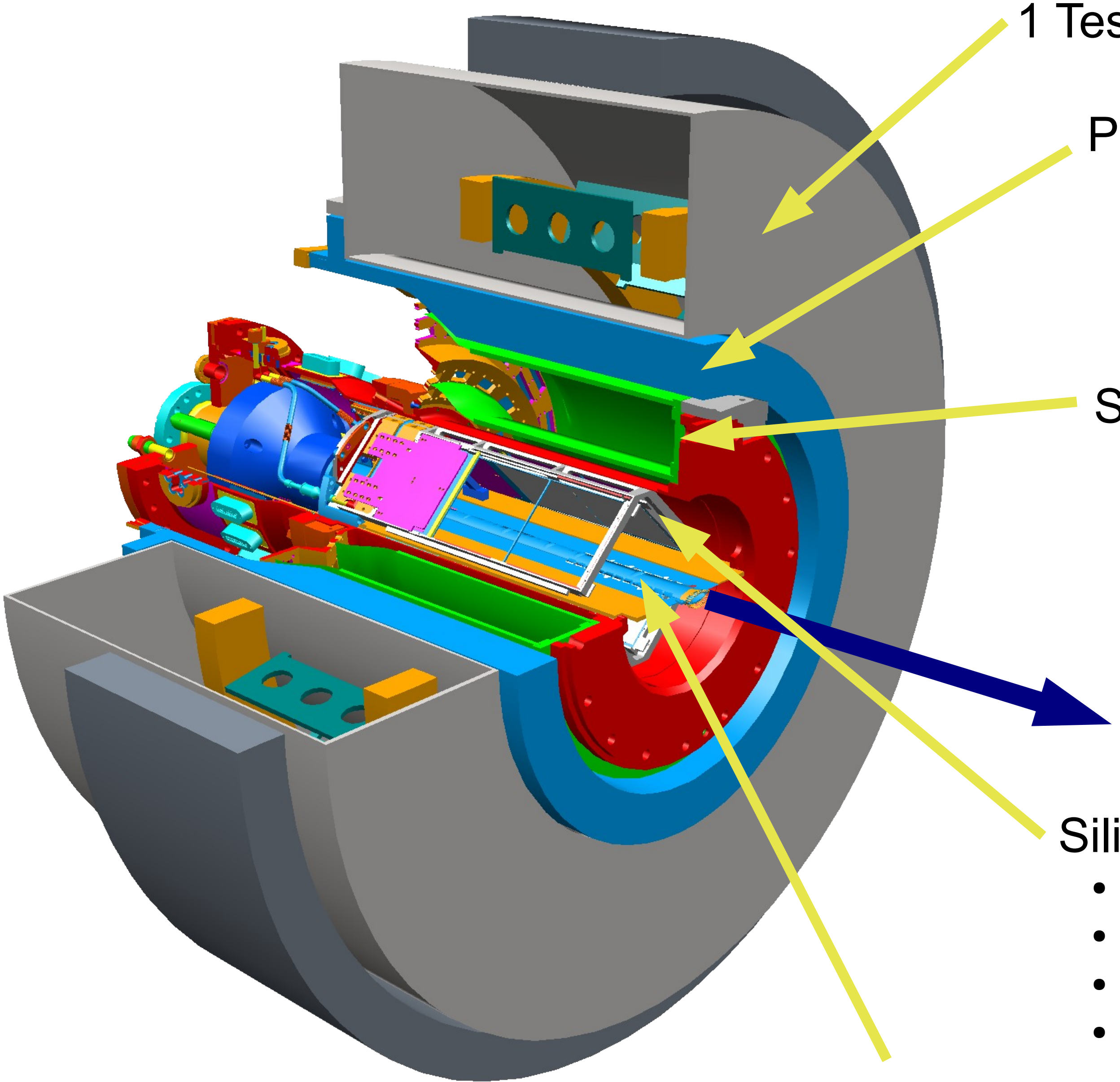
- Deeply virtual Compton scattering
- Recoiling proton undetected
- Process identified via missing mass
- About 15% background

DVCS at HERMES with Recoil Detector



- Recoiling proton detected
- Reduces background to about 1%

The HERMES Recoil Detector



1 Tesla Superconducting Solenoid

Photon Detector
- 3 Layers of Tungsten/Scintillator
- PID for p > 0.5 GeV/c

Scintillating Fiber Detector
- 2 Barrels
- 2 Parallel and 2 Stereo Layers in each Barrel
- Momentum Reconstruction & PID

Beam

Silicon Detector
- 2 Layers inside Vacuum
- 16 double-sided Sensors
- 128 x 128 strips
- Momentum Reconstruction & PID

Target-Cell

Motivation

- Installed December 2005
- Start of Data taking February 2006
- Commissioning of detector components
  - Fiber tracker finished in February 2006
  - Silicon detector finished in September 2006

⬇

Understand the detectors:

- Noise
- Common mode
- Correlations
- ...
- Lots of plots to produce
- Many ideas

⬇

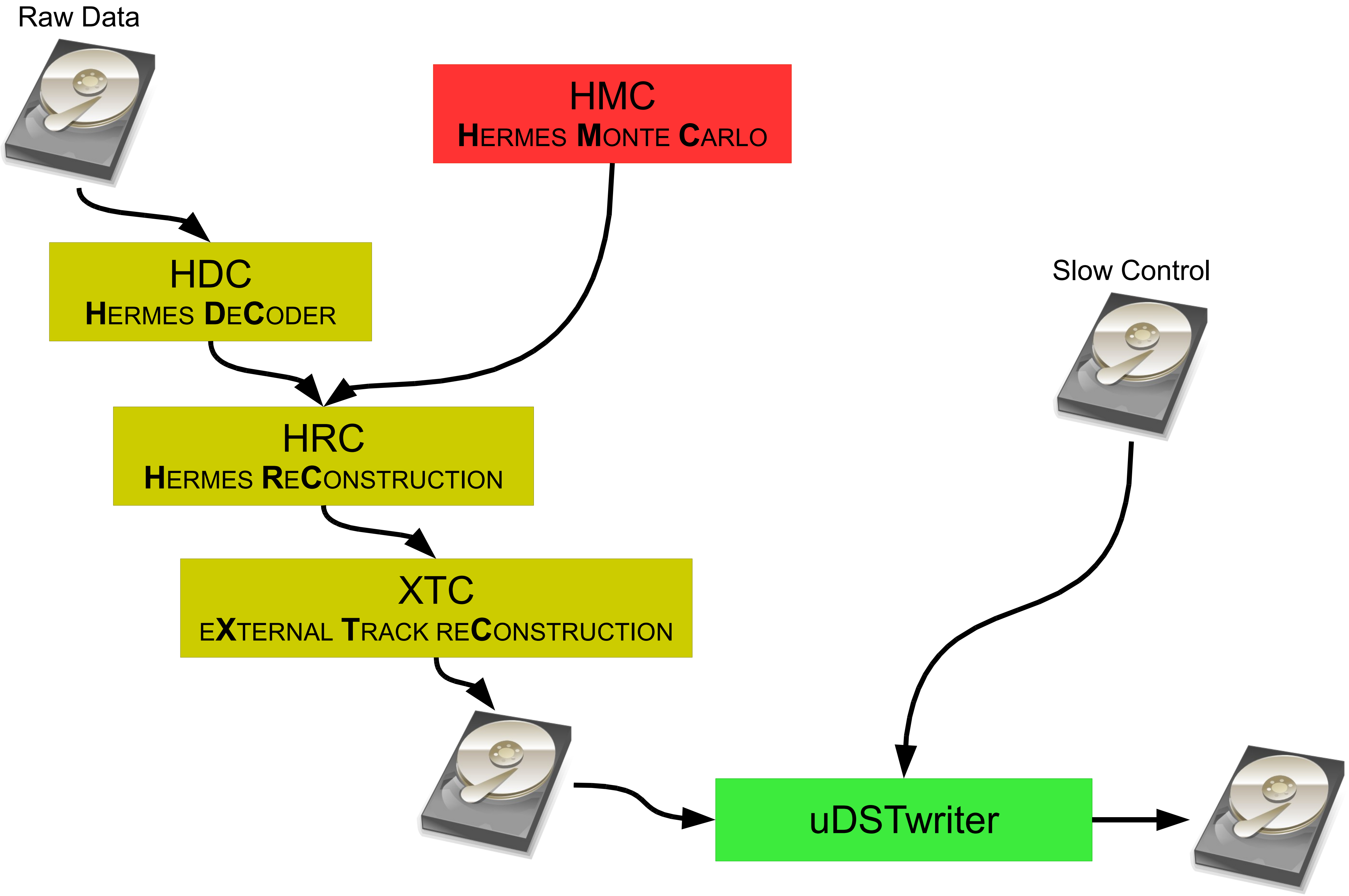
A flexible and modular analysis framework

- that uses many nice ROOT features
- not too complicated

Outline

- The HERMES Recoil Detector
- General Idea
- A brief Project Overview
- Selected Features
  - uDST Framework
  - Cut Management
  - Event Display
- Summary and Future Plans

