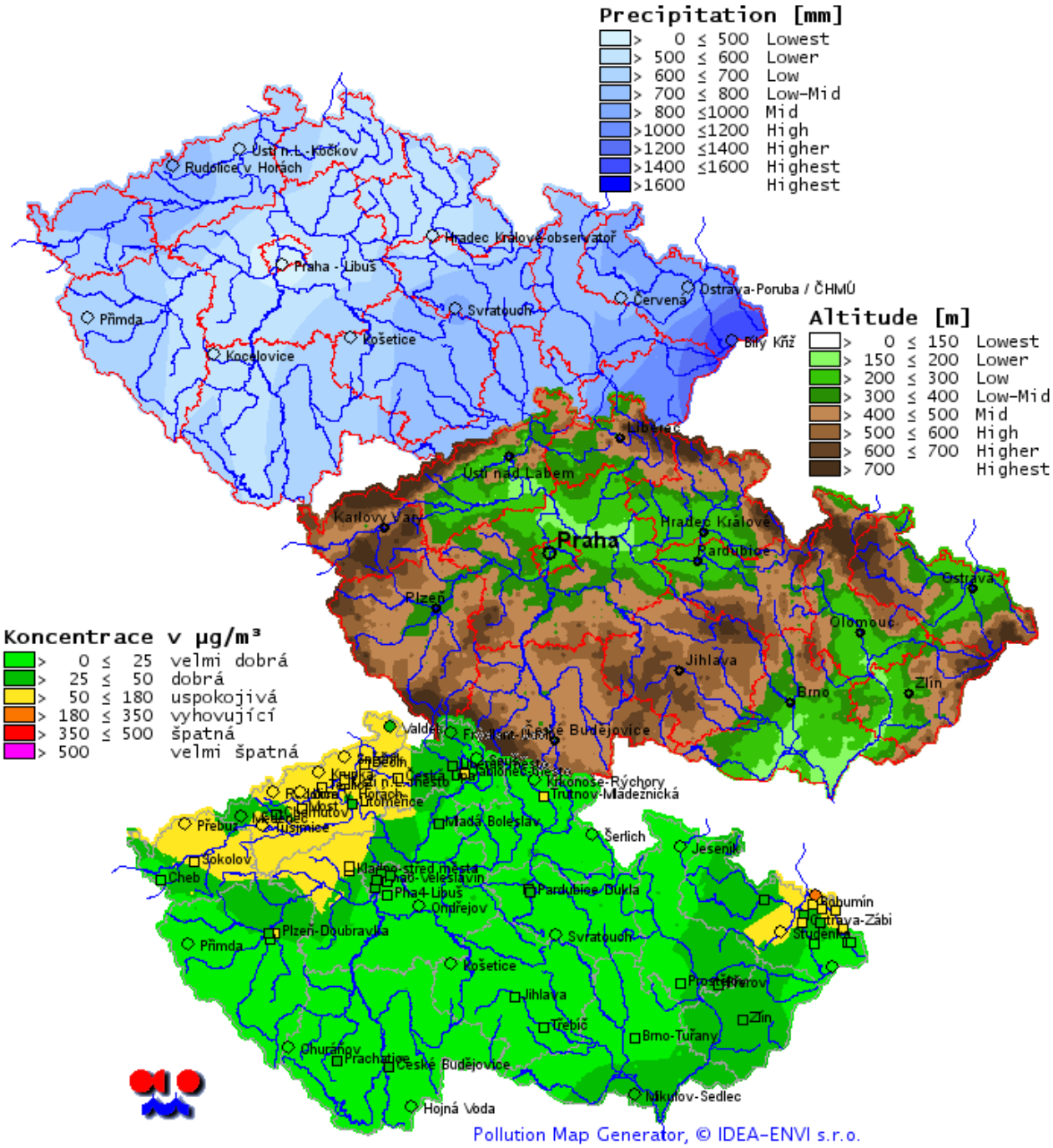
Map Generator

User Guide

IDEA-ENVI

Integrated development of applications



Author: Lumír Vaněk, vanek@idea-envi.cz

www.idea-envi.cz
Last change 7.2.2013

Content

1.Introduction

Map Generator is intended for creating geographic statistic maps.

