Meteorologisk institutt
met.no

Fimex Mini-Workshop

Usage of Fimex as library

24.09.2009

Outline

- General description
- Internals
- API: Usage scenarios
- API-use

What is Fimex

- Library to:
  - Read different gridded geospatial data-formats
  - Reproject/interpolate gridded data
  - Manipulate metadata
  - Extract subsets
  - Write different data-formats
- Currently only used by command-line program “fimex”

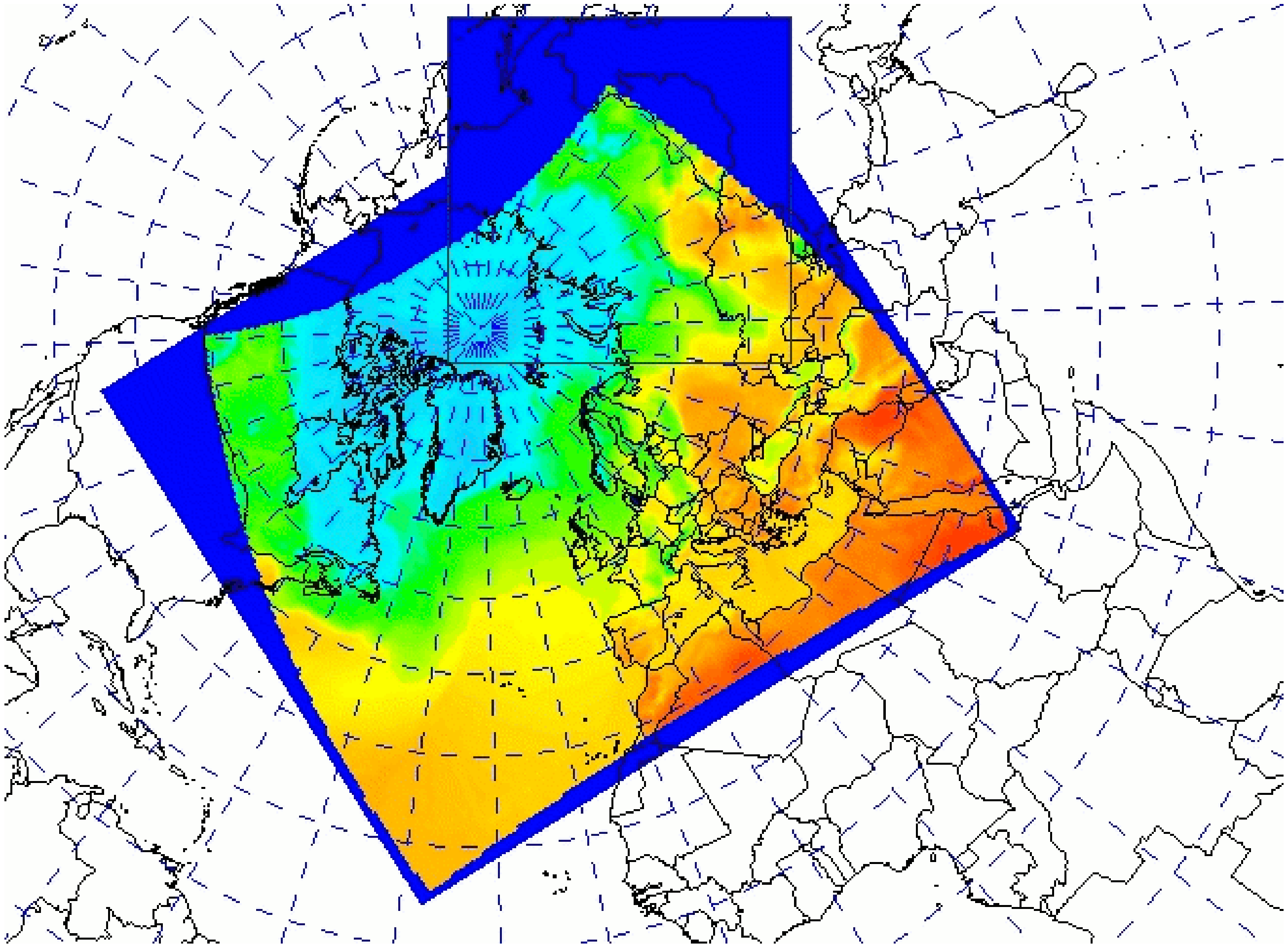
Status of functionality

- Supported I/O formats:
  - Netcdf 3,4
  - NcML (without aggregation)
  - Grib 1,2 (not in production yet)
  - Felt (reading) – C++ library by Vegar/Michael

