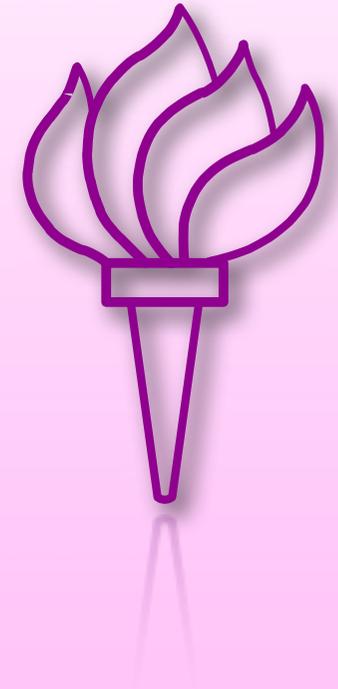


D3PDMaker/D3PDReader status and plans



Attila Krasznahorkay



NEW YORK UNIVERSITY

D3PDMaker status



- Current best tags in 16.0.2.7
 - Usable for “simple” studies out of the box
 - Will start the “rationalisation” of the code in the AtlasProduction-16.0.X.Y nightlies
 - Don't want to have a separate cache for this, as physics caches are always on top of AtlasProduction
- Many packages undergoing some changes already
 - Modifying what kind of trigger information is included with the offline info. (MuonD3PDMaker)
 - Adding some corrections to the objects before dumping them (TauD3PDMaker)

Organisational plans



- Have to improve the communication between the D3PDMaker developers themselves, plus the developers and the users
 - For internal discussions, just recently created atlas-sw-pat-d3pd-developers@cern.ch
 - Started collecting information from the developers to learn who is working on what
 - The feedback has been very good after just one e-mail
- Will re-write some of the D3PDMaker TWiki pages
 - Main TWiki page should tell about the package layout, which package is for what, and who is working on what

Continued plans



- Each (“core”) package should have its own TWiki, listing all the D3PDObject-s that it provides, and what they are for
- Should create some searchable library of what kind of D3PDObject-s are available
 - Not yet sure what’s the best way of doing it, but it will have to be done
- Trigger packages: D3PDObject concept not perfect for trigger information
 - Many packages started defining python functions which add all the information from that trigger signature
 - Will have to unify how these are defined/used

Other details



- Cut down on the number of branches with trigger names in them
 - Event level decisions, like “EF_mu20”, “L2_e15_medium”
 - Object level decisions, like “trig_EF_el_EF_e20_medium”
 - Some of these can remain, but we can't have hundreds of them
- Will have to produce a short document about naming conventions, and get all developers to follow it
 - Situation not tragic at the moment, but it could definitely be improved

