Coordinate System Transformations in Geotools and uDig

Jan Ježek

Czech Technical University in Prague
Faculty of Civi Engineering
Department of Mapping and Cartography

September 22, 2007

Outline of Part I

1. Google Summer of Code
   - Google Summer of Code - basic informations
   - Google Summer of Code and FOSS4G

Outline of Part II

2. Coordinate Transformations for GeoTools and uDig
   - Transformations builders
   - Transformations methods and algorithms
   - uDig plug-in
   - Use cases

What is GSoC

- Google Summer of CodeTM is a program that offers student developers stipends to write code for various open source projects. Google will be working with a several open source, free software and technology-related groups to identify and fund several projects over a three month period. Historically, the program has brought together over 1,000 students with over 100 open source projects, to create hundreds of thousands of lines of code. The program, which kicked off in 2005, is now in its third year, following on from a very successful 2006.

How does it work

- GSoC usually starts during the spring.
- Mentoring organizations delegates their candidate that should participate.
- Students can write a project proposal on an idea related to the needs of mentoring organization, but also any kind of new idea is welcomed.
- The proposals are then checked by mentors and the chart of received projects is made.
- Google then decides how many projects will be funded for each organization.

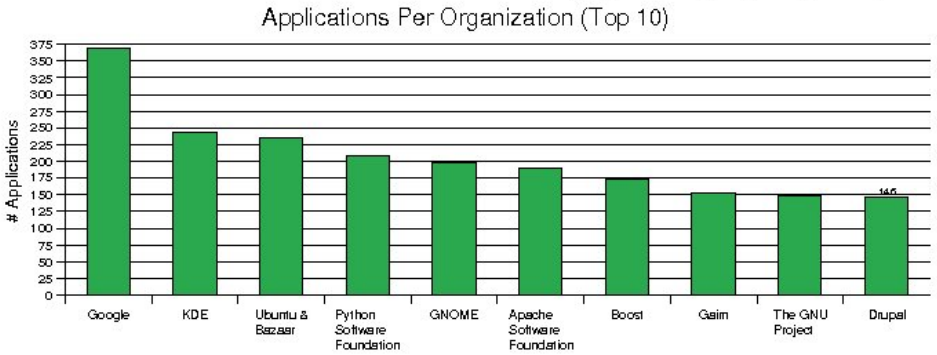
How does it work

- The real coding starts in June. The process is divided into three periods. At first students receive 500USD after their proposal is accepted.
- After approximately one month of coding their mentors are asked by Google to write the mid-term evaluation of the process that had been done. If their evaluation is positive, the student receives a 2000 USD check.
- The programs end in September, when a final evaluation is made and the student receives the last 2000 USD if they succeed.

GSoC Statistics - 2006

- 6338 Applications
- 3044 Applicants
- 1260 Mentors
- 102 Open Source Organizations
- 90 Countries
- 630 Applications accepted

Application Organizational Distribution



	Applications Per Org.
Min:	3
Max:	370
Sum:	6338
Mean:	62.14
Median:	42.5
Stdev:	59.95

	Applications Per Org.
Min:	3
Max:	370
Sum:	6338
Mean:	62.14
Median:	42.5
Stdev:	59.95

Note: Google acted as an "other" category and catch all for lower quality applications.