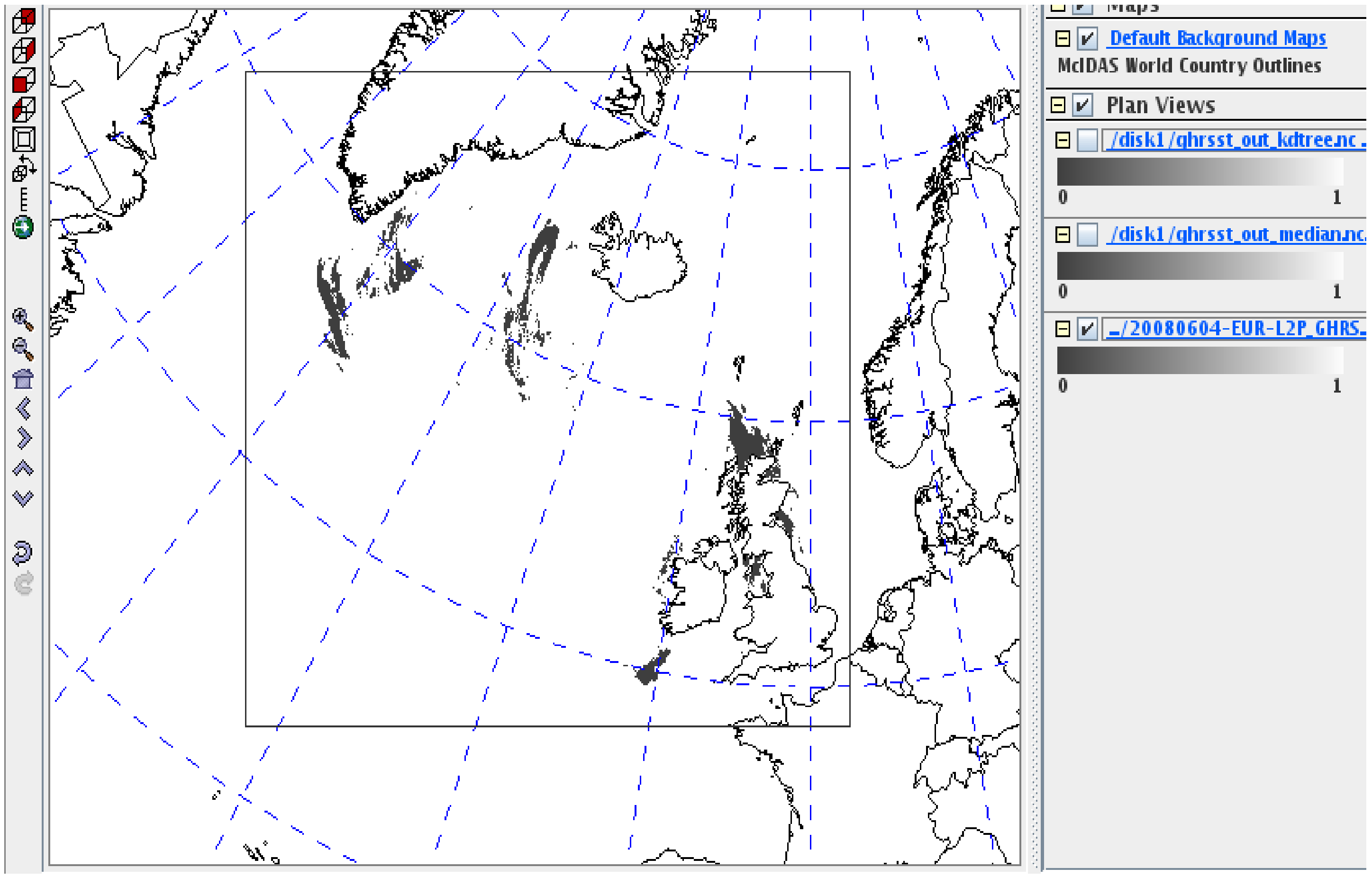
Status of functionality (cont.)

- Reprojection / Interpolation
  - Requires CF and/or proj4 description of projections
    - Support for ellipsoids (but not ellipsoid names)
  - Scalar and “vector along projection axes” reprojection
  - Forward and reverse interpolation using:
    - kD-tree, brute-force: for irregular data
    - next_neighbor, median, bilinear, bicubic: for regular data
  - Automatic determination of start- and end-position in output-projection
  - Linear time interpolation

Ex.: Bilinear interpolation: Hirlam



Ex.: Satellite, lat/long values without projection (GHRSST, ice-fraction)