1.1. Basic terms

1.1.1. Systems of coordinates

Map Generator uses internally two-dimensional **Cartesian coordinate system**. System is unitless - used units does not matter. Only is required to define relation between coordinate system unit and pixel on bitmap (map scale).

Map Generator also can use two-dimensional **Polar coordinate system**.

Aditionally, other coordinate systems can be used, if they can be converted to two-dimensional Cartesian coordinate system, such as Gauss-Krüger S-42, UTM, WGS 84 or others.

1.1.2. Geographic statistics map

Geographic statistics map (next fort short **map**) is intended for graphic representation of certain value in map with color background, where backgroud color in given position represents intensity of value with regard to choosed color scale.

Maps contain color background, aditional informations (borders, cities…) and legend with color scale and value ranges, that represents particular background colors.

Map has following properties:

1. **Geographic position** of left bottom corner
2. **Scale** – how big area in geographic units (e.g. meters, feets, ...) is represented by one pixel in map image
3. Map image **size** in pixels

1.1.3. Quantity

Quantity is type of values, displayed in map – it can be any statistical value, such as percent of unemployment, measured concentration of substance, temperature, amount of precipitation etc.

1.1.4. Input data

Input data are primary values, whose processed to create map. If we want to create map, we need input data of our quantity.

Every datum of input data must have following informations:

- Geographic position of place, where value was obtained or measured

- Name of place, where value was obtained – for example name of the measuring station
- The obtained or measured value

1.1.5. Matrix

Matrix, a.k.a **grid** is two-dimensional value array, that is basic block of data processing in Map Generator. For our purposes we divide matrixes to:

1. **Input matrix** – contains prepared values, such as terrain altitudes, population density etc. It can be loaded from file or database
2. **Computed matrix** – contains values, that are result of interpolation of input values
3. **Result matrix** – contains values, that are result of modification of values in another matrixes

Matrix have following properties:

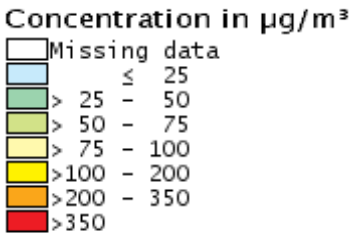
- Geographic position of left bottom cell
- Number of grid rows and columns
- Cell size – how big area in in geographic units (e.g. meters, feets, ...) is represented by one cell

1.1.6. Color scale

Color scale is used for conversion value to color. We recognize two basic types of color scales: **List scales** and **Linear scales**.

1.1.6.1 List color scale

This type of scale is ordered list of **value ranges** and corresponding **colors**. Last range is open – it not have high range value.