HERMES Data Production Chain



Raw Data

HMC
HERMES MONTE CARLO

HDC
HERMES DECODER

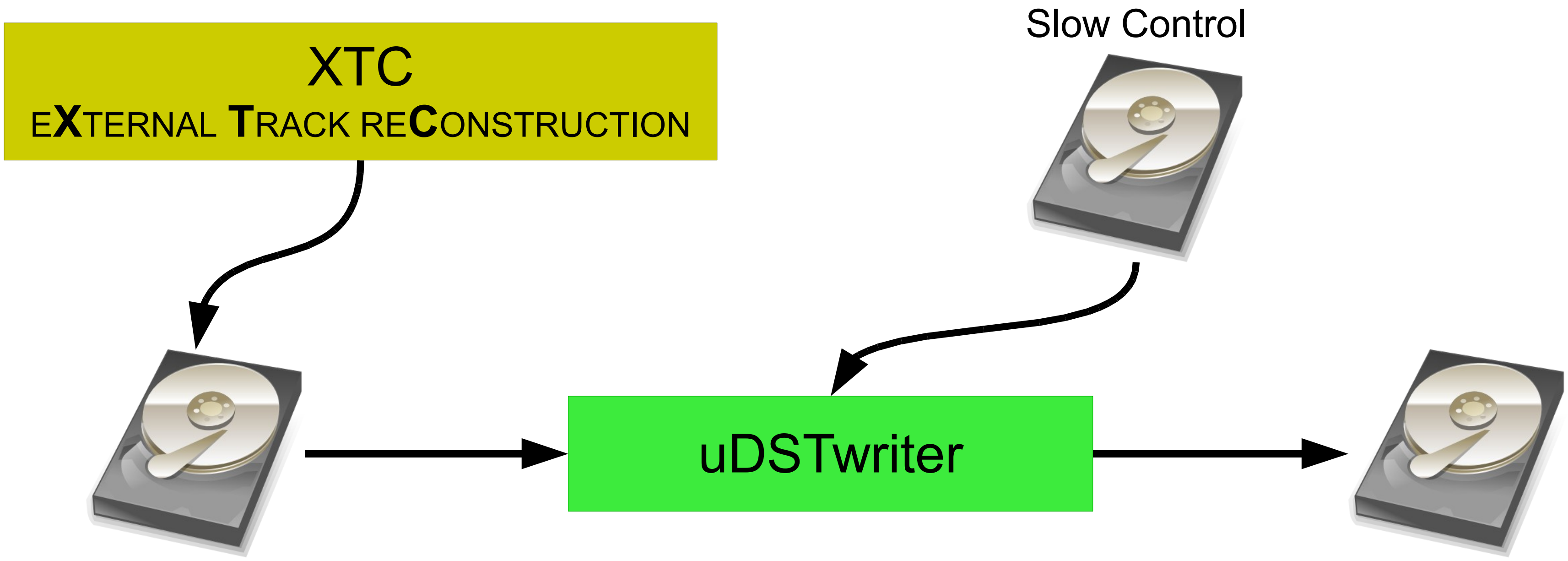
HRC
HERMES RECONSTRUCTION

XTC
EXTERNAL TRACK RECONSTRUCTION

Slow Control

uDSTwriter

HERMES Data Production Chain



XTC

E**X**TERNAL **T**RACK RE**C**ONSTRUCTION

Slow Control

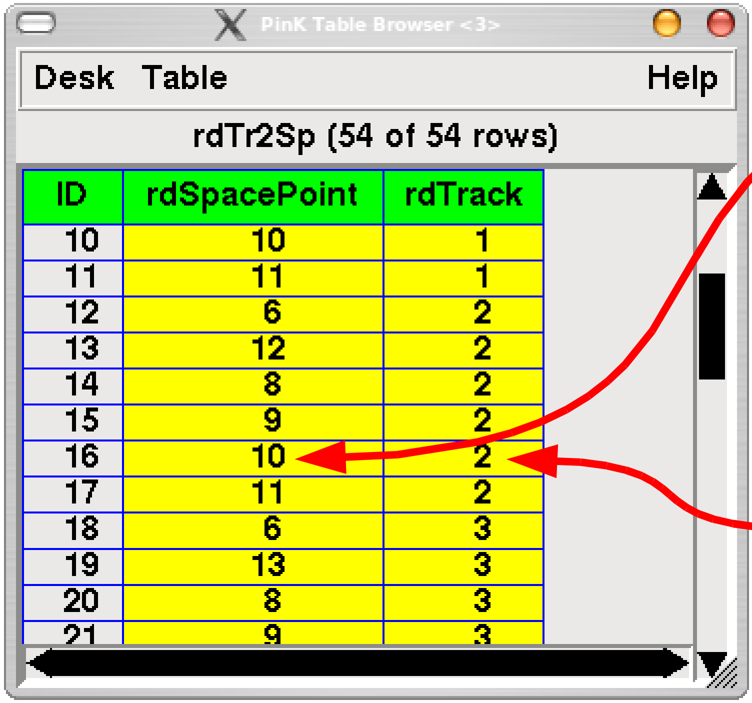
uDSTwriter

- Stored in ADAMO format
- Contains
  - Raw Data
  - Clusters
  - Spacepoints
  - Recoil Tracks
  - HERMES Tracks
- Used for detector studies

- Stored in ADAMO format
- Contains only reduced data
  - Clusters
  - Tracks
  - Slow Control
- Used for physics analysis

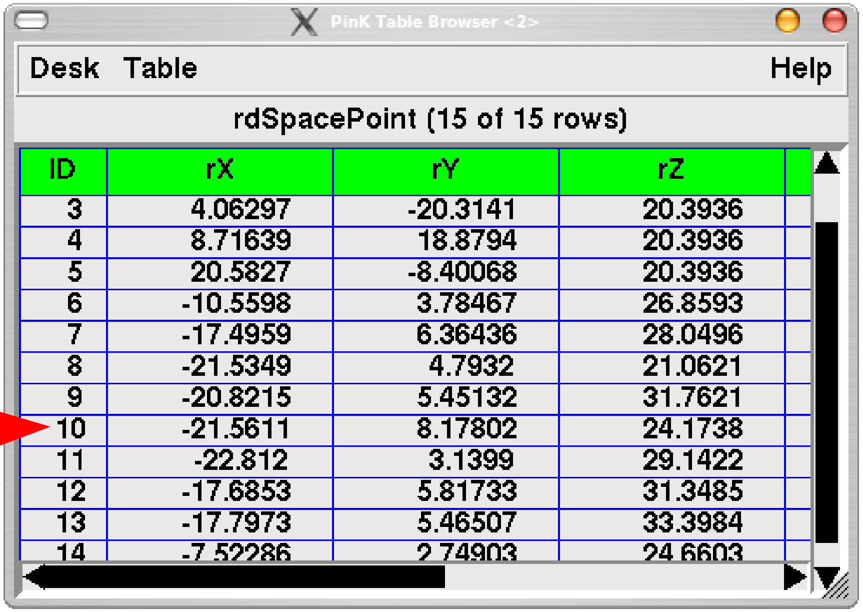
A Bit of ADAMO

- Based on Entity-Relationship Model
  - Entities with fixed number of attributes
  - Relationships between entities



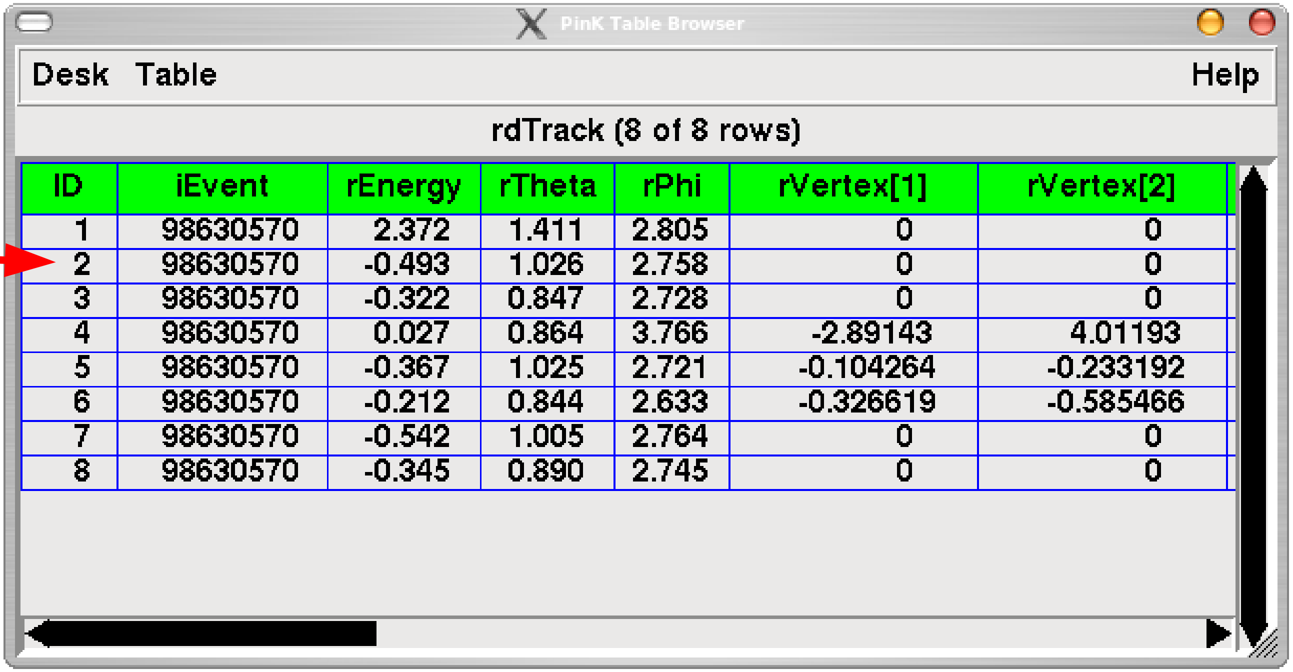
rdTr2Sp (54 of 54 rows)

ID	rdSpacePoint	rdTrack
10	10	1
11	11	1
12	6	2
13	12	2
14	8	2
15	9	2
16	10	2
17	11	2
18	6	3
19	13	3
20	8	3
21	9	3



rdSpacePoint (15 of 15 rows)

ID	rX	rY	rZ
3	4.06297	-20.3141	20.3936
4	8.71639	18.8794	20.3936
5	20.5827	-8.40068	20.3936
6	-10.5598	3.78467	26.8593
7	-17.4959	6.36436	28.0496
8	-21.5349	4.7932	21.0621
9	-20.8215	5.45132	31.7621
10	-21.5611	8.17802	24.1738
11	-22.812	3.1399	29.1422
12	-17.6853	5.81733	31.3485
13	-17.7973	5.46507	33.3984
14	-7.52286	2.74903	24.6603



rdTrack (8 of 8 rows)

ID	iEvent	rEnergy	rTheta	rPhi	rVertex[1]	rVertex[2]
1	98630570	2.372	1.411	2.805	0	0
2	98630570	-0.493	1.026	2.758	0	0
3	98630570	-0.322	0.847	2.728	0	0
4	98630570	0.027	0.864	3.766	-2.89143	4.01193
5	98630570	-0.367	1.025	2.721	-0.104264	-0.233192
6	98630570	-0.212	0.844	2.633	-0.326619	-0.585466
7	98630570	-0.542	1.005	2.764	0	0
8	98630570	-0.345	0.890	2.745	0	0

The General Idea

- Each ADAMO table represented by a class
  - rdTrack → TrdTrack
  - rdSpacePoint → TrdSpacePoint
- Relationships between tables handled by pointers, TRef and TRefArray
  - All ADAMO navigations done in common code (ADAMO to ROOT interface)
  - Let ROOT do the rest
- Analysis done via hierarchy of *Analysis Modules*
  - Common modules for standard tasks (raw histos etc)
  - A specific analysis module for each dedicated study
  - Modules can be used by everyone