GSoC 2006

GSoC Statistics - 2007

- 130 Open Source Organizations
- 900 Applications accepted
- 1500 Mentors

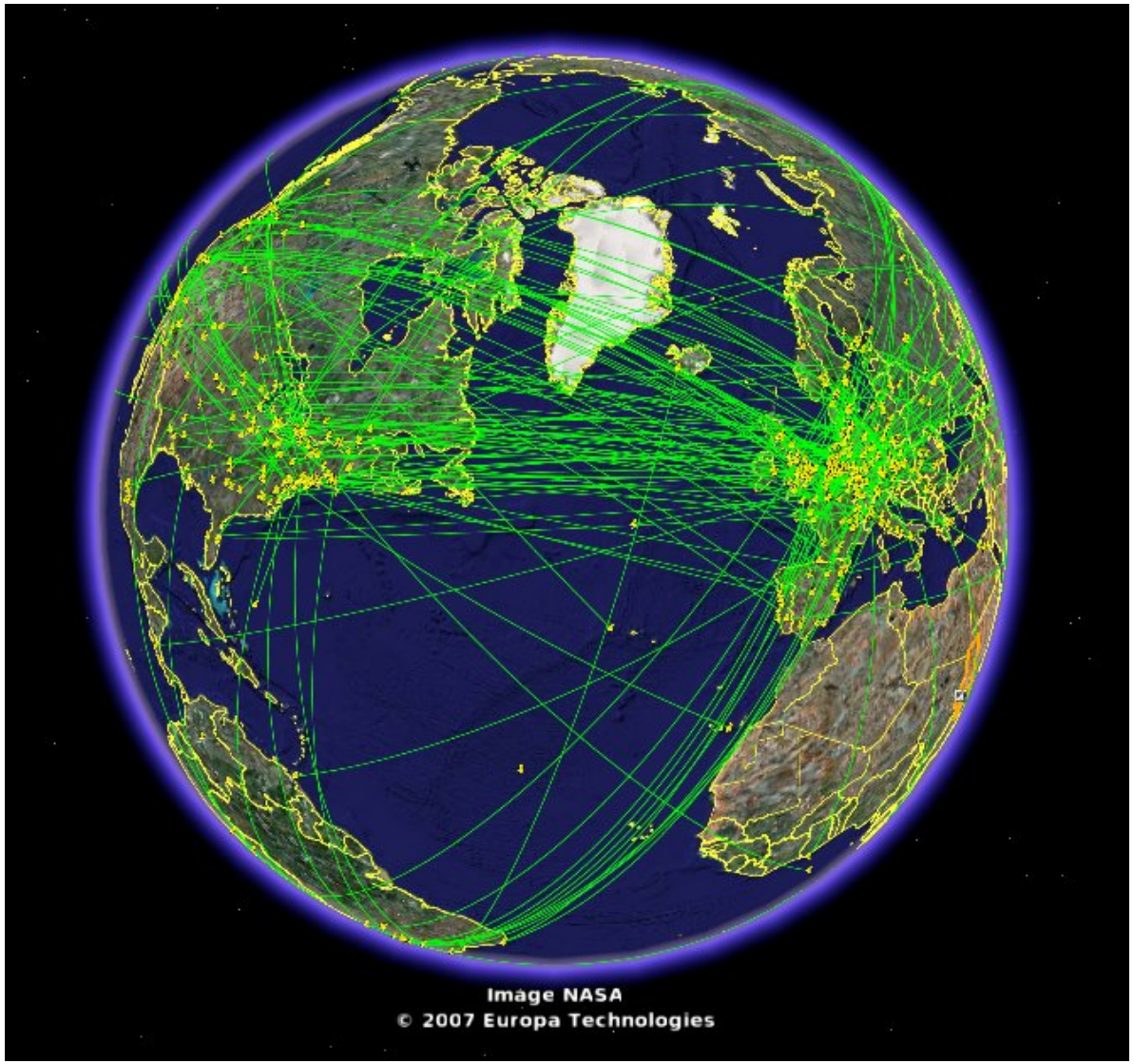


Image NASA
© 2007 Europa Technologies

GSoC - FOSS4G - 2006

Mentoring organization - Refractions Research

- 3 projects
- GDAL ImageIO integration (by Daniele Romagnoli, mentored by Simone Giannecchini)
- uDIG GPS Record Import and Spatial Report Processing (by Dan Eslinger, mentored by Cory Horner)
- Coordinate System Transformations (by Jan Jezek, mentored by Jesse Eicher)

GSoC - FOSS4G - 2006

Mentoring organization - Refractions Research

- 3 projects
- GDAL ImageIO integration (by Daniele Romagnoli, mentored by Simone Giannecchini)
- uDIG GPS Record Import and Spatial Report Processing (by Dan Eslinger, mentored by Cory Horner)
- Coordinate System Transformations (by Jan Jezek, mentored by Jesse Eicher)

