NeXus Code Camp

Mark Könnecke

Paul Scherrer Institute
Switzerland

October 22, 2011

API Topics

- NXclonehandle
- PHDF-driver
- PyTree-API
- C++ Tree API
- Cmake
- NXdict replacement design based on NXDL
- NXvalidate
- HDF-5 1.8.*
- Fedora installer
- 64 bit dataset sizes
- All dimensions unlimited

General Topics

- WWW-site from Manual
- NeXus for the impatient
- Encoding of axis dependencies
- Handling multi dimensional scans
- Sphinx discussion
- Cleanup NeXus applications
- NXdetetcor and Dectris

Presentations

- Cmake (Freddy)
- Sphinx (Pete J.)
- PHDF (Mark)
- HDF-5 1.6 to 1.8 (Freddy)
- C++ Tree API (Eugen W.)
- Public NeXus Talk (Mark K.)

- NXclonehandle
  - NAPI was made threadsafe on the last code camp
  - Each thread needs a file handle: NXclonehandle
- PHDF: driver for parallel HDF-5 to write and read really fast
- PyTree API:
  - Writen by Ray Osborn
  - Still needs unit tests
  - Critical for 4.3 NAPI release

- Eugen Wintersberger presents us his C++ tree API
- NXdict replacement:
  - NXdict is an additional C-API
  - Describes path of item in NeXus file via an external file
  - NXdict calls create structure as needed
  - CHANGE: use NXDL 4 structure description
  - CHANGE: review API
  - Inspiration: CDM-API
  - Seek collaboration with CDM? Affects also C++ API

- Switch from autoconf to Cmake
- NXvalidate: needs completion
- HDF-5 1.6.* to 1.8.*
- Fedora (rpm) installer
- 64 bit sizes: Freddie has to many events
- NX_UNLIMITED all over, HDF-5: yes, HDF-4, XML: No

- Short Manual (8 pages) about what NeXus is all about
- Content
  - Key Concepts
  - NeXus Benefits
  - Where do I find information?
  - How to access my data?
  - NeXus examples
  - Use cases

Axis Dependency Encoding

- Decided: extend NeXus to allow full mapping from CBF to NeXus
- Information to encode:

type	rotation or translation: DONE! transformation_type attribute
direction	vector around which to rotate or along which to translate: DONE! attribute
value	The angle of rotation or the length of translation, DONE!
dependency	The order of operations to place a component, to be discussed!

Expressing Axis Dependency in NeXus

- Implied: use existing NeXus coordinate system
- dependson attribute pointing to depending axis
- transform field in base classes which becomes a comma separated list of the path to the transformations required to position this component
- Create a special container to hold axis dependencies, NXdependency, to collect the dependencies in one place for easy access. This is what CIF does

sample,NXsample
    rotation_angle
    chi (dependson rotation_angle)
    phi (dependson phi)

sample,NXsample
    rotation_angle
    chi
    phi
    transform = rotation_angle,chi,phi

```
sample,NXsample
        rotation_angle
        chi
        phi
dependency,NXdependency
        sample/chi =
                sample/rotation_angle
        sample/phi =
                sample/chi
        instrument/detector/x_translation =
                instrument/detector/distance
        instrument/detector/distance =
                instrument/detector/polar_angle
```

NeXus

Handling Multi-Dimensional Scans

- Conflicting use cases:
  - Easy plotting
  - Careful data analysis

data,NXdata
- data[nx,ny]
  - @signal=1
- x[nx]
  - @axis=1
- y[ny]
  - @axis=2

data,NXdata
    data[nx,ny]
        @signal=1
    x[nx,ny]
    y[nx,ny]

data,NXdata
- data[nx,ny]
  - @signal=1
- x[nx,ny]
  - @axis=1,2
  - @label=1
- y[nx,ny]
  - @axis=1,2
  - @label=2

data,NXdata
- data[nx,ny]
  - @signal=1
  - @axes=x,y
  - @axesvalue=x_scan,y_scan
- x[nx]
  - @axis=1
- y[ny]
  - @axis=2
- x_scan[nx,ny]
- y_scan[nx,ny]

```
data,NXdata
        data[nx,ny]
                @signal=1
```

NXdetector Extensions for Dectris

- Dectris (Eiger, Pilatus, Mythen) going HDF-5 with NeXus conventions
- Additions to NXdetector for this kind of detector
- Programming model:
  - Dectris writes HDF-5 file with NXdetector
  - Local DAQ-system adds beamline metadata

Prioritise!