Maps

Default Background Maps

McIDAS World Country Outlines

Plan Views

/disk1/ghrsst_out_kdtree.nc

0 1

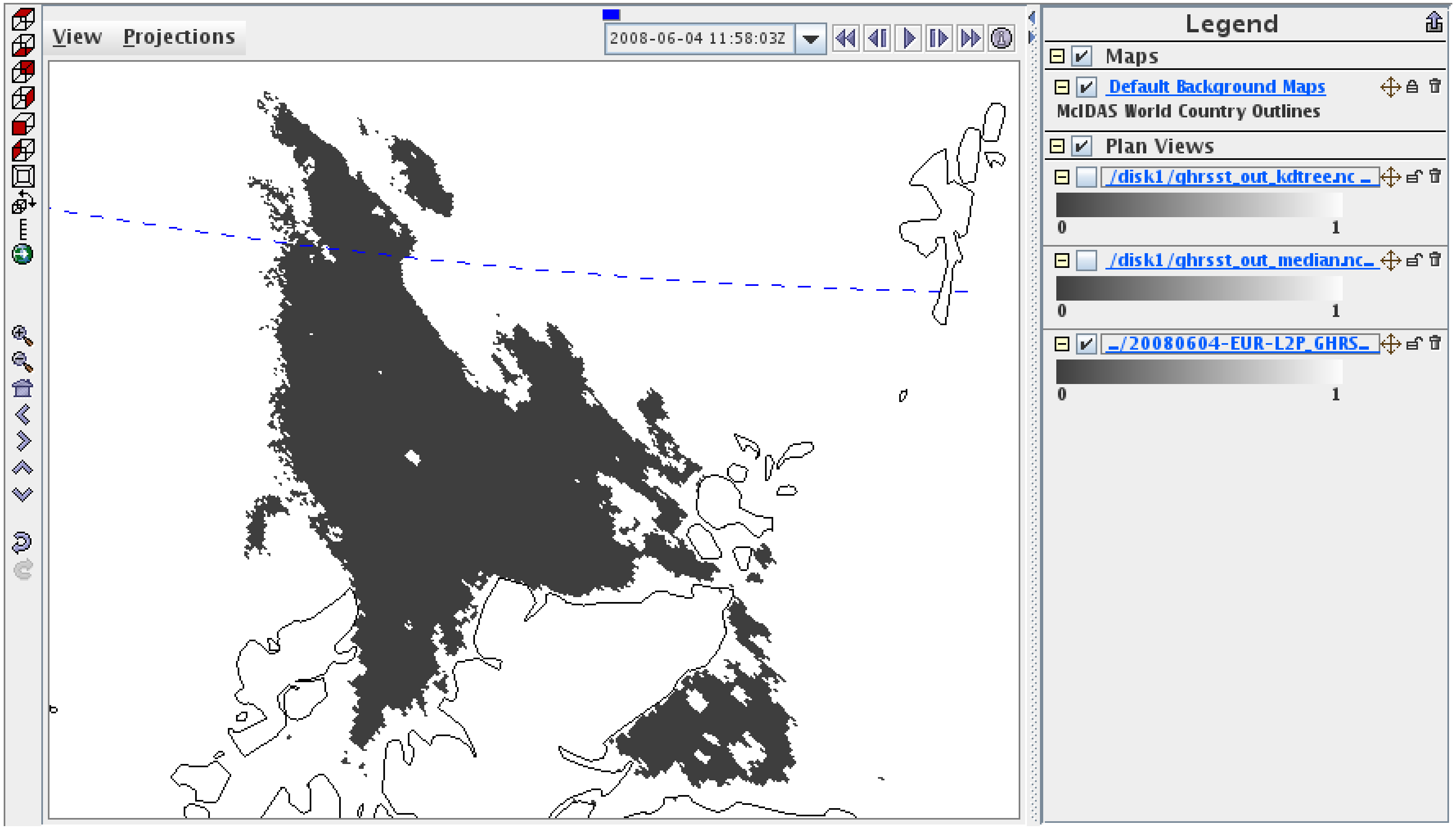
/disk1/ghrsst_out_median.nc