GSoC - FOSS4G - 2006

Mentoring organization - Refractions Research

- 3 projects
- GDAL ImageIO integration (by Daniele Romagnoli, mentored by Simone Giannecchini)
- uDIG GPS Record Import and Spatial Report Processing (by Dan Eslinger, mentored by Cory Horner)
- Coordinate System Transformations (by Jan Jezek, mentored by Jesse Eicher)

GSoC - FOSS4G - 2006

Mentoring organization - Refractions Research

- 3 projects
- GDAL ImageIO integration (by Daniele Romagnoli, mentored by Simone Giannecchini)
- uDIG GPS Record Import and Spatial Report Processing (by Dan Eslinger, mentored by Cory Horner)
- Coordinate System Transformations (by Jan Jezek, mentored by Jesse Eicher)

GSoC - FOSS4G - 2007

Mentoring organization - OSGeo

- 12 projects accepted
- ?? applications

- GRASS Modules for line generalization and smoothing by Daniel Bundala, mentored by Wolf Björn Mikael Bergenheim
- GDAL2Tiles - Utility for easy tile-based publishing of raster maps and KML SuperOverlay by Petr Přidal, mentored by Howard Butler
- Plugins for multidimensional raster data sources. by Daniele Romagnoli, mentored by Simone Giannecchini
- Caching data in uDig by Christophe ROUSSON, mentored by Ian Turton
- JTileCache by Christopher Whitney, mentored by Justin Deoliveira
- Shortest path in free (vector) space avoiding obstacles module in GRASS by Maximilian Maldacker, mentored by Wolf Björn Mikael Bergenheim
- GDAL: KML read support for the existing driver by Jens Oberender, mentored by Mateusz Loskot

GSoC - FOSS4G - 2007

Mentoring organization - OSGeo

- 12 projects accepted
- ?? applications

- GRASS Modules for line generalization and smoothing by Daniel Bundala, mentored by Wolf Björn Mikael Bergenheim
- GDAL2Tiles - Utility for easy tile-based publishing of raster maps and KML SuperOverlay by Petr Přidal, mentored by Howard Butler
- Plugins for multidimensional raster data sources. by Daniele Romagnoli, mentored by Simone Giannecchini
- Caching data in uDig by Christophe ROUSSON, mentored by Ian Turton
- JTileCache by Christopher Whitney, mentored by Justin Deoliveira
- Shortest path in free (vector) space avoiding obstacles module in GRASS by Maximilian Maldacker, mentored by Wolf Björn Mikael Bergenheim
- GDAL: KML read support for the existing driver by Jens Oberender, mentored by Mateusz Loskot

GSoC - FOSS4G - 2007

- New Transformation Algorithms for GeoTools and uDig by Jan Jezek, mentored by Jesse Eichar
- Coverage model and operations for PostGIS by Xing Lin, mentored by Timothy H. Keitt
- GeoServer Style Editor by Anthony Manfredi, mentored by Tim Schaub
- Implementation of An Interactive GeoRSS tool in uDig by Rui Li, mentored by Richard Gould
- 3D Rendering Pipeline for GeoTools by Hans Häggström, mentored by Jody Garnett
- OGC WMS GDAL driver by Adam Nowacki, mentored by Daniel Morissette

Part II

Coordinate Transformations for GeoTools and uDig

One of the frequently required operations in GIS is to fit rasters like scans of maps or remote sensing images that have unknown coordinate reference system into the real world coordinate reference system. In GeoTools there are already few possibilities to do so. The aim of this project is to add other algorithms for that purpose and then to make a simple GUI for uDig to apply these new functions.

Demo !

Transformations builders = objects that builds a transformations from known mapped positions.
Mapped Position = known pair of source and target position.
Two possibilties:

- Linear transformations (using LSM)
- Warping transformations

Linear Tranformations Builders

- Similar Transformation builder
- Affine Transformation builder
- Projective Transformation builder
- Bursa Wolf Transformation builder