1.1.6.2 Linear color scale

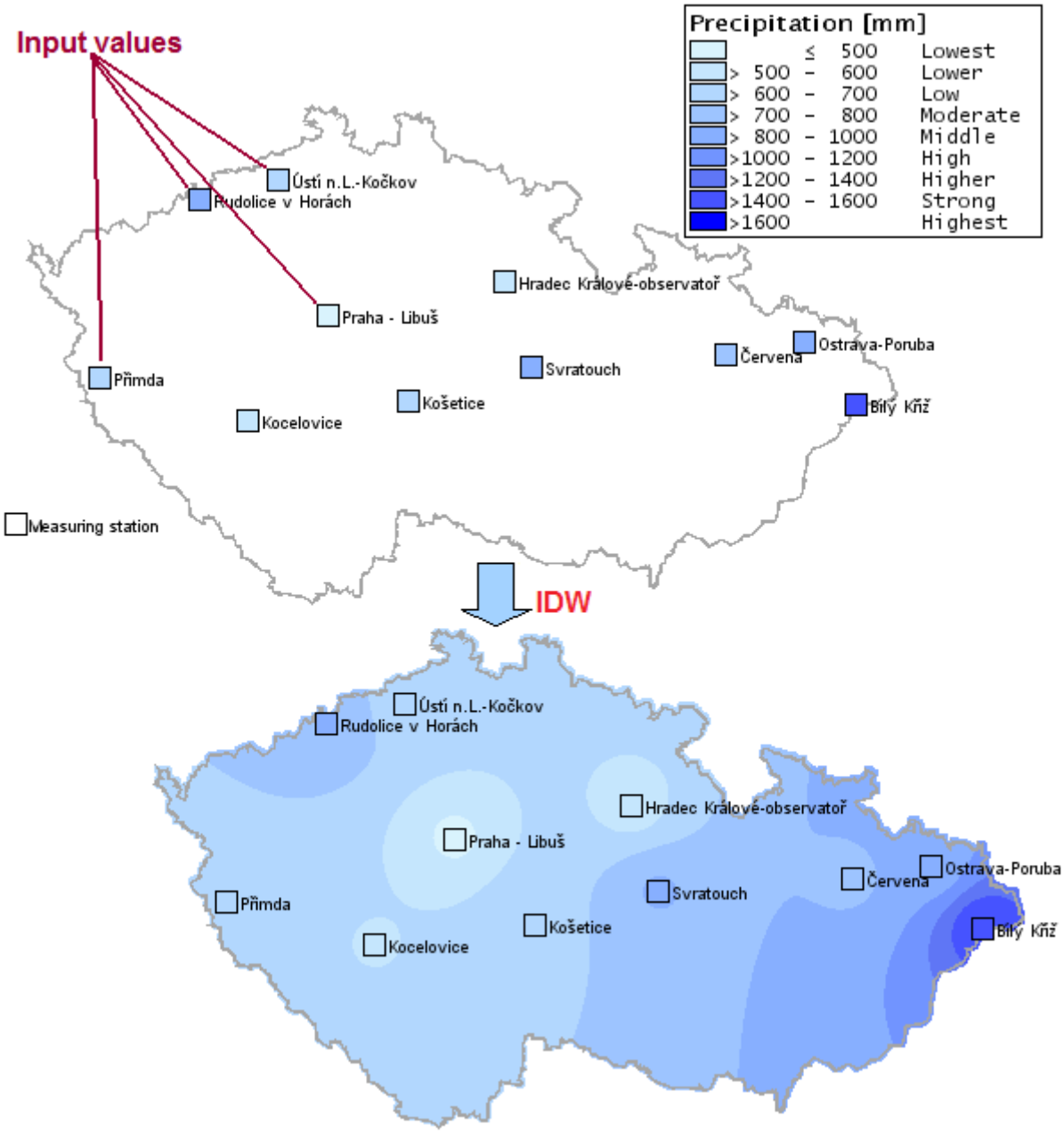
This type of scale is smooth transition between one color to another.

1.1.7. Interpolation

Interpolation is used for computing values into cells of computed matrix from known input values. Every computed matrix must have associated interpolation algorithm.

1.1.7.1 Simple Interpolation - IDWInterpolator

IDWInterpolator is basic interpolator, that can be used for determining values on places, where we not have obtained or measured values.