0 1

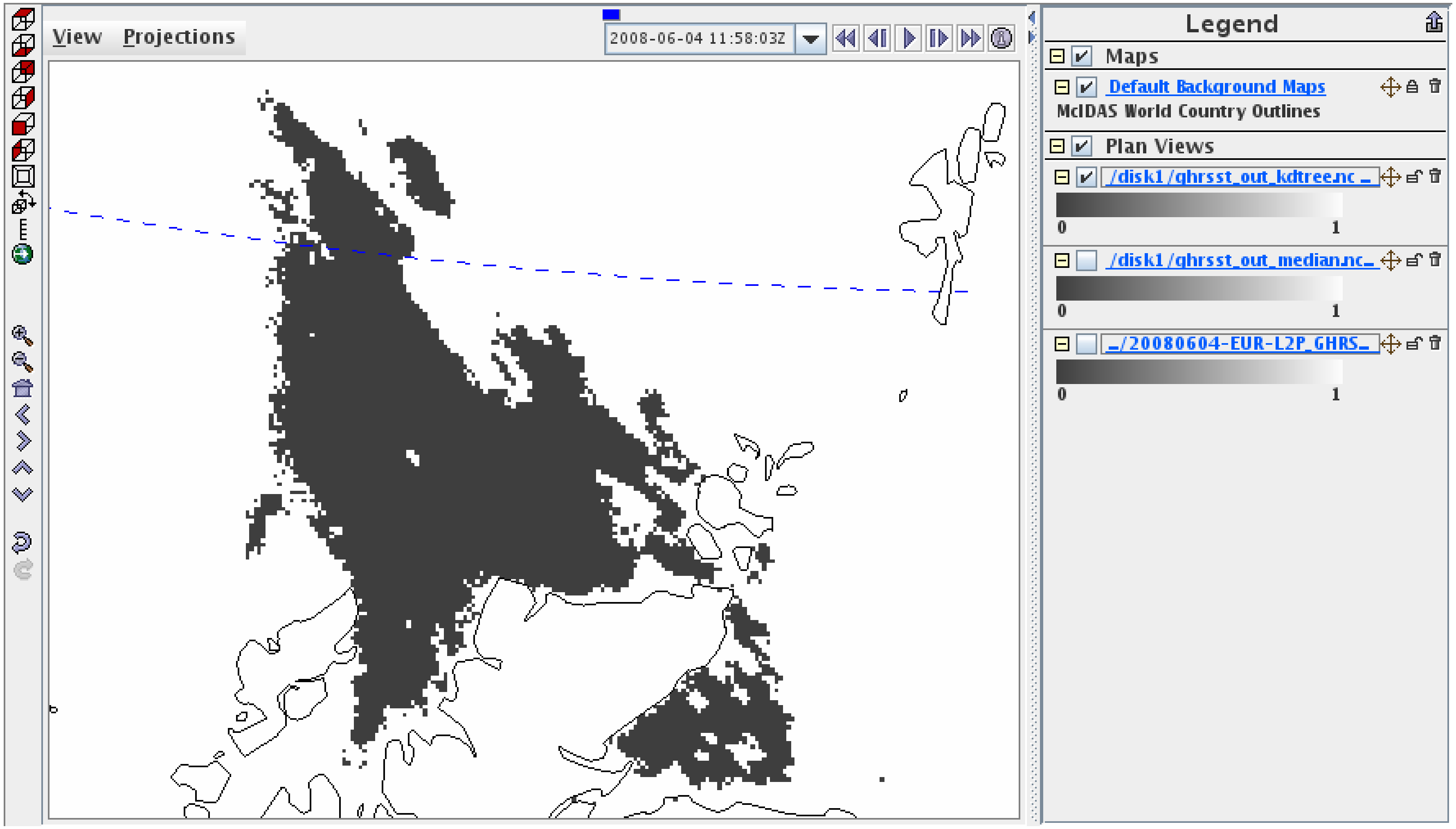
.../20080604-EUR-L2P_GHRS

0 1

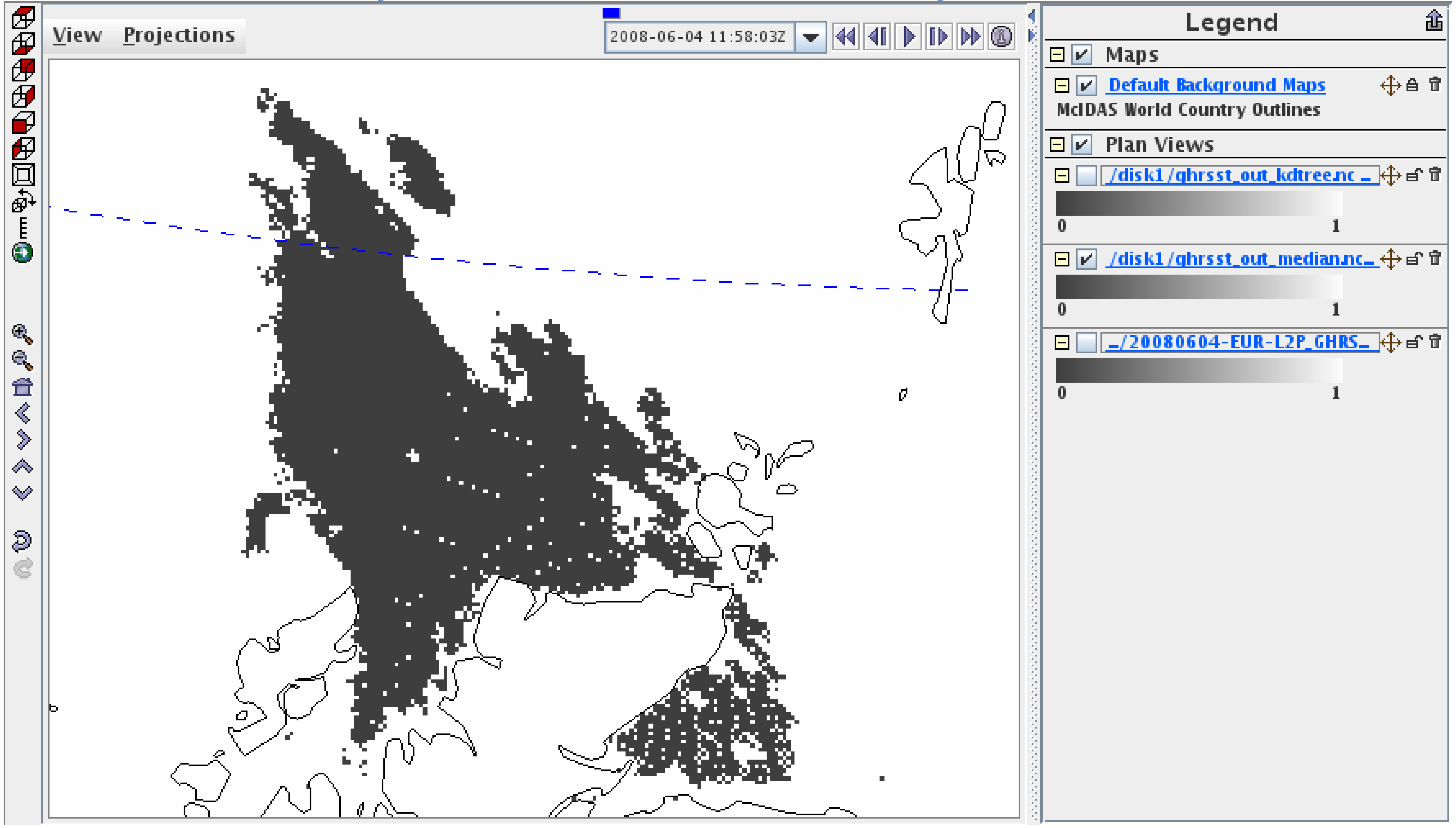
Ex.: Satellite (GHRSST, ice-fraction)