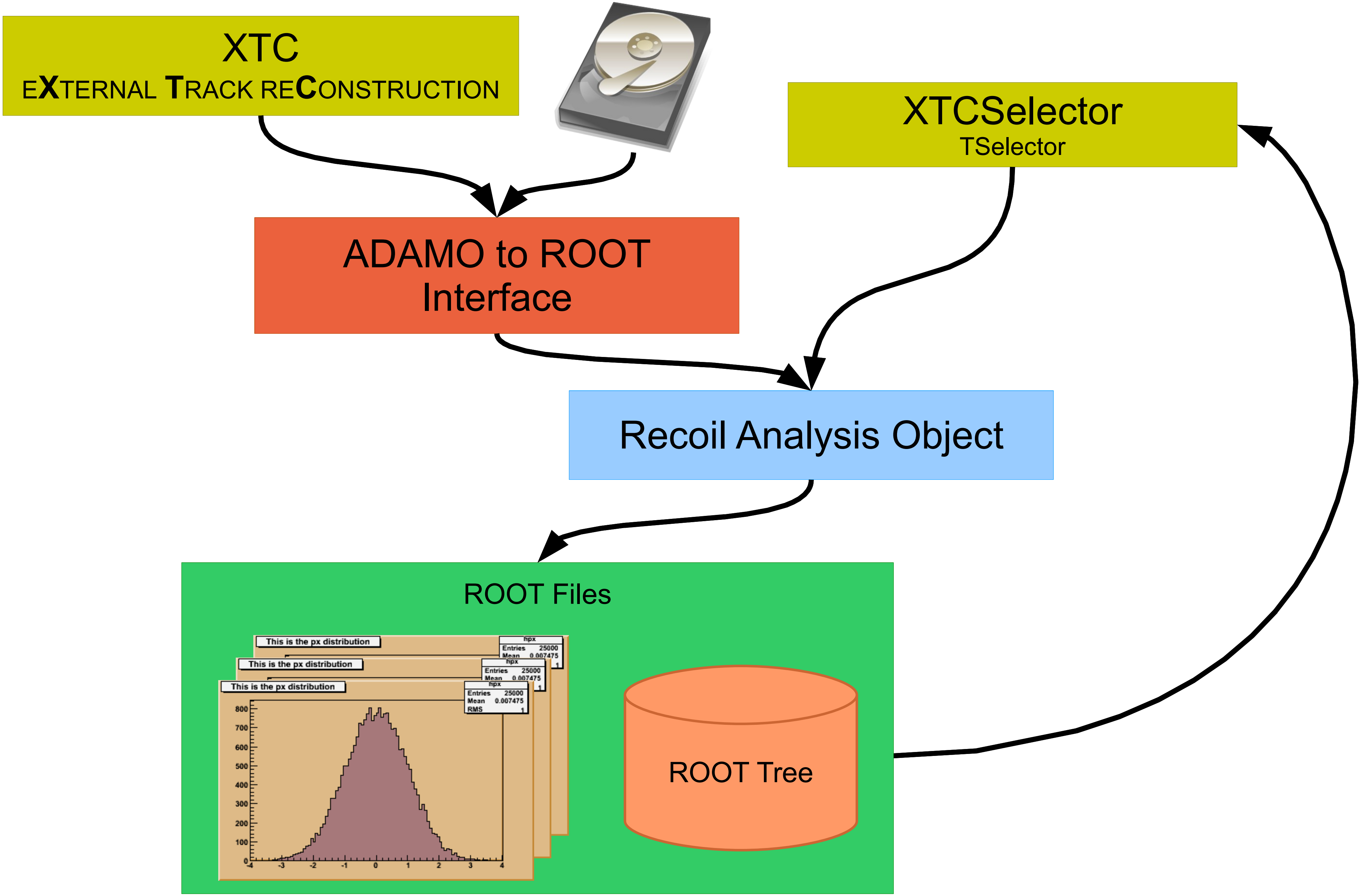
The General Idea

```
class TrdTrack : public TAdamoRow
{
public:
  TrdTrack();
  virtual ~TrdTrack();

  ...

  void             AddSpacePoint(TrdSpacePoint * hit);
  Int_t            GetNSpacePoints() const { return fNSpac
  TrdSpacePoint *  GetSpacePoint(Int_t idx) const;

  ...

  // Number of spacepoints used in this track
  Int_t                        fNSpacePoints;
  // Array of references to spacepoints
  TRefArray                    fSpacePoints;

  ClassDef(TrdTrack, 4);
};
```

```
class TrdSpacePoint : public TAdamoRow
{
public:
  TrdSpacePoint();
  virtual ~TrdSpacePoint();

  ...

  void          AddTrack(TrdTrack * track);
  Int_t         GetNTracks() { return fNTracks; }
  TrdTrack *    GetTrack(Int_t idx);

  ...

  // Number of tracks through spacepoint
  Int_t       fNTracks;
  // Array of references to tracks
  TRefArray   fTracks;

  ClassDef(TrdSpacePoint, 3);
};
```

- A Track knows from which spacepoints it is made of
- A Spacepoint knows which tracks it belongs to
- Member functions provide “navigation”
  - From track to spacepoints
  - From spacepoint to tracks

The General Idea



XTC
EXTERNAL TRACK RECONSTRUCTION

XTCSelector
TSelector

ADAMO to ROOT Interface

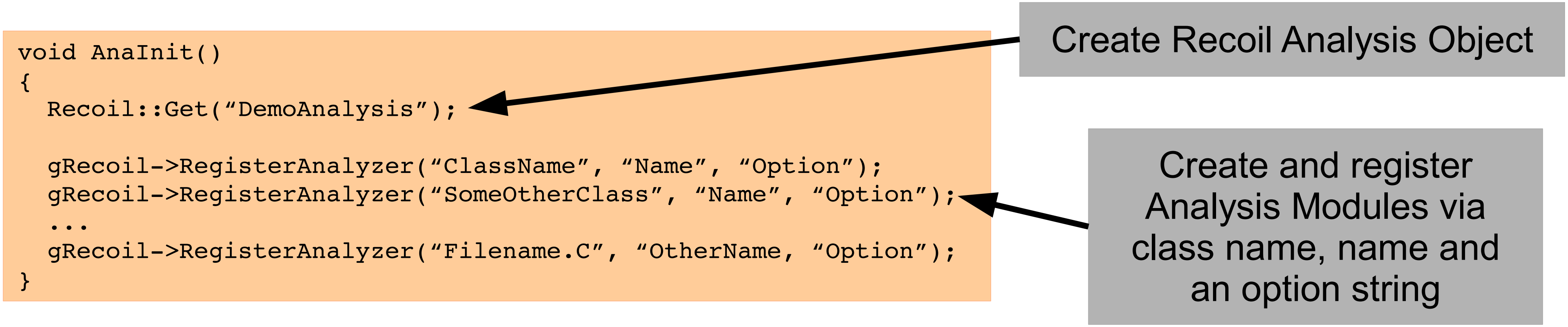
Recoil Analysis Object

ROOT Files

ROOT Tree

The Recoil Analysis Object

- Handles processing of *Analysis Modules (RecoilAnalyzer)*
- Provides access to data (tracks, clusters, ...)
- Takes care of output
  - One output file with histos and cuts in directory structure
  - Optional second file with a ROOT Tree
  - Common interface to tree



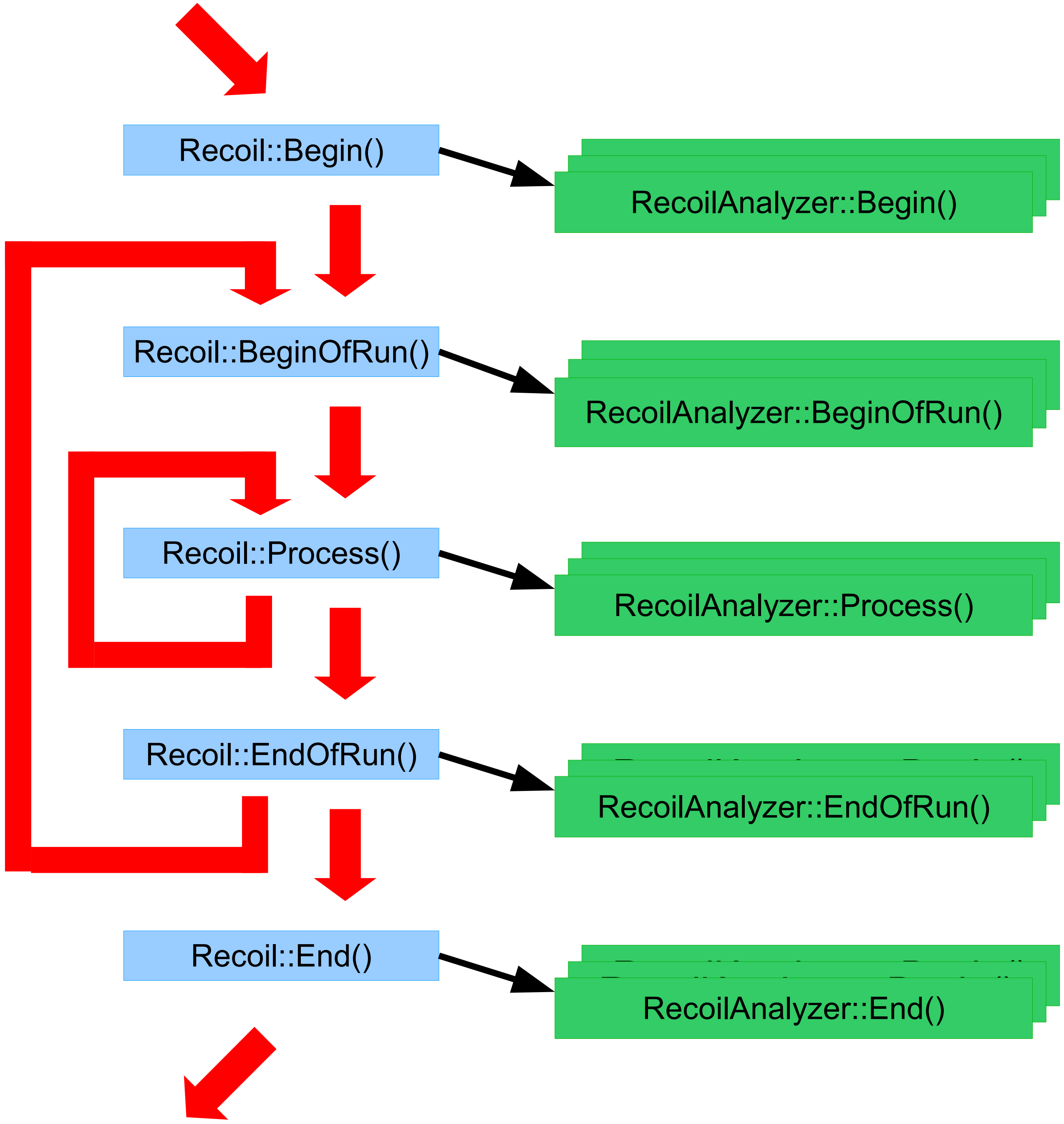
```
void AnaInit()
{
  Recoil::Get(“DemoAnalysis”);

  gRecoil->RegisterAnalyzer(“ClassName”, “Name”, “Option”);
  gRecoil->RegisterAnalyzer(“SomeOtherClass”, “Name”, “Option”);
  ...
  gRecoil->RegisterAnalyzer(“Filename.C”, “OtherName, “Option”);
}
```