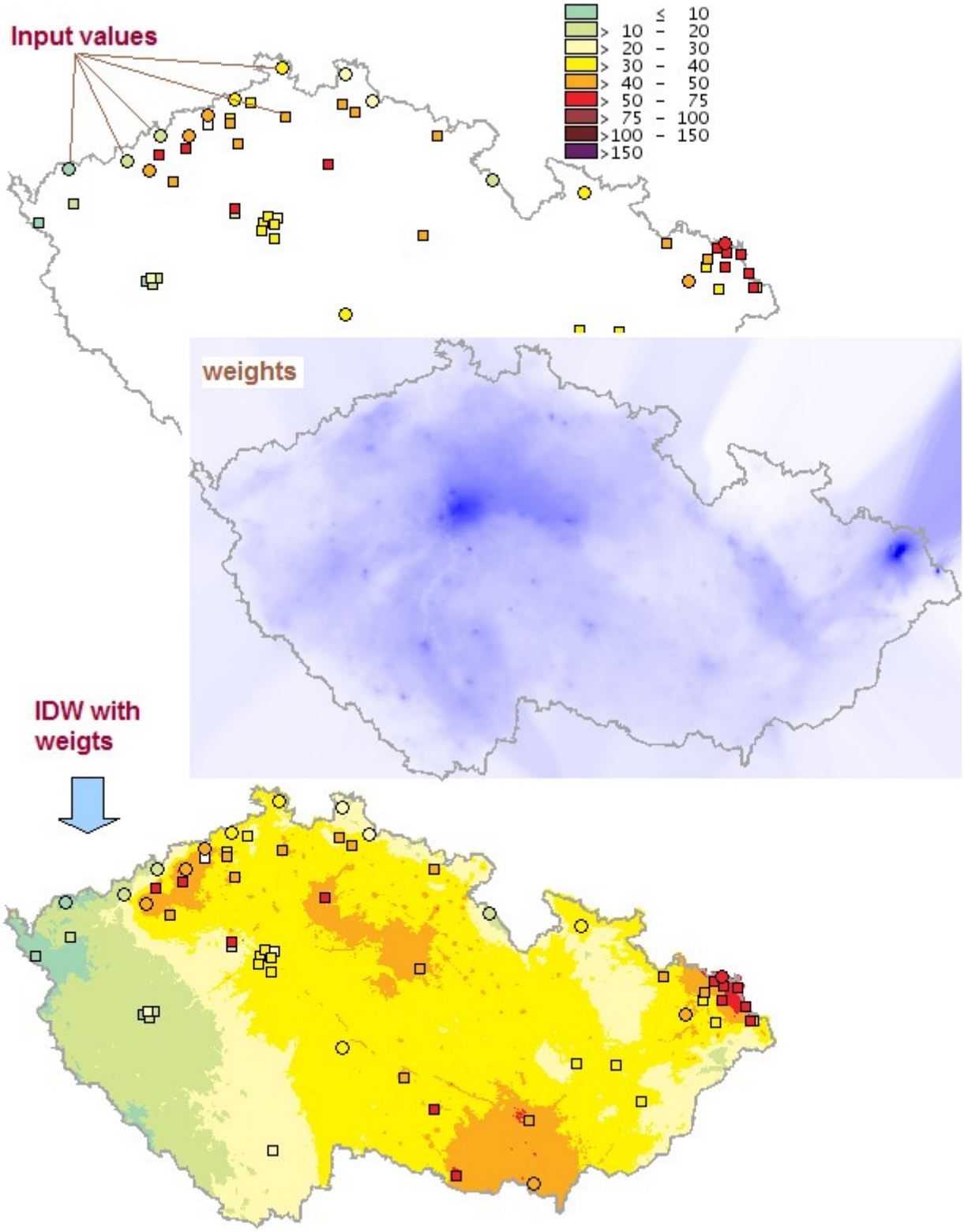
IDWInterpolator computes for each cell in computed matrix its value, and those values can be converted to colors using color scale and used for drawing map background.

1.1.7.2 More sophiscated interpolators

Some interpolators does computation in more complex way. They are able to load another pre-computed matrixes or coefficients, and use it's values to create output that better represents real situation.

- **IDWCoefWeightInterpolator** – uses two input matrixes, one with coefficient, second with weights values
- **IDWConstCoefWeightInterpolator** - uses common coefficient and input matrix with weights values
- **IDWConstCoefComputedWeightInterpolator** – uses common coefficient, weight matrix is computed by regression from two input matrixes

As you may see on following image, output of this interpolation is much different, because final matrix values not affected only by input values, but also with weights matrix, that contain pre-computed model of measured value behavior.

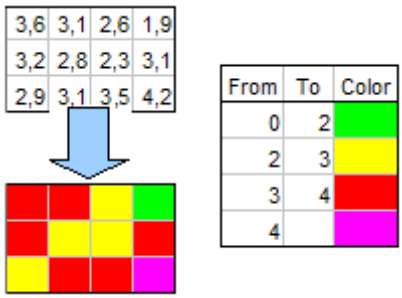

Input values
≤ 10
> 10 - 20
> 20 - 30
> 30 - 40
> 40 - 50
> 50 - 75
> 75 - 100
>100 - 150
>150
weights
IDW with weigts

1.1.8. Modification

Modification is used for traverse cells of 1..N matrixes (input or computed) and make correction using particular mathematical formula. The result may be stored into original matrix, or into newly created result matrix.

1.1.9. Rendering

Rendering get matrix, that contain final values and create map background. Every cell value is converted into color and stored into particular pixel in map.



In next phase, color background is repainted with vector shapes – borders, legend, measuring stations etc.

1.1.10. Runset

Runset takes control over data processing in Map Generator from loading input values, interpolations into matrixes, combinations, modifications, rendering, to storing final map into file or database.

Runset is XML file, where particular XML elements represents objects in Map Generator. By editing Runset it is possible to change Map Generator behavior and tune the result by our requirements.