Ex.: Satellite (GHRSST, ice-fraction, kd-tree)



Ex.: Satellite (GHRSST, ice-fraction, forward) forward is mass-conservative, but leads to artifacts if input-resolution !<< output-res.



Setup for interpolation

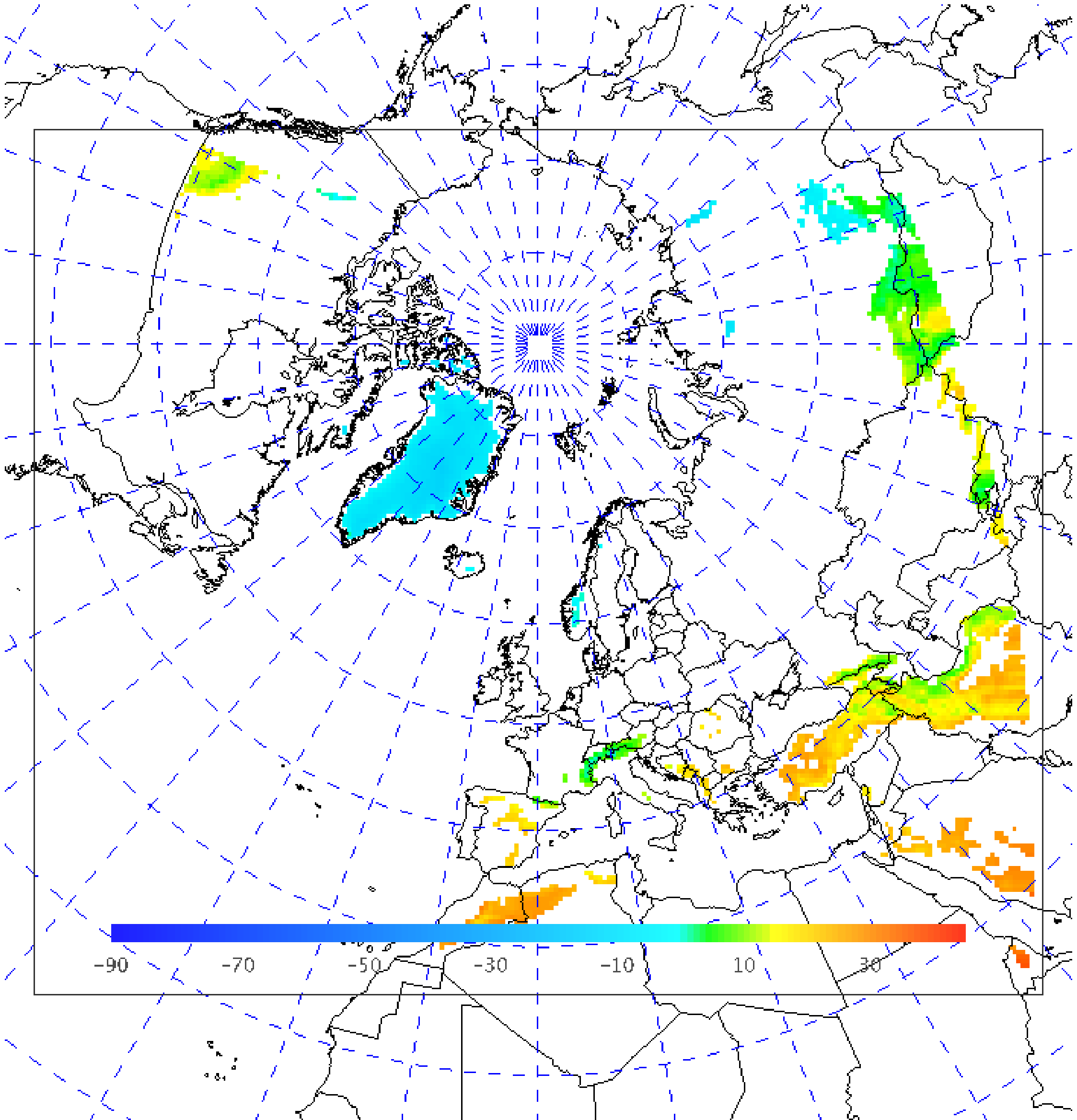
```
[input]
file=20080604-EUR-L2P.._1158.nc
[output]
file=ghrsst_out_kdtree.nc
[interpolate]
#method=forward_median
method=coord_kdtree
projString= +proj=stere +lat_0=90 +lon_0=0 +lat_ts=60
+elips=sphere +a=6371000 +e=0
xAxisValues=0,2000,...,x,x+2000;relativeStart=0
yAxisValues=0,2000,...,x,x+2000;relativeStart=0
xAxisUnit=m
yAxisUnit=m
```