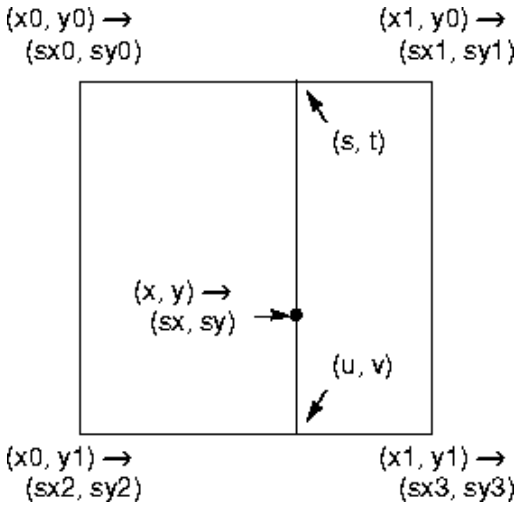
Code example

```
DirectPosition source = new DirectPosition2D(crs, x, y);
DirectPosition target = new DirectPosition2D(crs, x, y);

MappedPosition mp = new MappedPosition(source, target);
  ......
List /*<MappedPositions>*/ vectors = new ArrayList();
vectors.add(mp),

MathTransformBuilder builder = new ProjectiveTransformBuilder(vectors);
MathTransform trans = builder.getMathTransform();
```

Warping Tranformations Builders

Warping transformation = no differences between known source points after transformation and target points.
To build the transformation we have to do these steps:

- From irregular pairs of mapped positions generate regular grid of shifts (using some kind of interpolation)
- Generated shifts are applyed on datasets.



Warping Transformations Builders

- TPS Transformation builder
- IDW Transformation builder
- Rubber Sheet Transformation builder

Warping Transformations

- WarpGridTranasform2D (extends WarpTransform2D) - encapsulates JAI WarpGrid
- RubberSheetTransform

Inverse distance weigted

Inverse distance weighting (IDW) is a method for multivariate interpolation, a process of assigning values to unknown points by using values from known points. A simple IDW weighting function, as defined by Shepard[1], is:

$$w(d) = \frac{1}{d^p}$$

,

where w(d) is the weighting factor applied to a known value, d is the distance from the known value to the unknown value, and p is a positive real number, called the power parameter. Here weight decreases as distance increases from the interpolated points. Greater values of p assign greater influence to values closest to the interpolated point. The most common value of p is 2.

Inverse distance weighted interpolation

A general form of interpolating a value using IDW is:

$$Z = \frac{\sum_{i=1}^{N} \frac{Z_i}{d_i^p}}{\sum_{i=1}^{N} \frac{1}{d_i^p}}$$

where Z is the value of the interpolated point, Zi is a known value, and N is the total number of known points used in interpolation.

Thin plate spline

Thin plate splines (TPS) were introduced to geometric design by Duchon (Duchon, 1976). The name thin plate spline refers to a physical analogy involving the bending of a thin sheet of metal. In the physical setting, the deflection is in the z direction, orthogonal to the plane. In order to apply this idea to the problem of coordinate transformation, one interprets the lifting of the plate as a displacement of the x or y coordinates within the plane. In 2D cases, given a set of K corresponding points, the TPS warp is described by $2(K + 3)$ parameters which include 6 global affine motion parameters and 2K coefficients for correspondences of the control points. These parameters are computed by solving a linear system, in other words, TPS has close-form solution.

Thin plate spline

Thin plate splines (TPS) were introduced to geometric design by Duchon (Duchon, 1976). The name thin plate spline refers to a physical analogy involving the bending of a thin sheet of metal. In the physical setting, the deflection is in the z direction, orthogonal to the plane. In order to apply this idea to the problem of coordinate transformation, one interprets the lifting of the plate as a displacement of the x or y coordinates within the plane. In 2D cases, given a set of K corresponding points, the TPS warp is described by $2(K + 3)$ parameters which include 6 global affine motion parameters and 2K coefficients for correspondences of the control points. These parameters are computed by solving a linear system, in other words, TPS has close-form solution.

