

Partial 3D data support

Motivation:	Stepping up from a pure 2d world to partial 3d support, the OGR,PostGIS,MapServer way
Contact:	Andrea Aime
Tracker:	http://jira.codehaus.org/browse/GEOT-2537
Tagline:	Handling 3d data in a 2d world

This page represents the **current** plan; for discussion please check the tracker link above.

- [Description](#)
- [Previous discussion](#)
- [Status](#)
 - [Tasks](#)
 - [API Changes](#)
 - [Documentation Changes](#)

Description

At the time of writing GeoTools is aggressively making sure only 2 dimensional data enters feature geometries. In particular, in most data sources that might have a 3D component the code is flattening the coordinates so that only the horizontal component is provided.

Coordinate reference systems are always 2 dimensional, too. Finally, only 2 dimensional geometry types are handled (JTS types).

On the other side of the 3D support spectrum there are 3 or more dimensional coordinate reference systems (fully 3D, or compound), 3D coordinates and solid geometries (such as spheres, cones, boxes).

This proposal attempts to bring some 3D support into GeoTools, to the extent that is already covered by various other software.

In particular, the geometry coordinates will be allowed three (or more) dimensions, whilst the coordinate reference system used will still be two dimensional. This approach is already followed by various software around:

- PostGIS, which only handles two dimensional SRS but has an explicit coord_dimension column to declare the actual dimension of the coordinates
- GML3 posList element, which has at the same time a srsName and a srsDimension attribute, allowing the two to be out of synch. Various CityGML examples do in fact sport a 2D SRS (or no SRS at all) but explicitly set srsDimension="3"
- shapefiles generated by ESRI software typically have a 2D WKT definition in the shapefile.prj file whilst the shapefile nature is pointz or arcz
- Oracle Spatial previous to 11G, and even 11G without explicit settings, allow 3D data to be handled whilst the spatial reference system used is by dimensional.

All the above systems make no different between a Z and a M (measure), basically handling every dimension past the first two as sort of out of band information that does not get transformed when the first two dimensions are. In particular, coordinate system transformations do change the planar part whilst the extra dimensions are carried over unaltered.

The proposed changes to the GeoTools code base are relatively small:

- allow data sources that happen to have non flat data to return it as is, unless the existing Hints.FEATURE_2D is passed down (in which case, the geometries will be flattened as today)
- keep the current flat CoordinateReferenceSystem in the GeometryDescriptor, and add a well known COORDINATE_DIMENSION entry in its userData map so that the coordinate dimensionality is explicit even in absence of data (for code that needs to deal with metadata only)
- leverage the CoordinateSequence.getDimension() when only the geometry is available (this might mean we'll need to subclass the default JTS coordinate sequence, CoordinateArraySequence, as that has a hard coded dimension of 3, so that we can report the actual dimension, or else, to check the Z of all coordinates in it and decide the real dimensionality based on the values of the Z, 2d if the Z is always a NaN, 3d if there is at least one non NaN value)
- extend ReferencedEnvelope to support more than 2 dimensions
- modify the geometry and envelope transformation code so that only the first two dimensions are transformed, whilst the other dimensions are carried along unaltered

Previous discussion

The topic has already been publicly discussed on the mailing list in this thread: <http://n2.nabble.com/Handling-2D%2B1-data-in-GeoTools-td2739186.html#a2739186>

Status

This proposal is under construction.

Voting has not started yet:

- [Andrea Aime](#) +1
- [Ben Caradoc-Davies](#): +1
- [Christian Mueller](#) +1
- [Ian Turton](#)
- [Justin Deoliveira](#) +1
- [Jody Garnett](#) (current OSGeo representative)
- [Michael Bedward](#) +0
- [Simone Giannecchini](#)

Tasks

This section is used to make sure your proposal is complete (did you remember documentation?) and has enough paid or volunteer time lined up to be a success

	no progress		done		impeded		lack mandate /funds/time		volunteer needed
--	-------------	---	------	---	---------	---	--------------------------	---	------------------

1. declare the COORDINATE_DIMENSION hint in Hints (aaime)
2. modify the PostGIS datastore to add the hint in the feature types built, avoid flattening data while reading by using EWKB format, and add support for the FEATURE_2D parameter (aaime)
3. change the rendering code to force 2D coordinates when reading (aaime)
4. change the geometry transformation code to carry on the third dimension (aaime)
5. update the docs

6. upgrade the shapefile datastore ?
7. upgrade the jdbc-ng datastores ?

Updating the datastores to allow returning the third dimension is an activity that can be performed in time, a datastore that keeps behaving as before won't be broken by the changes and won't affect client code either (it will just miss the ability to return multidimensional data).

API Changes

Since access to coordinate dimensions is explicit and JTS always offered a Z in the Coordinate class, no real change in the user code is needed. Only, if certain code absolutely needs 2D data, it will have to pass down the FEATURE_2D hint, like this:

```
FeatureSource fs = ...;
    DefaultQuery q = new
DefaultQuery("my3dType");
    q.setHints(new Hints(Hints.FEATURE_2D,
true));
    FeatureCollection coll =
fs.getFeatures(q);
```

Also, it is expected ReferencedEnvelope to gain support for further dimensions. The interface is already n-dimensional capable, but fields will need to be added to support dimensions over the first two.

Documentation Changes

- [06 FeatureSource](#) should be modified, and an example of using FEATURE_2D should be added
- A new page should be added with a description of how the 3D support is implemented in GeoTools