D3PDReader

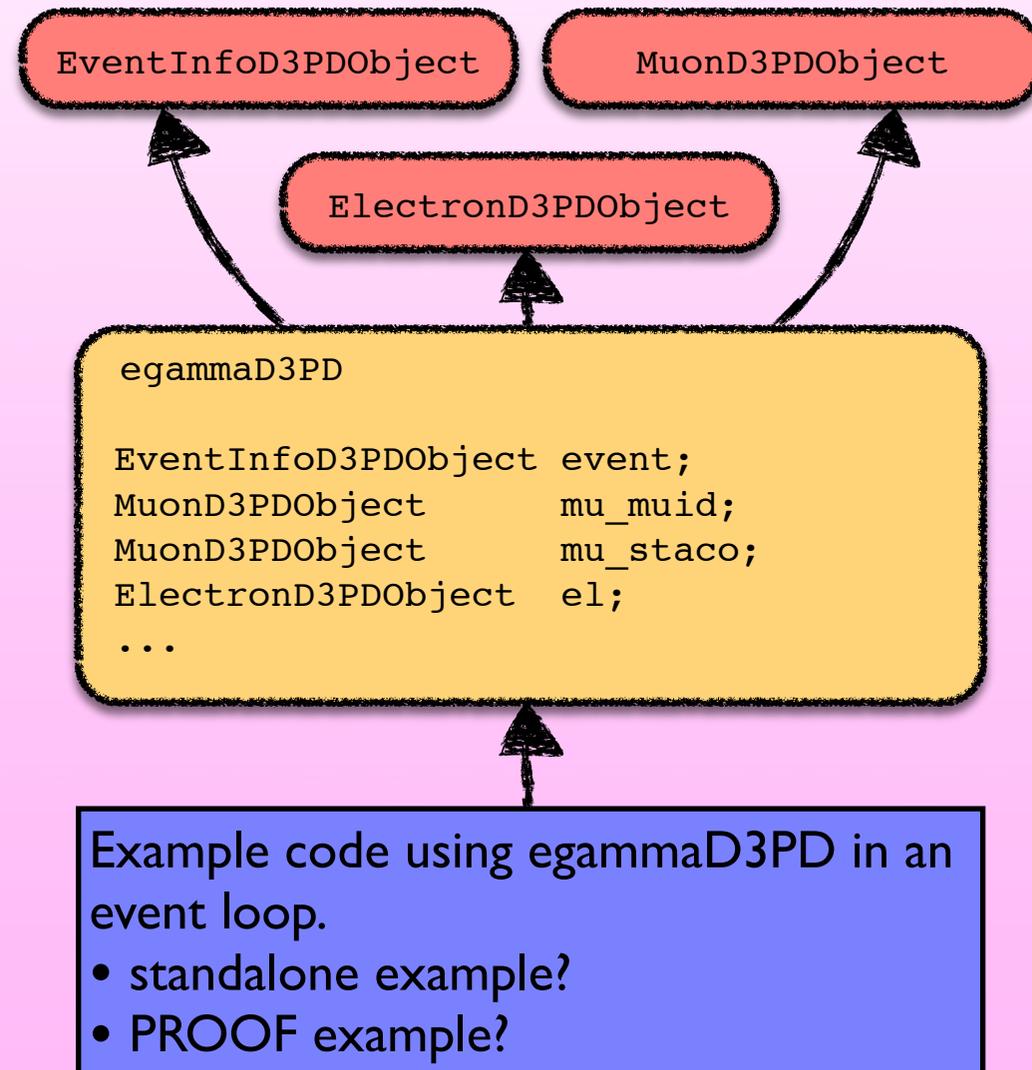


- Discussed/presented in some meetings already:
 - <http://indico.cern.ch/getFile.py/access?contribId=12&resId=0&materialId=slides&confId=88419>
 - <http://indico.cern.ch/getFile.py/access?contribId=11&resId=0&materialId=slides&confId=88417>
- Meant to help people write efficient/simple C++ code for reading D3PDs
- Collected input from a number of people

Basic ideas



- One C++ class for reading back the variables produced by one D3PDObject
 - Reduces the class size (no huge classes with thousands of variables)
 - Improves readability
- These can be combined into one class describing a full D3PD if needed
- Finally, provide a full example on top of everything
- Users can adapt the code starting from any of these levels



Code creation



- Code automatically generated by specialised D3PDMaker classes
 - Code generation done by special Athena jobs
- Special D3PD object collects the variables that the various tools want to create, and at the finalisation they create the C++ source files from this information
- Athena code structure allows to have multiple code generator versions around
 - New reader code ideas can be tested without interfering with existing code generators
- Prototype under:
<https://svnweb.cern.ch/trac/atlasusr/browser/krasznaa/D3PD/D3PDMakerReader/trunk>

Code creator job



- Currently:

```
from D3PDMakerReader.CodeGenerators import makeCppRootReaderV1
from EventCommonD3PDMaker.EventInfoD3PDObject import EventInfoD3PDObject

makeCppRootReaderV1( EventInfoD3PDObject( 10 ), "EventInfoD3PDObject" )
```

- Eventually (will need some modifications in the core D3PDMaker code, works privately):

```
import D3PDMakerReader
d3pdalg = D3PDMakerReader.ReaderAlg( "d3pdReader", Directory = "output/" )

from EventCommonD3PDMaker.EventInfoD3PDObject import EventInfoD3PDObject
d3pdalg += EventInfoD3PDObject( 1 )
from egammaD3PDMaker.ElectronD3PDObject import ElectronD3PDObject
d3pdalg += ElectronD3PDObject( 2 )
```

Direct variable access



- All variable access done through proxy classes:
`D3PDReader::VarHandle<>`
 - Proxies know the name of the branch that they have to handle
 - They only read the content of the branch when/if the user accesses them
 - Accessing a single variable:

```
D3PDReader::VarHandle< Int_t > n( ..., "el_n", ... );  
...  
std::cout << "Number of electrons: " << n() << std::endl;
```

- The *D3PDObject classes are constructed using such proxy objects

Using the D3PDObject classes



```
Long64_t master = 0;
D3PDReader::EventInfoD3PDObject event( &master );
TTree* tree = ...;
event.readFrom( tree );
for( master = 0; master < tree->GetEntries(); ++master ) {
    std::cout << "Run/Event numbers: " << event.RunNumber() << " / "
                << event.EventNumber() << std::endl;
}
}
```

- All proxies get the current entry number from a single variable -> User doesn't have to explicitly load the variables (s)he wants to use
- The same D3PDObject can be switched to a new TTree (when opening a new file) by just calling readFrom(. . .) again

Other capabilities



- Use the proxy objects to create new derived ntuples in an analysis

```
D3PDReader::EventInfoD3PDObject event( 0 );
TTree* tree = ...; ///< Create an output TTree
event.setActive(); ///< Enable the writing of all the variables
event.writeTo( tree ); ///< Instruct the object to write itself in the output
for( ... ) {
    event.RunNumber() = ...;
    tree->Fill();
}
```

- Or... to skim/slim/thin a D3PD!

```
D3PDReader::EventInfoD3PDObject event(...);
TTree* itree = ...;
event.readFrom( itree ); ///< Read the variables from the input
TTree* otree = ...;
event.RunNumber.setActive(); event.lbn.setActive(); ...
event.writeTo( otree ); ///< Write the very same variables to the output
```

Outlook



- For a complete example of what's possible with the current prototype, see:
<https://svnweb.cern.ch/trac/atlasusr/browser/krasznaa/D3PD/D3PDMakerReader/trunk/test/test.cxx>
- Current plans:
 - Make some changes in core D3PDMaker packages to make it possible to generate the reader code by only modifying existing D3PDMaker jobOptions a little
 - Once the code can generate the full source for reading e.g. the egammaD3PD, move the package into the offline repository

Summary



- Quite a lot of manpower in the D3PDMaker development
 - With a bit of central organisation the code could become much more usable soon
- D3PDReader prototype already working quite nicely
 - Will need to add some modifications to core D3PDMaker packages to make it more user friendly
 - Once in a good shape, will create some tutorials for how to use it
 - And maybe also on how to use ROOT efficiently?