Create Recoil Analysis Object

Create and register Analysis Modules via class name, name and an option string

- Allows multiple *Analysis Modules* of the same type
- Modules identified via class name and name
- Option string can be used to control module behavior
- Modules can be compiled at run-time

Eventloop



Recoil::Begin()

RecoilAnalyzer::Begin()

Called once at startup

Recoil::BeginOfRun()

RecoilAnalyzer::BeginOfRun()

Called at beginning of each HERMES run

Recoil::Process()

RecoilAnalyzer::Process()

Called for each event

Recoil::EndOfRun()

RecoilAnalyzer::EndOfRun()

Called at end of each HERMES run

Recoil::End()

RecoilAnalyzer::End()

Called once at end

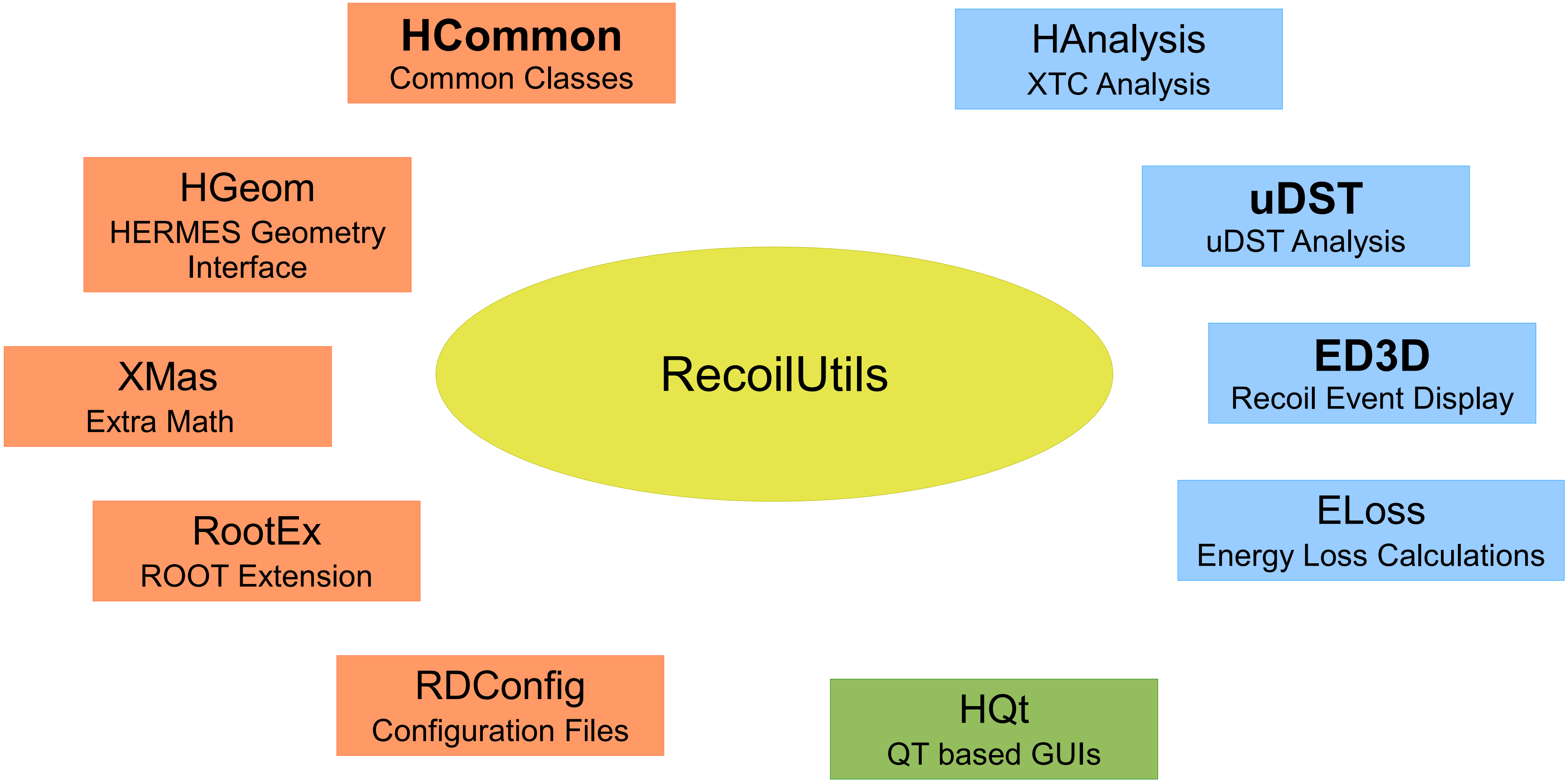
Analysis Modules

- Can/Must implement
  - *Begin()*
    - Create histos etc.
    - Add branches to output tree
    - Create slave analysis modules
  - *BeginOfRun()*
  - *Process()*
    - Analysis of data is done here
  - *EndOfRun()*
  - *End()*
    - e.g. fit histos
- Registered with the analysis object
  - Processed for each event
- Slave module of another analysis module
  - Processed on demand

Outline

- The HERMES Recoil Detector
- General Idea
- A brief Project Overview
- Selected Features
  - uDST Framework
  - Cut Management
  - Event Display
- Summary and Future Plans