Status of functionality (cont.)

- Add/Remove/Change Metadata Attributes
- Extract subsets
  - Remove variables
  - Extract reduced variables (time,level,region)
  - Extract values with constraints, e.g. val > 0; val only if otherval == 'ok' (quality)

Temperature above 1000m

Topography as Constraint for Temperature



Setup for Constraints

```
<variable name="bla">
  <status_flag_variable name="blub">
    <allowed_values>1,2,...,6</allowed_values>
    <!-- or config by highest valid or lowest valid or all valid values  -->
    <!-- highest and lowest will be retrieved per data-slice, not for the
whole variable -->
    <!-- <allowed_values use="(highest|lowest|all|min:xxx.x|
max:xxx.x)" /> -->
  </status_flag_variable>
</variable>
```

Setup for Constraints

```
<variable name="bla">
  <status_flag_variable name="blub">
    <allowed_values>1,2,...,6</allowed_values>
    <!-- or config by highest valid or lowest valid or all valid values  -->
    <!-- highest and lowest will be retrieved per data-slice, not for the
whole variable -->
    <!-- <allowed_values use="(highest|lowest|all|min:xxx.x|
max:xxx.x)" /> -->
  </status_flag_variable>
</variable>
```

```
<variable name="air_temperature">
  <status_flag_variable name="altitude">
    <allowed_values use="min:1000" />
  </status_flag_variable>
</variable>
```

Missing functionality?

- Aggregation
- check flt2flt functionality
- smoothing (check with diana)

Internals

Split between

- multidimensional arrays “Data”
- additional information “Metadata”

Data

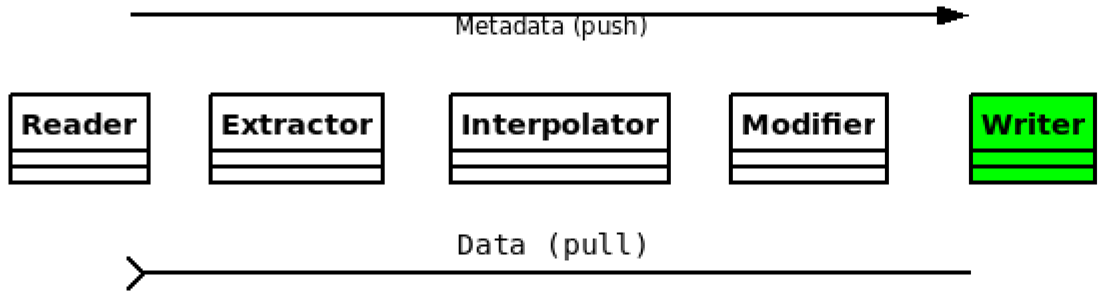
2.4	3.4	1.2	40.
45	55.3	45.	.45
.42	0.0	4.9	93.
9.	45.4	87.	944.
86.	.45	192.	3.3
8.6	4.2	22.2	18.2
0.3	8.2	8.4	2.3

Metadata

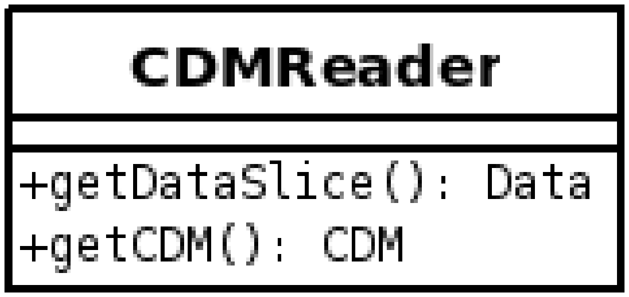
- Temperature
- Unit
- Projection
- Dimensions
- Producer

Internals (cont.)