$$T = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

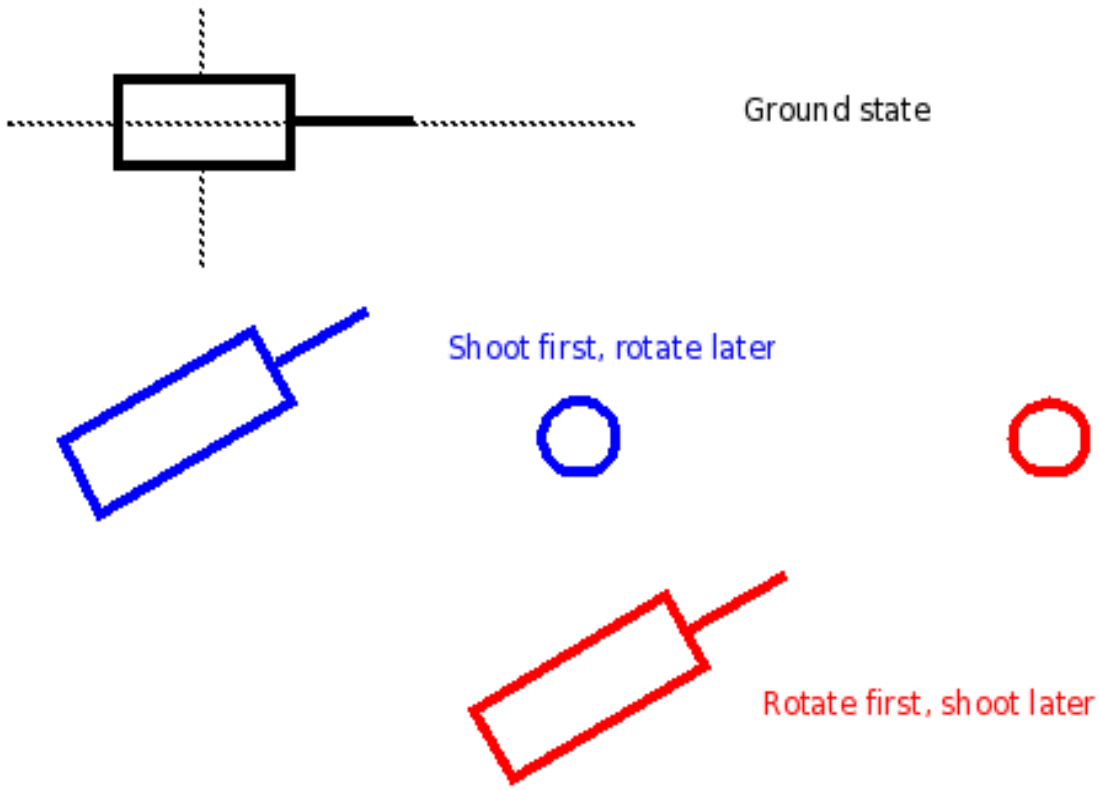
NeXus

$$T = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R = \begin{pmatrix} r11 & r12 & r13 & 0 \\ r21 & r22 & r23 & 0 \\ r31 & r32 & r33 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

NeXus

Combining Transformations



Some Properties

- Transformations can be combined by matrix multiplications
- Individual matrices can be derived by looking at the situation when everything else is 0
- Absolute positions can be obtained by multiplying the resulting matrix with its transpose
- Defines new coordinate systems at components
- CIF contains a duplication: vector, offset scheme

NeXus

What Use Is This?

- Allows to calculate absolute positions of components in the laboratory coordinate systems
- Can directly convert from a detector coordinate system to vectors in Lab coordinate system
- Calculate things like impact of primary beam on detector, SAS
- Allows arbitray axis to be expressed
- Intuitively describe an instrument with angles and translations and still be able to recover absolute coordinates

NeXus Axis Mapped

- rotation_angle, polar_angle, rotate 0 1 0
- azimuthal_angle, rotate 0 0 1
- distance, translate 0 0 1
- chi, rotate 0 0 1
- phi rotate, 0 1 0
- NeXus polar coordinate system: rotate azimuthal_angle, rotate polar_angle, translate by distance

CIF Dependency Table

axis-id	type	equipment	dependson	vector	offset
gonio_phi	rotation	goniometer	.	1,0,0,	...
det_z	translation	detector	.	0,0,-1	0 0 0
det_y	translation	detector	det_z	0,1,0	0,0,0
det_x	translation	detector	det_y	1,0,0	0,0,0

NeXus