RecoilUtils Overview



RecoilUtils

HCommon
Common Classes

HGeom
HERMES Geometry Interface

XMas
Extra Math

RootEx
ROOT Extension

RDConfig
Configuration Files

HAnalysis
XTC Analysis

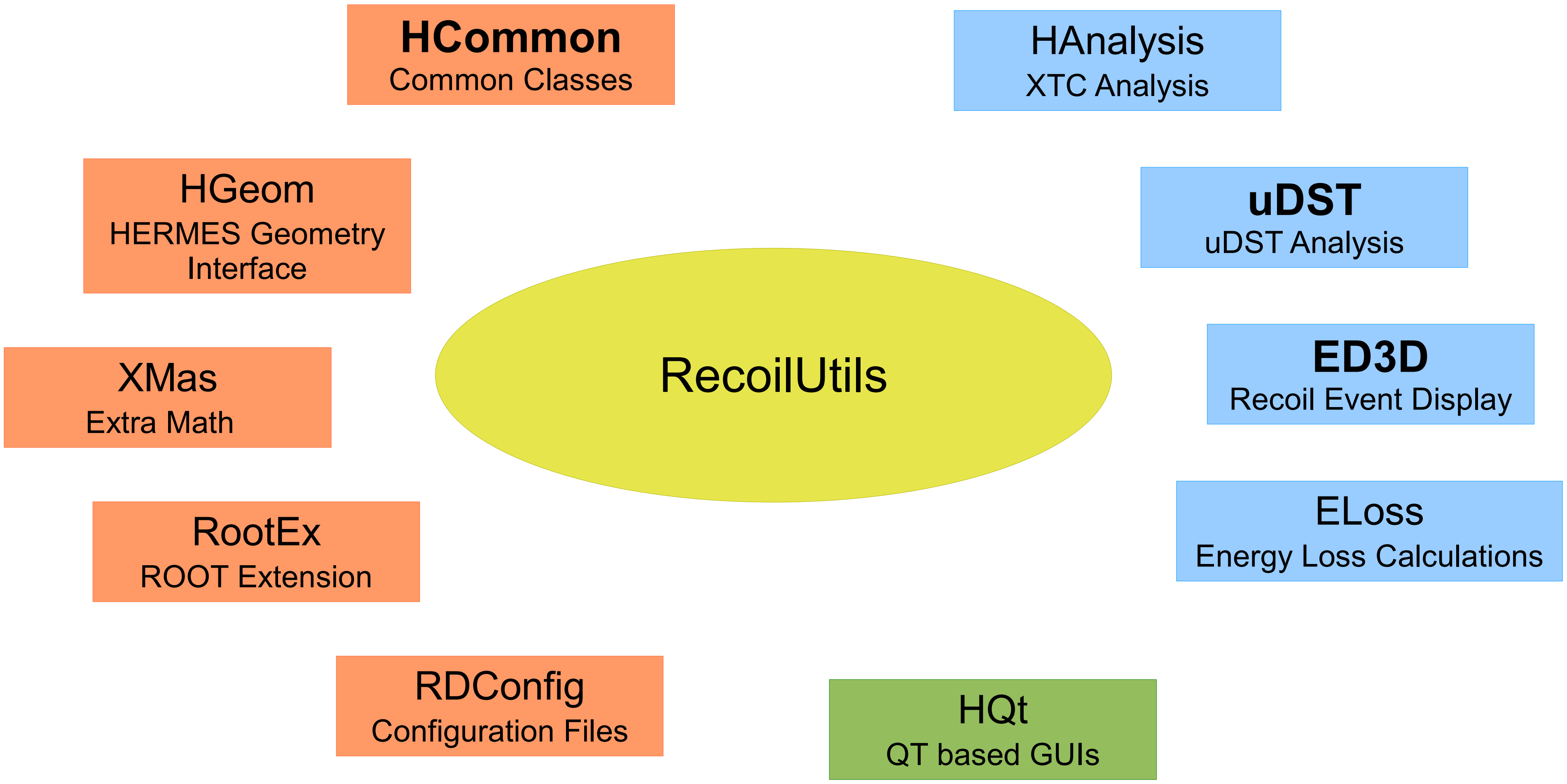
uDST
uDST Analysis

ED3D
Recoil Event Display

ELoss
Energy Loss Calculations

HQt
QT based GUIs

RecoilUtils Overview



HCommon
Common Classes

HAnalysis
XTC Analysis

HGeom
HERMES Geometry Interface

uDST
uDST Analysis

XMas
Extra Math

RecoilUtils

ED3D
Recoil Event Display

RootEx
ROOT Extension

ELoss
Energy Loss Calculations

RDConfig
Configuration Files

HQt
QT based GUIs

865	source files
85000	lines of code
319	classes

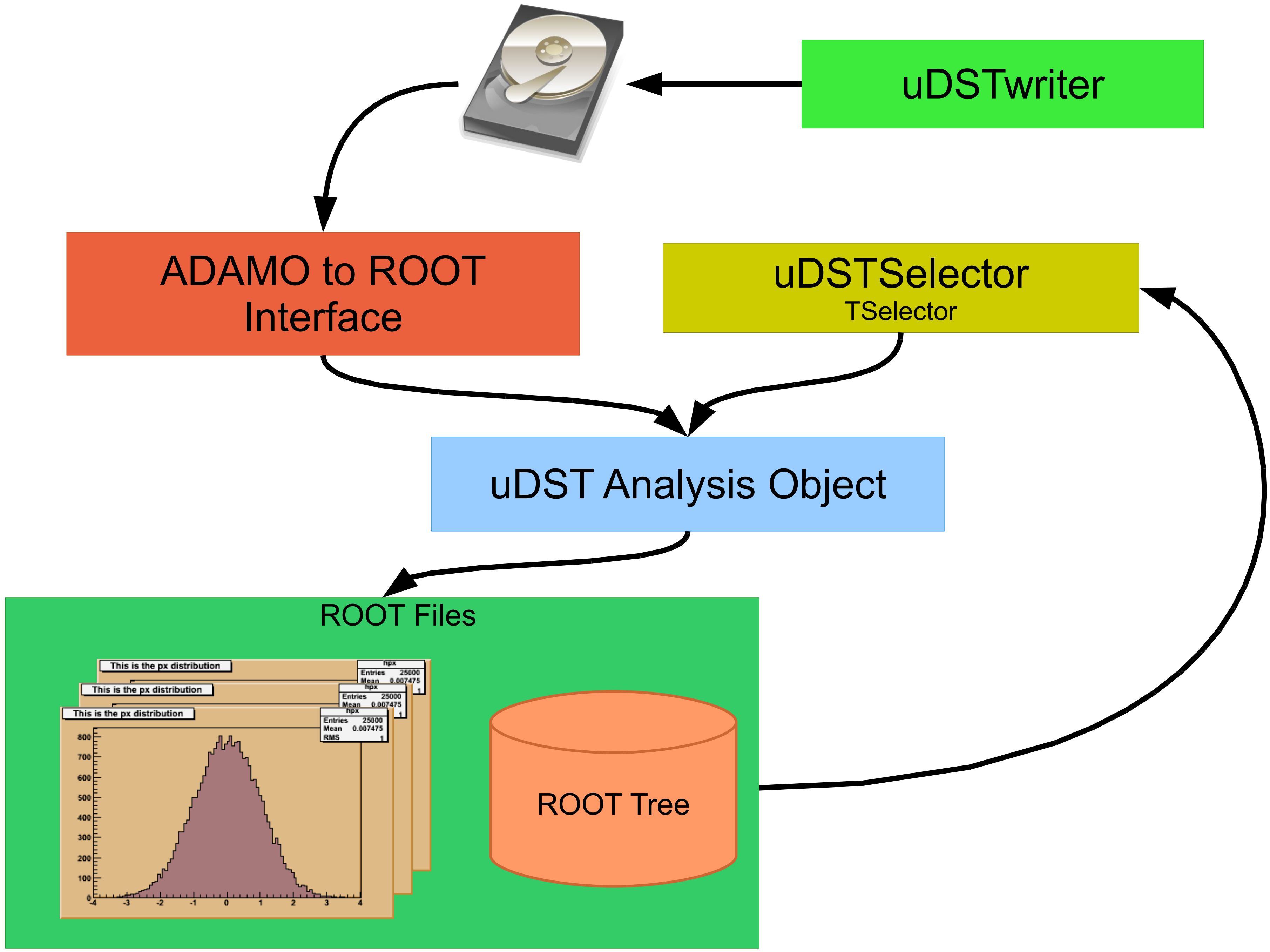
Outline

- The HERMES Recoil Detector
- General Idea
- A brief Project Overview
- Selected Features
  - uDST Framework
  - Cut Management
  - Event Display
- Summary and Future Plans

The uDST Framework

- Used for physics analysis
- Very similar to XTC framework
  - Main analysis object is called *uDST*
  - Analysis modules derived from *uDSTAnalyzer*
- But...
  - uDSTs contain only stripped information (no raw data and hits)
  - uDSTs contain slow control information
  - Data is split into bursts (10 s)

The uDST Framework



uDSTwriter

ADAMO to ROOT Interface

uDSTSelector

TSelector

uDST Analysis Object

ROOT Files

ROOT Tree

The uDST Framework

- Additional methods in *uDSTAnalyzer*
  - *BeginOfBurst()*
  - *EndOfBurst()*
- Certain bursts may be skipped due to data quality
- Introduce burst selector base class *uDSTVBurstSelect*
- Burst selector must implement *IsGoodBurst()*
  - Burst selector is processed at beginning of each burst
  - Burst is skipped if *IsGoodBurst()* returns *false*

```
void AnaInit()
{
  uDST::Get(“uDSTDemo”);

  guDST->RegisterBurstSelect(“ClassName”, “Name, “Option”);

  guDST->RegisterAnalyzer(“ClassName”, “Name”, “Option”);
  ...
}
```

Outline

- The HERMES Recoil Detector
- General Idea
- A brief Project Overview
- Selected Features
  - uDST Framework
  - Cut Management
  - Event Display
- Summary and Future Plans

Cut Management

- Analysis modules can be used by different people
  - Different analyzers might want / try different cuts
- Cuts must not be hidden somewhere in the code
  - Documentation
  - Transparency
- Values of cuts should not be hard-coded
- Using a different set of cuts should work without recompiling the code

Cut Management

- All cuts declared in header files of analysis modules
- Substitute “basic” types by corresponding cut classes
  - *Double_t* → *THCutD*
  - *Int_t* → *THCutI*
  - *TF1* → *THCutF1*
- Cut classes provide all methods and operators known from “basic” types
- Set default values for cuts in constructor of analysis module
- At startup: analysis modules register cuts with a cut manager (TCutManager)

TCutManager

- TCutManager stores cuts in folders
- Cuts are identified by
  - Name and type of the cut
  - Name of the analysis module
  - Classname of the analysis module
  - Path in analysis module hierarchy
- Provides XML IO
  - All cuts with value and description in one file
  - Cuts can be loaded at startup
- Allows multiple analysis modules of same type but with different cuts

Cut Management – An Example

DemoModule.h

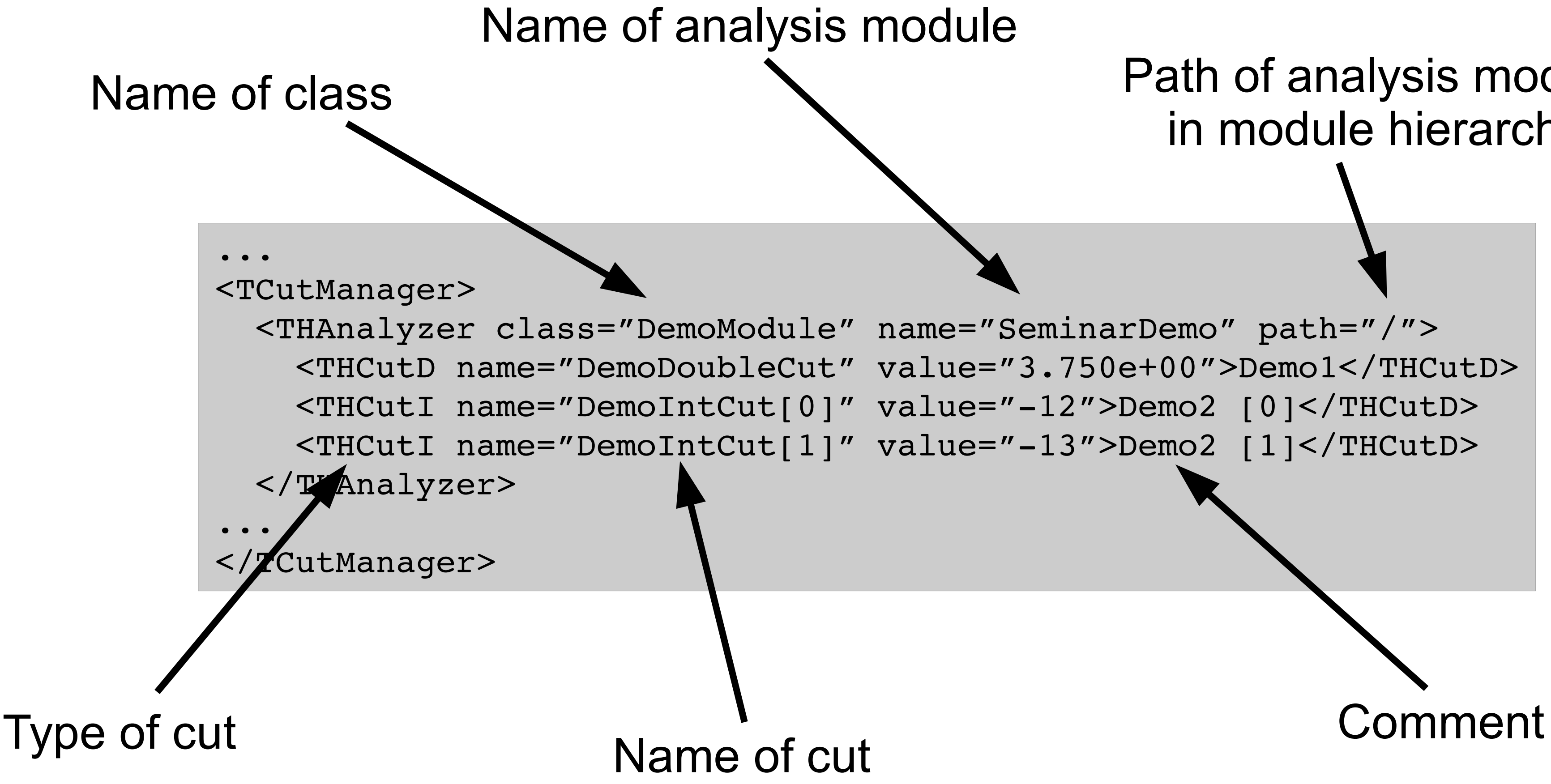
```
class DemoModule : public RecoilAnalyzer
{
public:
  DemoModule(const char * name, const char * option);
  virtual ~DemoModule();

  virtual void   Process();
  ...

protected:

  THCutD      DemoDoubleCut; // Demo1
  THCutI      DemoIntCut[2]; // Demo2

  ClassDef(DemoModule, 0)
};
```

DemoModule.C

```
DemoModule::DemoModule(const char * name, const char * option)
  :RecoilAnalyzer(name, option)
{
  DemoDoubleCut = 3.75;
  DemoIntCut[0] = -12;
  DemoIntCut[1] = -13;
}

...

void DemoModule::Process()
{
  if (DemoDoubleCut>2.5)
    do something

  ...
}
```

```
void AnaInit()
{
  Recoil::Get(“DesyITSeminarDemo”);

  gRecoil->RegisterAnalyzer(“DemoModule”, “SeminarDemo”, “no option for now”);
}
```