Thin plate spline

$$E = \iint \left[\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial xy} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right] \mathrm{d}x \, \mathrm{d}y$$

And for a smoothing TPS, it is

$$E_{tps} = \sum_{i=1}^{K} \|y_i - f(x_i)\| + \lambda \iint \left[\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial xy} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right] \mathrm{d}x \, \mathrm{d}y$$

Then smoothing TPS is defined as

$$f_{tps} = \arg\min_f E_{tps}$$

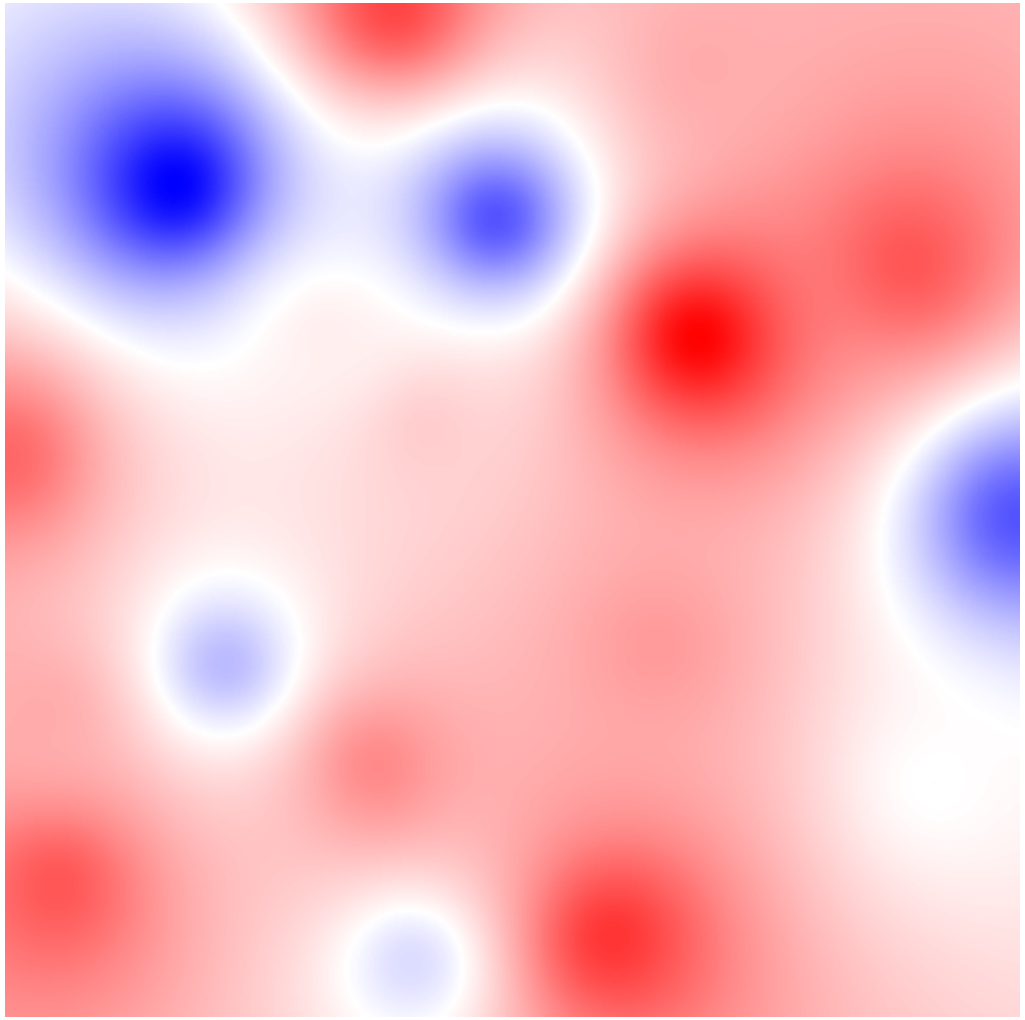
Rubber Sheeting

- First step The bounding box of the source dataset is divided into triangles where the specified points are the vertices of this triangles. The delaunay triangulation has been implemented for this. These triangles are mapped to the triangles made by destination points.
- Second step There is affine transformation applied on each triangle. The result is that the points on the edge of the triangle are calculated twice, but the result is the same because of the properties of affinity.