- Chain of responsibilities



- One interface for all parts



Usage of chain

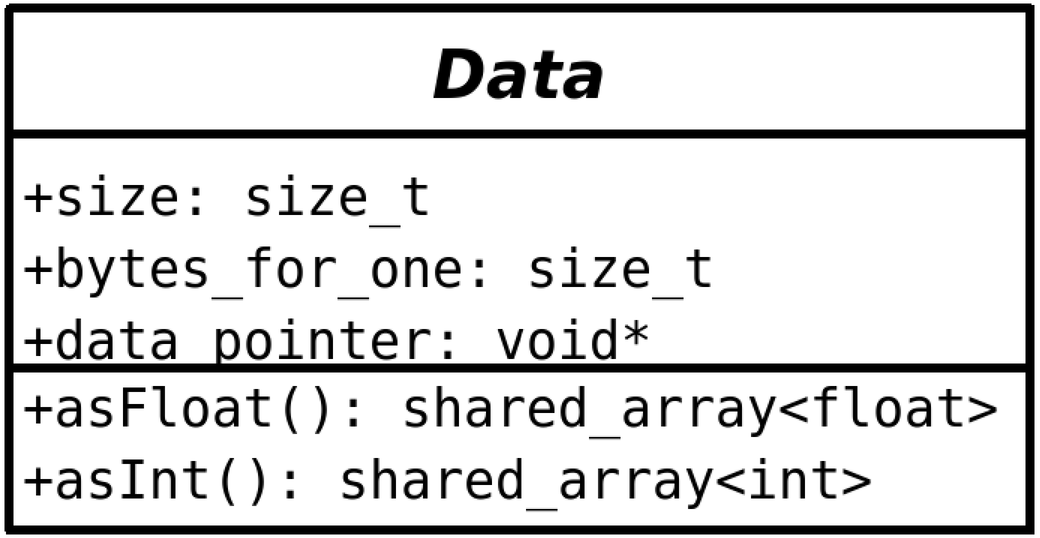
```
b::sh_ptr<CDMReader> feltReader(
    new FeltCDMReader("flth00.dat","feltsetup.xml"));
b::sh_ptr<CDMReader> qualityExtractor(
    new CDMQualityExtractor(feltReader,"qualtiy.xml"));
b::sh_ptr<CDMInterpolator> interpolator(
    new CDMInterpolator(qualityExtractor));
interpolator->changeProjection(MIUP_BILINEAR, "+proj=stere ...",
                               xAxis, yAxis, "m", "m");
b::sh_ptr<CDMExtractor> extractor(
    new CDMExtractor(interpolator));
extractor->removeVariable("sea_ice");

NetCDF_CDMWriter(extractor, "test.nc");
```

Usage of chain in perl?

```
# not all of this exists yet
use Fimex;

my $fltReader = new FeltReader("flth00.dat","feltsetup.xml");
my $qA = new QualityExtractor($fltReader, "qualtiy.xml");
my $interpol = new Interpolator($qA);
$interpol->changeProjection(...);
my $ex = new Extractor($interpol);
$ex->removeVariable("sea_ice");

new Netcdf_Writer($ex, "test.nc");
```

Data

- Abstract class, implementations by templates: DataImpl<double>, DataImpl<int>,...