Cut Management – An Example XML File



Name of analysis module

Name of class

Path of analysis module in module hierarchy

```
...
<TCutManager>
  <THAnalyzer class=”DemoModule” name=”SeminarDemo” path=”/”>
    <THCutD name=”DemoDoubleCut” value=”3.750e+00”>Demo1</THCutD>
    <THCutI name=”DemoIntCut[0]” value=”-12”>Demo2 [0]</THCutD>
    <THCutI name=”DemoIntCut[1]” value=”-13”>Demo2 [1]</THCutD>
  </THAnalyzer>
...
</TCutManager>
```

Type of cut

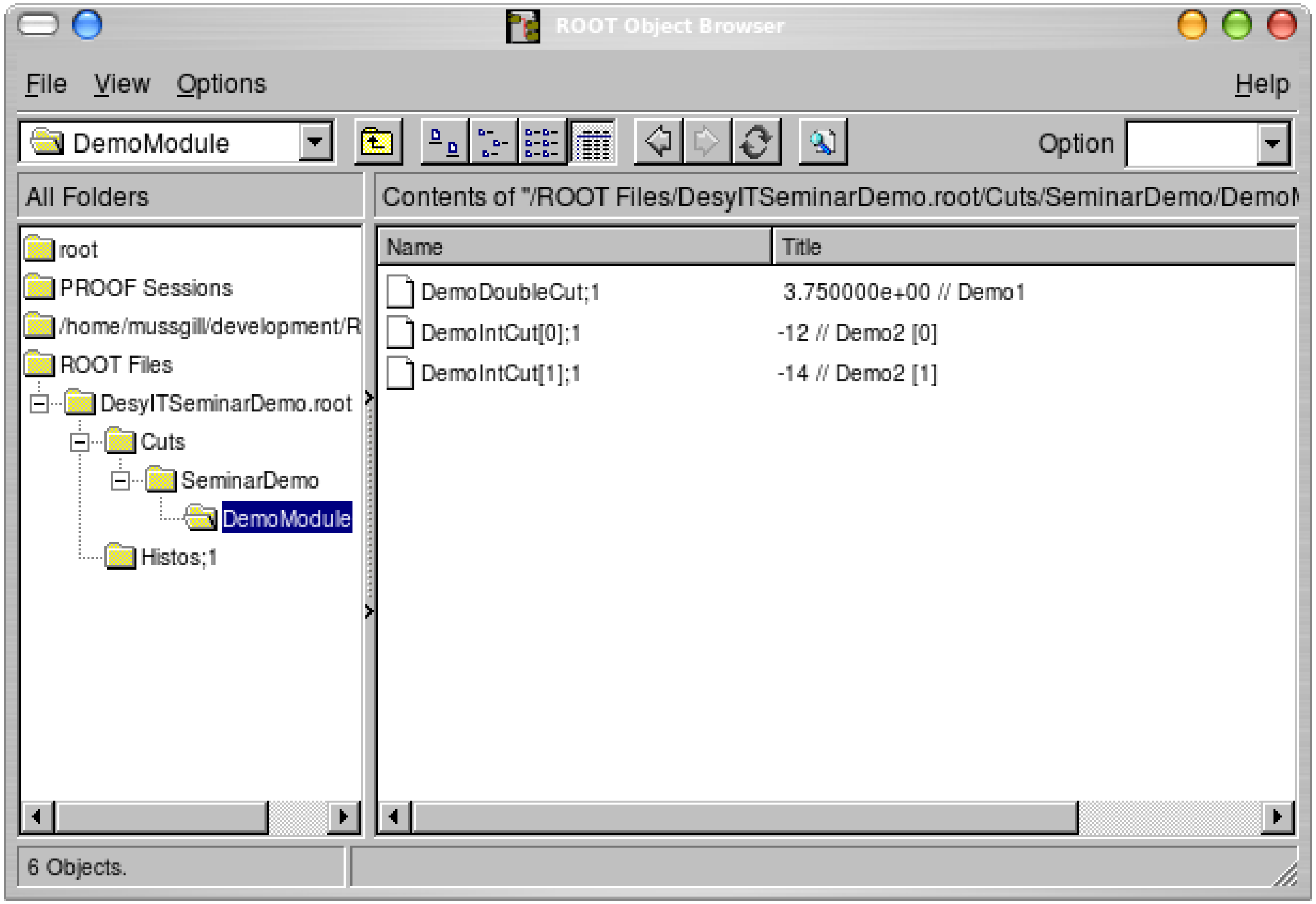
Name of cut

Comment

- Analysis module header file used for documentation of cuts
- Dictionary provides all information during run-time
  - Type of cut
  - Name of cut
  - Comment → Documentation

Cut Management

- All cuts are saved to output root file
  - With directory structure
  - Cuts are “browseable”



Contents of "/ROOT Files/DesyITSeminarDemo.root/Cuts/SeminarDemo/Demo

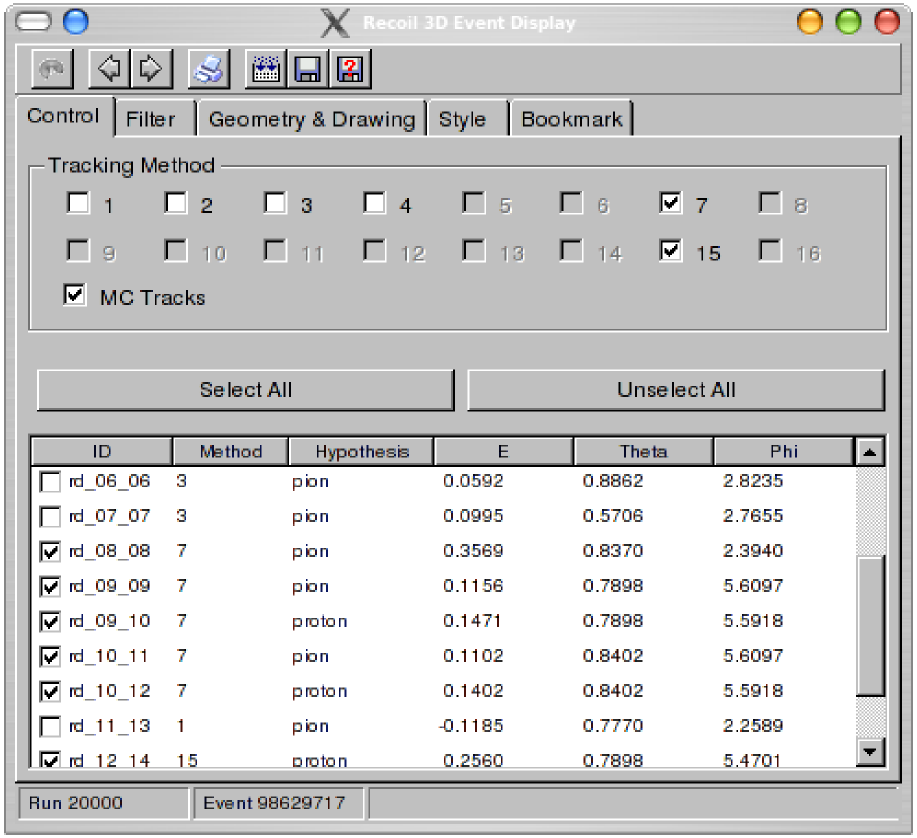
Name	Title
DemoDoubleCut;1	3.750000e+00 // Demo1
DemoIntCut[0];1	-12 // Demo2 [0]
DemoIntCut[1];1	-14 // Demo2 [1]

6 Objects.

Outline

- The HERMES Recoil Detector
- General Idea
- A brief Project Overview
- Selected Features
  - uDST Framework
  - Cut Management
  - Event Display
- Summary and Future Plans