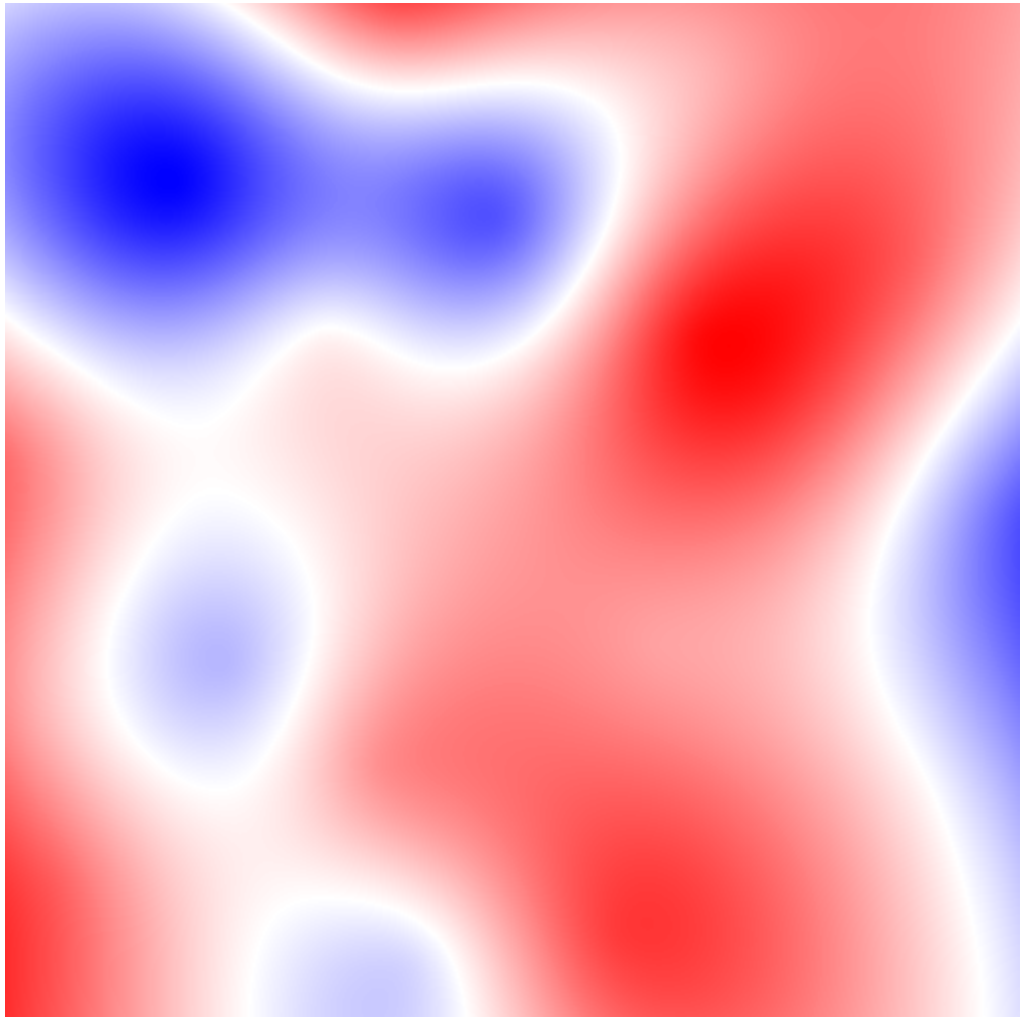
Rubber Sheeting

- First step The bounding box of the source dataset is divided into triangles where the specified points are the vertices of this triangles. The delaunay triangulation has been implemented for this. These triangles are mapped to the triangles made by destination points.
- Second step There is affine transformation applied on each triangle. The result is that the points on the edge of the triangle are calculated twice, but the result is the same because of the properties of affinity.