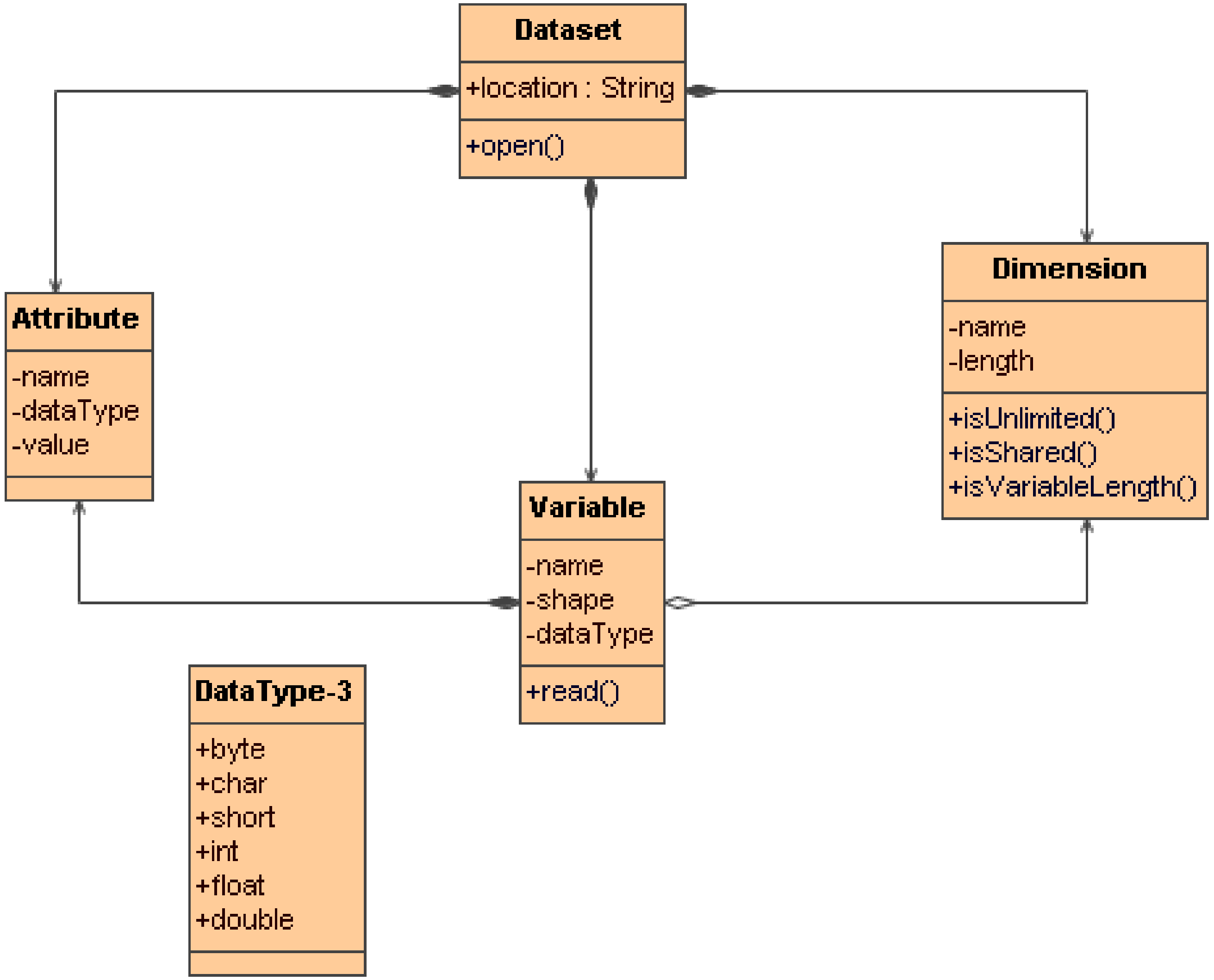
```
Data

+size: size_t
+bytes_for_one: size_t
+data_pointer: void*

+asFloat(): shared_array<float>
+asInt(): shared_array<int>
```

Data (cont.)

- Data usually read as (dim-1) slice, i.e. (x,y,z,t) read as (x,y,z) – 'Unlimited dimension'
- Additional helper-classes for slicing exist
- Original datatype stored in RTTI, and as part of the Metadata, but should not be needed?

Interna: Metadata

- ‘common data model’ CDM-1 – think netcdf-file:



Interna: Metadata standards

Stick to CF-1.x as much as possible

- CF-1.x projections and dimension attributes required for interpolation
- Netcdf attributes required for scaling (scale_offset, add_offset, _FillValue)
- Extraction and Netcdf-Writer don't require any metadata-standards

Requirements

- c/c++
- boost
- proj4
- Libxml2
- udunits/udunits2

Optional:

- grib-api
- netcdf

Example: fimex for OSISAF

- Metamod
  1) Search dataset: osisaf
  2) Change dataset
     - ole@somewhere.com

Usage Scenarios: Tune and reorder chain

- Avoid artifacts with forward interpolation:
  - interpolate to higher resolution (supersampling)
  - forward interpolate to other projection
- Remove variables at several steps
  - Remove never used variables
  - Quality extract data
  - Remove quality-status variable

Languages: C++, C?, Perl?

Usg. Scen.: Seamless integration of chain into environment

- No system-calls
  - Better error handling
  - Higher performance

  Languages: PHP, Perl, Java, C, C++

Usg. Scen.: General data access library

- Writing program being able to read different data-formats/projections in a unique way
  - Full metadata/CDM required (difficult for non-C++)? Or enough with a basic set?
  - Performance requirements <=> getDataslice() reads always (dim-1) data, generally no pre-selection possible
  - Do we need a special CoordinateSystem class?

Languages: C++, C, PerlDL?

More Scenarios ...

- Read from WDB