ED3D – The Recoil Event Display



Recoil 3D Event Display

Control | Filter | Geometry & Drawing | Style | Bookmark

Tracking Method

□ 1 □ 2 □ 3 □ 4 □ 5 □ 6 ☑ 7 □ 8

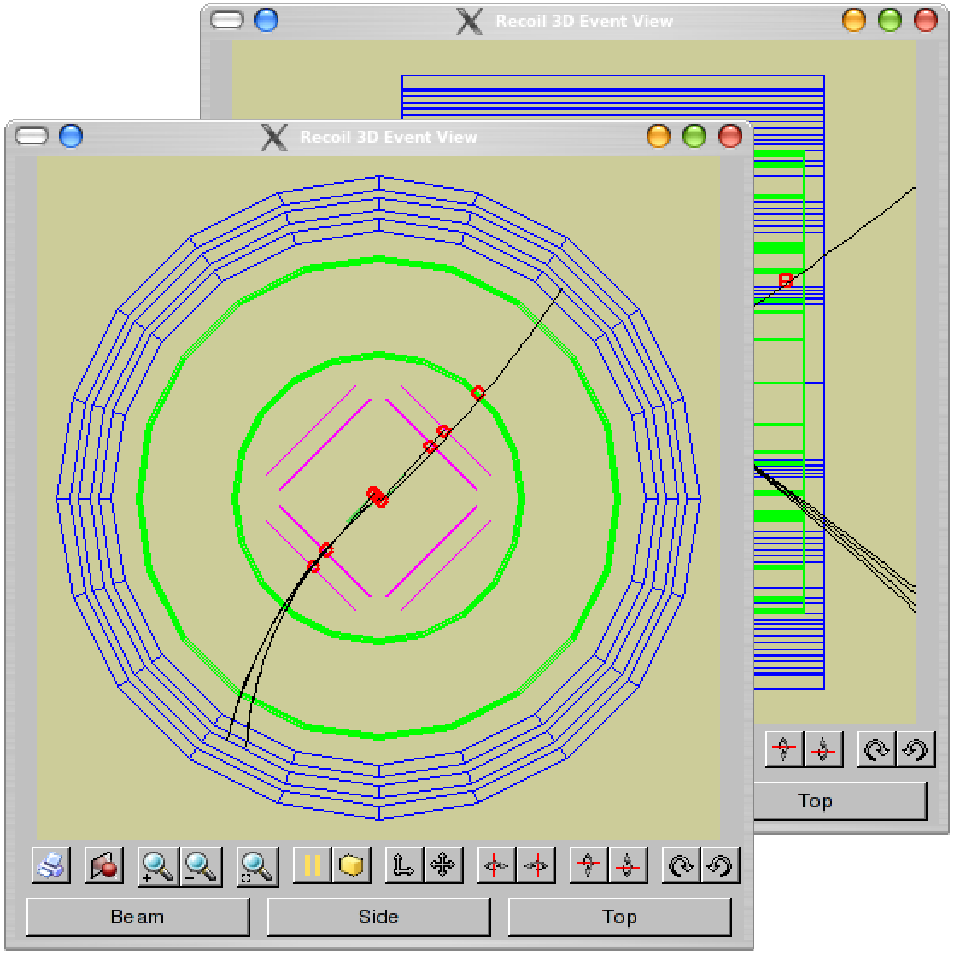
□ 9 □ 10 □ 11 □ 12 □ 13 □ 14 ☑ 15 □ 16

☑ MC Tracks

Select All | Unselect All

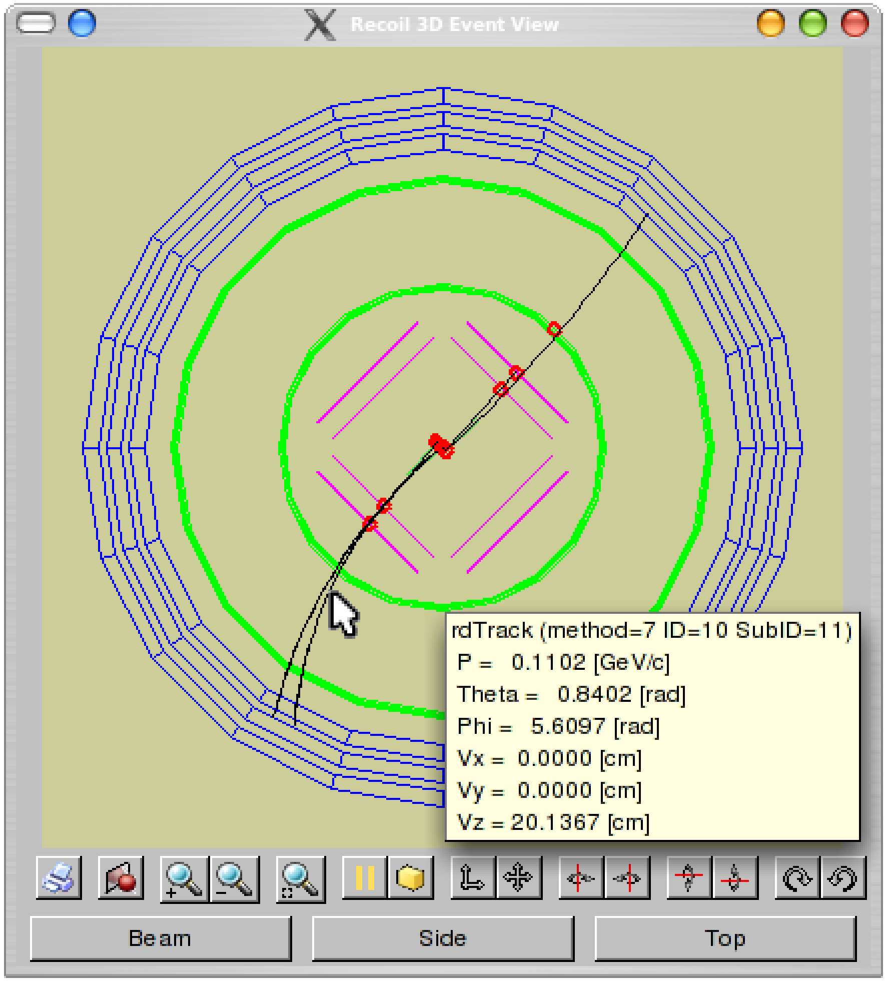
ID	Method	Hypothesis	E	Theta	Phi
□ rd_06_06	3	pion	0.0592	0.8862	2.8235
□ rd_07_07	3	pion	0.0995	0.5706	2.7655
☑ rd_08_08	7	pion	0.3569	0.8370	2.3940
☑ rd_09_09	7	pion	0.1156	0.7898	5.6097
☑ rd_09_10	7	proton	0.1471	0.7898	5.5918
☑ rd_10_11	7	pion	0.1102	0.8402	5.6097
☑ rd_10_12	7	proton	0.1402	0.8402	5.5918
□ rd_11_13	1	pion	-0.1185	0.7770	2.2589
☑ rd_12_14	15	proton	0.2560	0.7898	5.4701

Run 20000 | Event 98629717

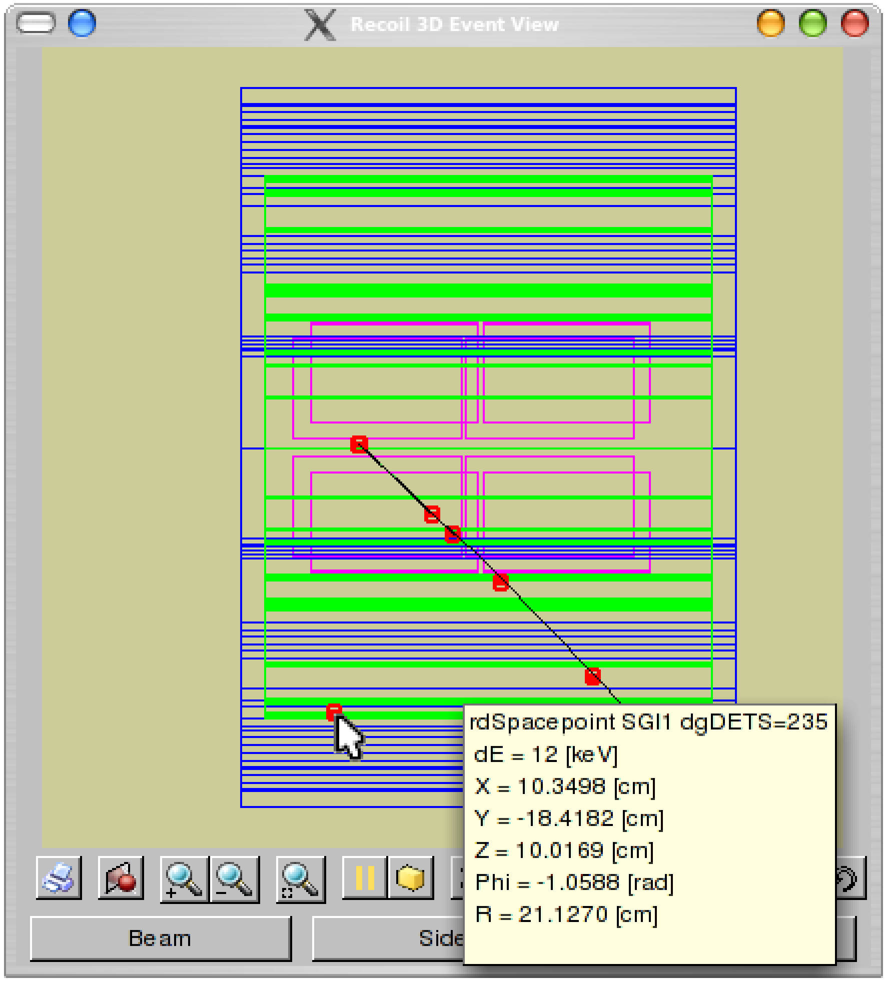


- Uses ROOT GUI classes and TGeoManager
- Allows multiple independent 3D views
- Tooltip information for selected tracks and spacepoints
- Filter on track and event parameter
- Bookmarks

ED3D – Tooltips



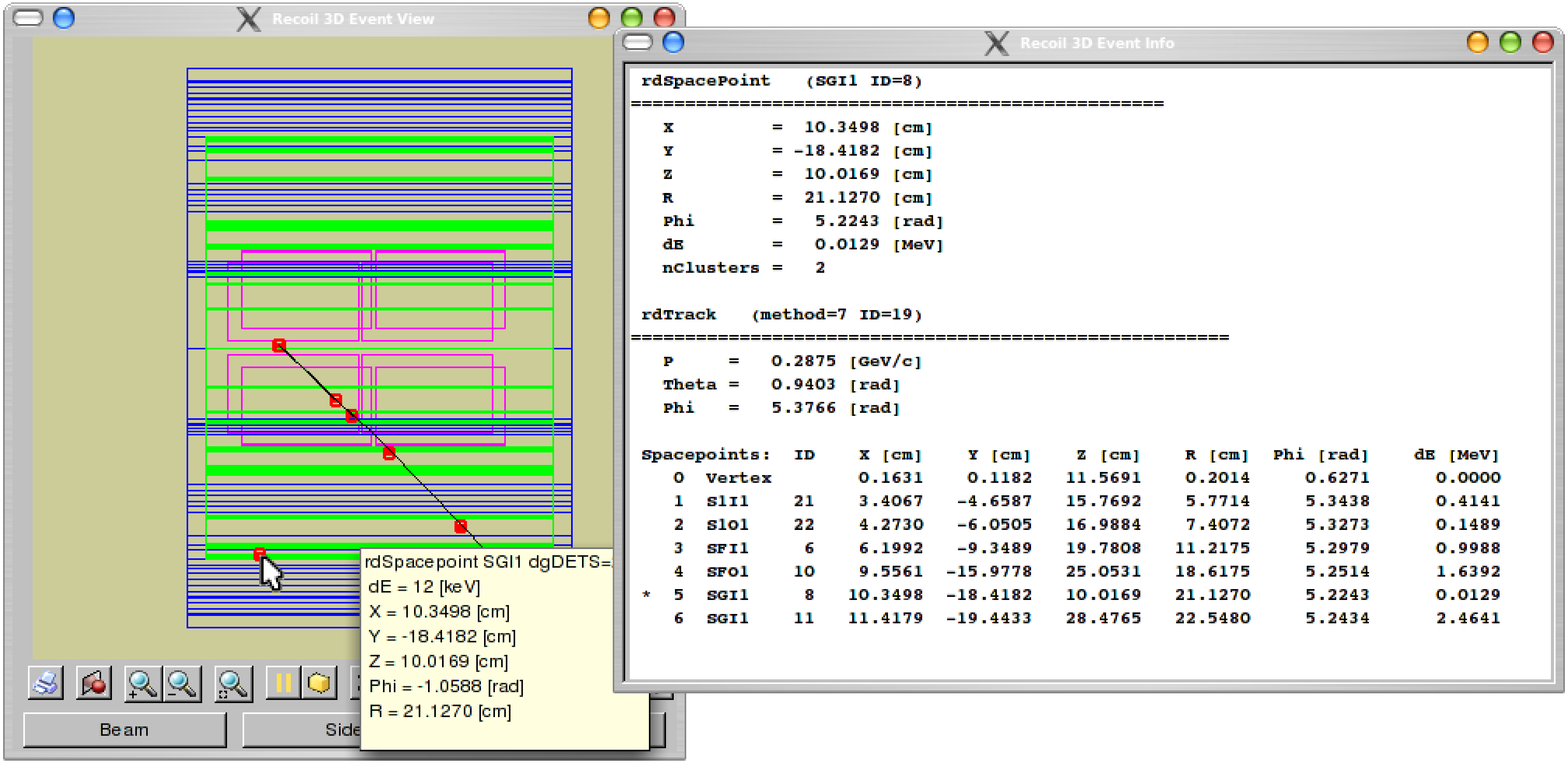
rdTrack (method=7 ID=10 SubID=11)
P = 0.1102 [GeV/c]
Theta = 0.8402 [rad]
Phi = 5.6097 [rad]
Vx = 0.0000 [cm]
Vy = 0.0000 [cm]
Vz = 20.1367 [cm]



rdSpacepoint SGI1 dgDETS=235
dE = 12 [keV]
X = 10.3498 [cm]
Y = -18.4182 [cm]
Z = 10.0169 [cm]
Phi = -1.0588 [rad]
R = 21.1270 [cm]