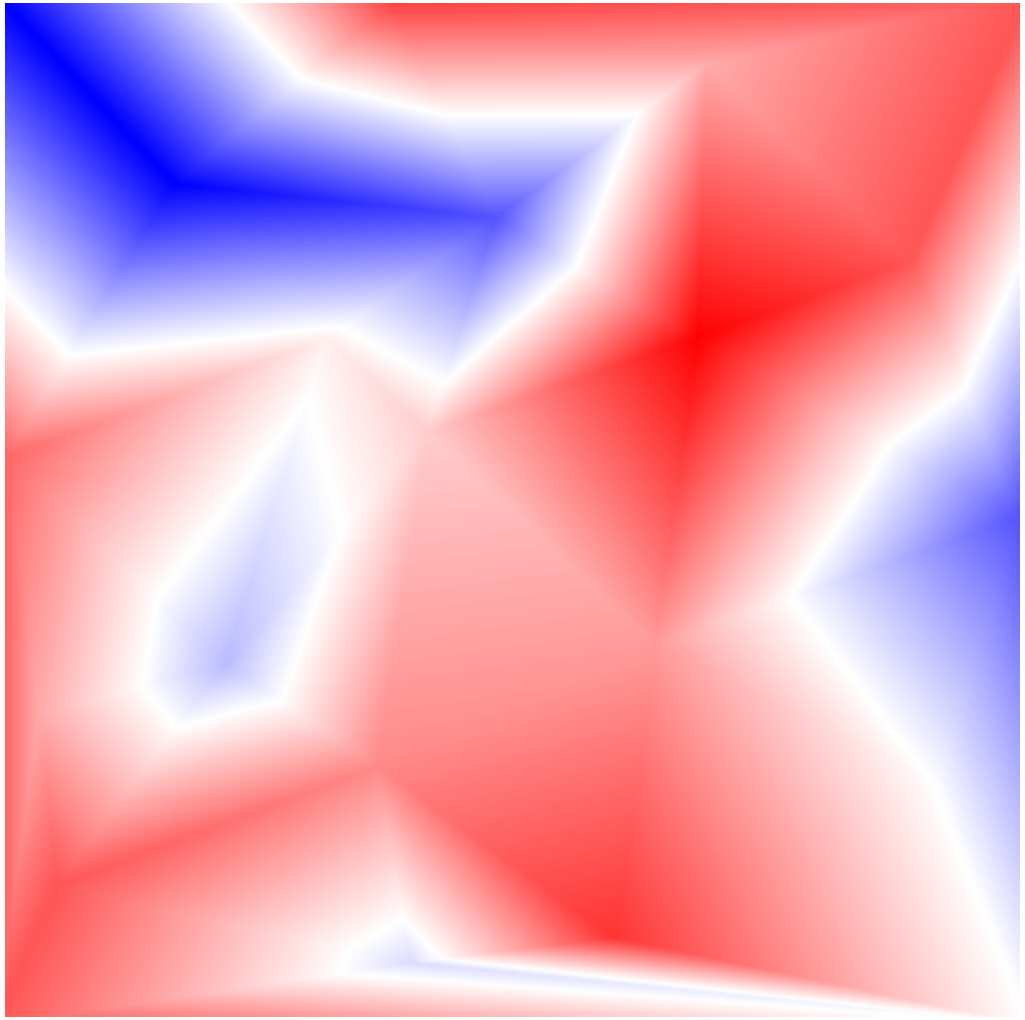
Inverse distance weigted interpolation



Thin plate spline



Rubber sheeting interpolation



Standalone interpolation - code example

```
Envelope env = new GeneralEnvelope(.....);

// Lets Generate some known points that will define interpolation
Map /*<DirectPosition2D, Float>*/ pointsAndValues = new HashMap();

pointsAndValues.put(new DirectPosition2D(crs, 130, 805), 6.5);
            ...
//now we can construct the Interpolation Object
AbstractInterpolation interpolation =
              new TPSInterpolation(pointsAndValues, 2, 2, env);

// we can get and show coverage
(new GridCoverageFactory()).create("Intepolated Coverage",
 interpolation.get2DGrid(), env).show();
```

uDig plug-in

- Tools for creating mapped positions
- Tools to choose proper transformation
- Possibility to do interpolation settings
- Tools to perform transformation (Rasters as well as vectors).

Use cases

- Rectification
- Improvement of existing global transformation

Use cases

WGS84 → Local systems

- WGS84 $\xrightarrow{B.W.Transform.}$ local system - cca 2 m occuracy
- Calculation of nad2nad files

Thanks Jesse Eicher, thanks Martin Desruisseaux, thanks Google, thanks community!

I've (l)earned a lot!!

Thanks for your attention!