- Tracks and spacepoints are “selectable”
- Tooltips show basic information
  - Tracks: Momentum, Angles and Vertex
  - Spacepoints: Energy and Position

ED3D – Event Info View

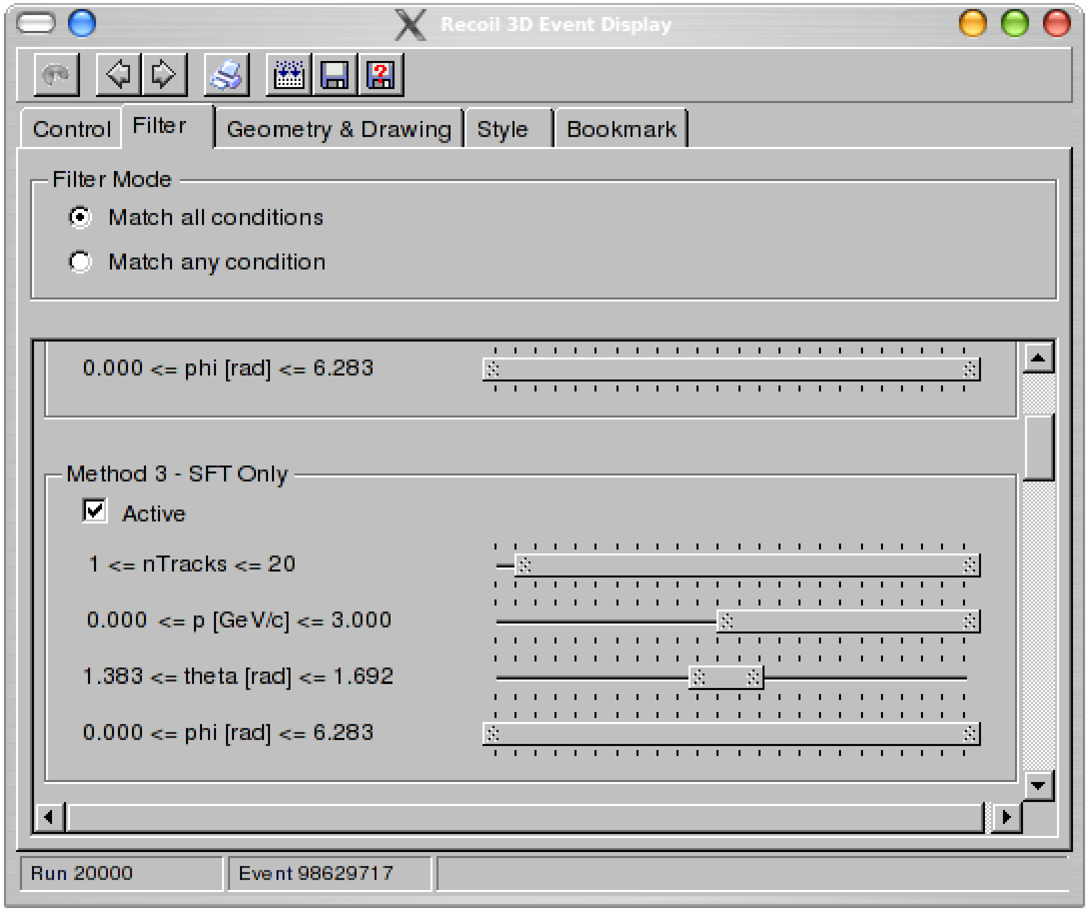


```
rdSpacepoint SGI1 dgDETS=
dE = 12 [keV]
X = 10.3498 [cm]
Y = -18.4182 [cm]
Z = 10.0169 [cm]
Phi = -1.0588 [rad]
R = 21.1270 [cm]
```

```
rdSpacePoint   (SGI1  ID=8)
==================================================
   X          =  10.3498 [cm]
   Y          = -18.4182 [cm]
   Z          =  10.0169 [cm]
   R          =  21.1270 [cm]
   Phi        =   5.2243 [rad]
   dE         =   0.0129 [MeV]
   nClusters  =   2

rdTrack    (method=7  ID=19)
========================================================
   P      =   0.2875 [GeV/c]
   Theta  =   0.9403 [rad]
   Phi    =   5.3766 [rad]

Spacepoints:  ID      X [cm]     Y [cm]     Z [cm]     R [cm]   Phi [rad]    dE [MeV]
   0  Vertex           0.1631     0.1182    11.5691     0.2014     0.6271      0.0000
   1  s1I1     21      3.4067    -4.6587    15.7692     5.7714     5.3438      0.4141
   2  s1o1     22      4.2730    -6.0505    16.9884     7.4072     5.3273      0.1489
   3  SFI1      6      6.1992    -9.3489    19.7808    11.2175     5.2979      0.9988
   4  SFo1     10      9.5561   -15.9778    25.0531    18.6175     5.2514      1.6392
*  5  SGI1      8     10.3498   -18.4182    10.0169    21.1270     5.2243      0.0129
   6  SGI1     11     11.4179   -19.4433    28.4765    22.5480     5.2434      2.4641
```

- Event information is printed in an extra window
- For a “selected” track all spacepoints with energies and coodinates are shown
- For a “selected” spacepoint all associated tracks are shown

ED3D – Track Filter



Filter Mode

● Match all conditions

○ Match any condition

0.000 <= phi [rad] <= 6.283

Method 3 - SFT Only

☑ Active

1 <= nTracks <= 20

0.000 <= p [GeV/c] <= 3.000

1.383 <= theta [rad] <= 1.692

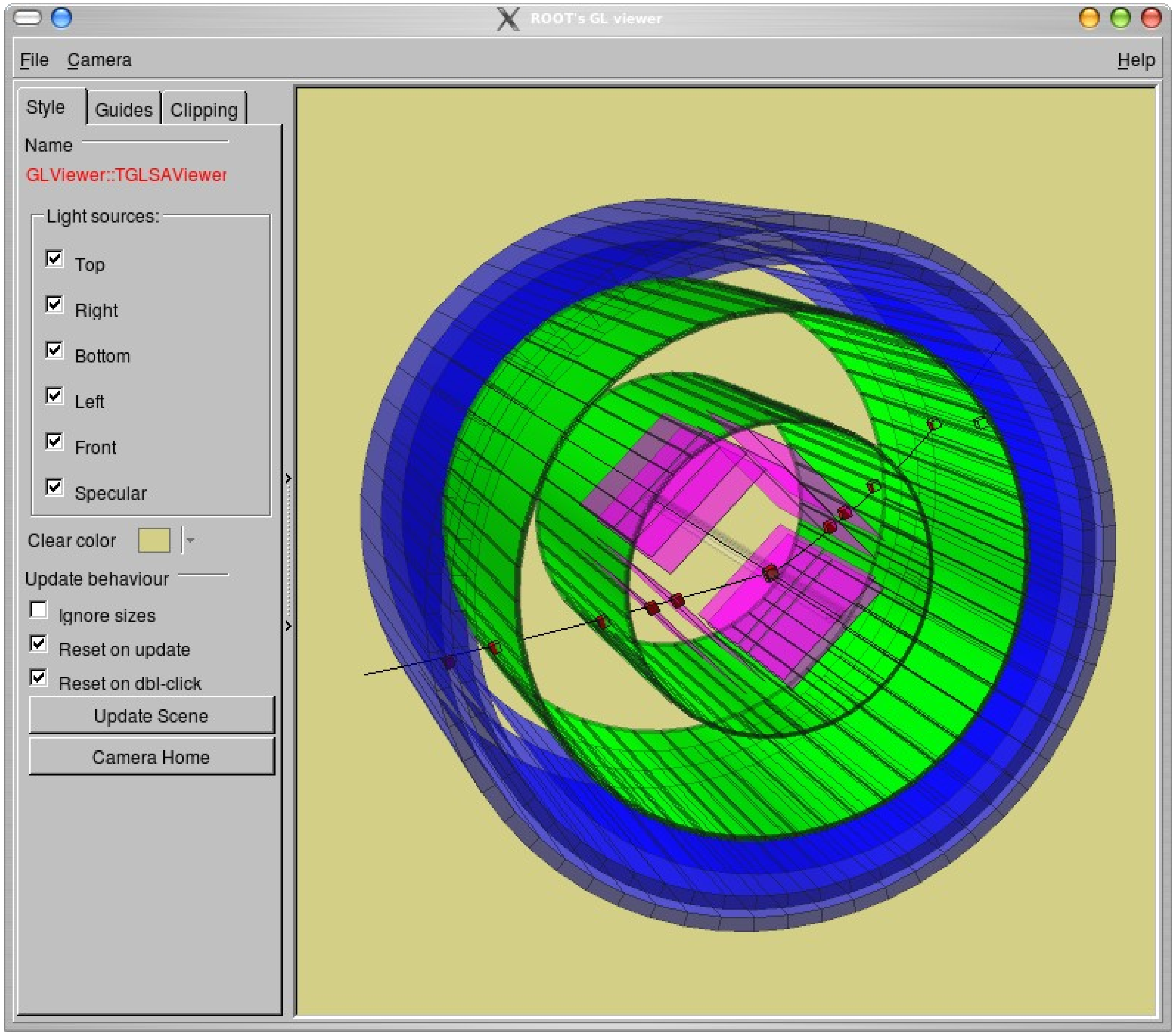
0.000 <= phi [rad] <= 6.283

Run 20000

Event 98629717

- Display only tracks/events that fulfill certain conditions
- Extendable by *user filters*
  - Code will be compiled on startup of event display
  - Filters will appear in GUI

ED3D – OpenGL View



Name

GLViewer::TGLSAViewer

Light sources:

☑ Top

☑ Right

☑ Bottom

☑ Left

☑ Front

☑ Specular

Clear color

Update behaviour

□ Ignore sizes

☑ Reset on update

☑ Reset on dbl-click

- Uses ROOT's standard OpenGL viewer
- Tracks and spacepoints are not selectable

Outline

- The HERMES Recoil Detector
- General Idea
- A brief Project Overview
- Selected Features
  - uDST Framework
  - Cut Management
  - Event Display
- Summary and Future Plans

Summary and Future Plans

- XTC Framework
  - well tested
  - heavily used
- uDST Framework
  - needs a bit more testing
  - first DVCS analysis is currently done

- Geant4
  - offers different low energy models (interesting for Silicon Detector)
- uDST Framework
  - need more tested analysis modules

Get more people to use the software