Runset have following hierarchical structure. There are only most important attributes, other are bypassed for simplicity and will be described later in documentation of objects and their XML elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<MapGenerator>
	<Description></Description>
	<Matrixes rows="..." columns="..." cellSize="..."/>
		<CartesianGeoPosition id="leftBottomCorner" x='...' y='...'/>
		<InputMatrix id="..."/>
		<ComputedMatrix id="...">
			<DataValues/>
			<Interpolator/>
		</ComputedMatrix>
		<ComputedMatrix id="..." .../>
			<DataValues .../>
			<Interpolator .../>
		</ComputedMatrix>
```

```
            <Modificators>
                  <Modificator .../>
                  <Modificator .../>
            </Modificators>
      </Matrixes>
      <Renderer matrix="...">
            <ColorScale ... />

            <GeographicMap scale="...">
                  <CartesianGeoPosition id="leftBottomCorner" x='...' y='...'/>
                  <CartesianDimension id="dimension" width="..." height="..."/>


                  <Shape .../>
                  <Filter .../>
                  <Shape .../>
            </GeographicMap>
      </Renderer>
      <MapWriters>
            <MapWriter .../>
            <MapWriter .../>
      </MapWriters>

      <MatrixWriters>
            <MatrixWriter .../>
            <MatrixWriter .../>
      </MatrixWriters>
</MapGenerator>
```

MapGenerator – main object, controls whole process. Contains:

- **Matrixes** – Matrixes list. Contains 1 – N matrixes, input or computed, identified by unique id.
  - **CartesianGeoPosition** – Geographic position of left bottom corner of all matrixes.
  - **InputMatrix** – input matrix
  - **ComputedMatrix** – computed matrix
    - **DataValues** – input values list
    - **Interpolator** – Interpolation algorithm
  - **Modificators** – List of 0 – N modification algorithms
    - **Modificator** – modification algorithm 1
    - **Modificator** – modification algorithm N
- **Renderer** – It's responsible for rendering final matrix into map
- **ColorScale** – The color scale
  - **GeographicMap** – Map, it have position, size and contains 0 – N vector shapes and filters
    - **CartesianGeoPosition** – Geographic position of left bottom pixel in map
    - **Dimension** – Size of map in pixels
    - **Shape** – shape
    - **Filter** - filter
    - **Shape** – shape
    - ...
    - **MapWriters** – List of 1 – N map writers
      - **MapWriter** Map writer, for example into file
      - **MapWriter** Map writer, for example into database
    - **MatrixWriters** – List of 0 – N matrix writers
      - **MapWriter -** Matrix writer, for example into file

- **MapWriter -** Matrix writer, for example into database
  - **DataValuesWriter -** Writer of matrix input values

Runset is processed by following steps:

1. XML file is loaded and parsed. From every XML element is created particular Java object - in this way are created and initialized whole hierarchy of objects.
2. For all computed matrixes input values are loaded and then interpolated into matrix.
3. Modification algorithms are subsequently performed
4. Renderer transforms final matrix using color scale into pixels in maps
5. Renderer overlaps map with vector shapes and filters are processed
6. Map writers are subsequently performed
7. Matrix writers are subsequently performed

2. Using program

2.1. System requirements

Map Generator requires Java Runtime Environment version 1.8 or newer. You can obtain it on:

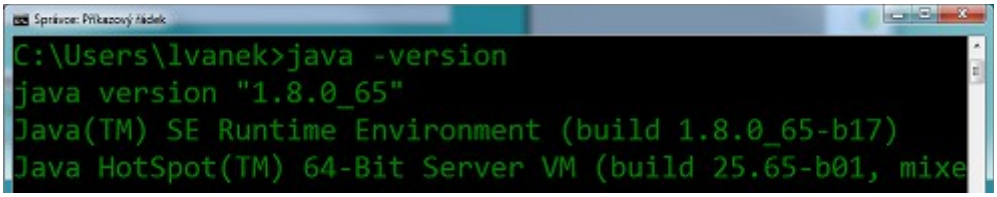
http://www.java.com/en/download

2.1.1. Checking Java version

Use command

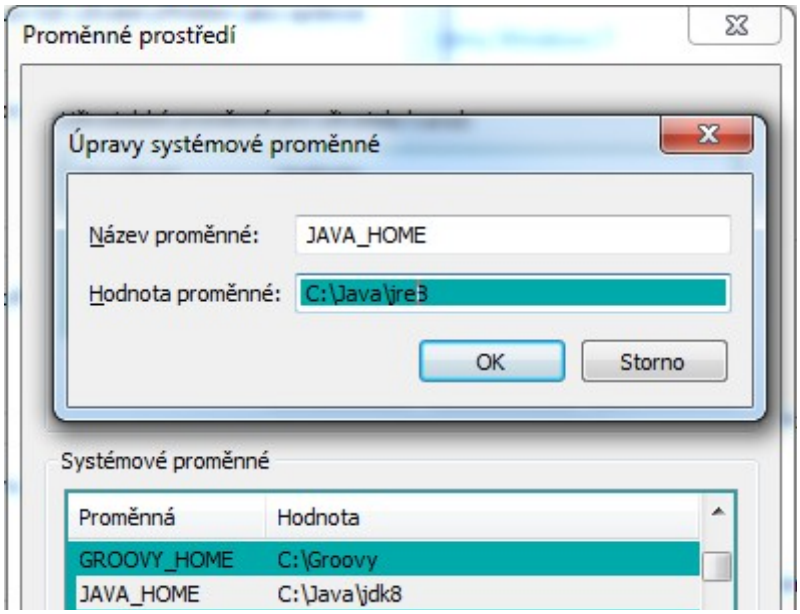
```
java -version
```

on command line to check your Java installation.



2.1.2. Placing Java bin directory to PATH

It is good practice to have added Java **bin** directory into PATH system variable, to be able run Java from command line without scecifying full path. On Windows, use Control Panel - System.



2.2. JVM parameters

It is possible to pass some arguments for Java Virtual Machine, that have influence to performance of program running on it. Because some map generation processing can run in multiple threads, it is good idea to use following JVM parameters on multiprocessor machines:

```
-XX:+UseParallelGC -XX:+UseAdaptiveSizePolicy
```

For best performance, you can tune another parameters:

```
-XX:+UseParallelGC -XX:+UseParallelOldGC -XX:ParallelGCThreads=6
```

For creating large maps, you can tune memory usage:

```
-Xms512m -Xmx4g
```

Interpolators and modificators objects contains in Runset parameter **threads**, that allows divide time-consuming tasks into more threads. Generally speaking, number of threads may be equals to number of machine physical processors.

2.3. General Map Generator parameters

Some Map Generator parameters are required each time so I document them here to avoid duplication.

2.3.1. Parameter logLevel

Determine level of logging as number by following table:

logLevel	Abbreviation	Description
0	ALL	Log all
1	FINEST	Highly detailed trace messages
2	FINER	Fairly detailed trace messages
3	FINE	Trace messages
4	CONFIG	Configuration messages
5	INFO	Iformational messages
6	WARNING	Messages indicating a potential problem
7	SEVERE	Messages indicating a serious failure
8	OFF	Logging is turned off

For production use is suitable level 4.

2.3.2. Parameter logFile

If you require logging into file, use it's name as parameter. If not used, program logs only to console.

2.3.3. Parameter runsetFile

Use filename including path (may be relative) with XML Runset.

3.Running Map Generator

Map Generator can be launched from command line:

```
java -XX:+UseParallelGC -XX:+UseAdaptiveSizePolicy -jar MapGenerator.jar /X
… …
```

where first parameter after slash determine program behavior:

- **/F** – Process one XML Runset from file
- **/D** - Demo mode – display map example

3.1. /F - Process one Runset from file

For process Runset from XML file is used parameter /F (File). Another parameters:

- **logLevel** – Level of logging
- **logFile** – log file
- **runsetFile** – path to XML file with Runset

Example:

```
java -XX:+UseParallelGC -XX:+UseAdaptiveSizePolicy -jar MapGenerator.jar /F
3 log.txt .\runsets\RunSet1.xml
```

3.2. /D – Display demo map

For display demonstration map created by Java API used parameter /D (Demo). Another parameters:

- **logLevel** – Level of logging
- **logFile** – log file

Example:

```
java -jar MapGenerator.jar /D 3 log.txt
```

4.Runset - Reference manual

This part describes particular XML elements in Runset, from wich are created objects that generates map.

Most of objects have common attributes, so I document them here to avoid duplication.

4.1. Attribute id

If program require to unambiguously determine objects in list, they have **id** attribute. For example every XML element **InputMatrix** and **ComputedMatrix** have **id** attribute, wich value must be unique for all matrixes in the Runset.

4.2. Attribute class

While developing of Map Generator we emphasis on flexibility and extensibility. Map Generator is able to generate statistics maps from every area and it's abilities can be extended.

This extensibility in practice is provided by ability of Java to load classes at runtime using classloader. Attribute **class** contain class-name, that provide required functionality. Loaded class must be inside MapGenerator.jar, or can be found in extending module in Java language that user create.

For example, in the case that user of Map Generator found existing Interpolation algorithm as unsatisfactory, it is possible to code own one and in Runset replace attribute **class** content `class="idea.map.interpolator.IDWInterpolator"` by name of your own class.

Only one condition is, that user interpolation class must be derived from class **`idea.map.interpolator.AbstractInterpolator`**, to comply contract between user class and rest of program.

By this way is possible to create own combination a modification algorithms, input data sources, map writers, graphic objects (shapes), filters etc.

4.3. Attributes threads, maxThreads and timeout

Objects types Interpolator and Modificator, that performs time-consuming tasks with matrixes are able to divide matrix to smaller blocks and process them in separate threads. Those objects have two atributes, that setup multithreading.

- **threads** – number of threads. If not set, default is number of processors available to the Java virtual machine.
- **maxThreads** – when **threads** is not specified, you can limit maximum number of threads.

For example on machine with 8 CPUs, set **maxThreads** to 6 to leave 2 CPUs for another tasks.

- **timeout** – Number of seconds for finishing task. When exceeded, operation is terminated. If not set, default is 60.

Number of threads can be equals to number of machine processors. Also is good idea to use optimalization parameters on multiprocessors machines – see chapter JVM parameters.

4.4. Geometric objects

Those objects serves for define position, size or offset in **pixels**. There are submerged in objects, whose properties sets.

Example:

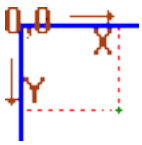
```
<Shape class="idea.graphics.shapes.simple.Image" fileName="./img/logo.gif">
        <CartesianPosition id="position" x="50" y="35"/>
</Shape>
```

Graphic object (shape) of type **Image** will be placed into map at pixels position defined by attributes of XML element **CartesianPosition**.

4.4.1. CartesianPosition

```
<CartesianPosition id="position" x="100" y="300"/>
```

Object absolute position in pixels, 0-0 is left top. This coordinate system is used by Java 2D API for drawing the image of map.



4.4.2. CartesianDimension

```
<CartesianDimension id="dimension" width="900" height="548"/>
```

Size of object in pixels.

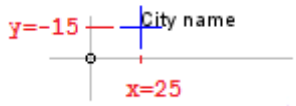
4.4.3. CartesianOffset

```
<CartesianOffset id="offset" x="900" y="548"/>
```

Cartesian relative offset, specified in pixels. Offsets are used mostly for positioning part of shape with regard to rest of it.

Example:

```
<Shape class="idea.graphics.shapes.complex.geo.City" name="City name" size="MEDIUM"
markColor="#000000" textColor="000000">
       <CartesianPosition id="position" x='50' y='50'/>
       <CartesianOffset id="labelOffset" x="25" y="-15"/>
</Shape>
```



This example shows, how to define relative offset between city mark (the circle) and text with city name using **CartesianOffset**.

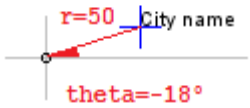
4.4.4. PolarOffset

```
<PolarOffset id="firstEdge" angle="30" radius="70"/>
```

Polar relative offset, radius is in pixels, angle in degrees. Angle is interpreted such that 0 degrees is at the 3 o'clock position.

Example:

```
<Shape class="idea.graphics.shapes.complex.geo.City" name="City name" size="MEDIUM"
markColor="000000" textColor="000000">
    <CartesianPosition id="position" x='50' y='50'/>
    <PolarOffset id="labelOffset" angle="-18" radius="50"/>
</Shape>
```



This example shows, how to define relative offset between city mark (the circle) and text with city name using **PolarOffset**.

4.5. Geographic objects

Those objects serves for define position, size or offset in **geographic units** (e.g. meters). They are submerged in objects, whose properties sets.

Example:

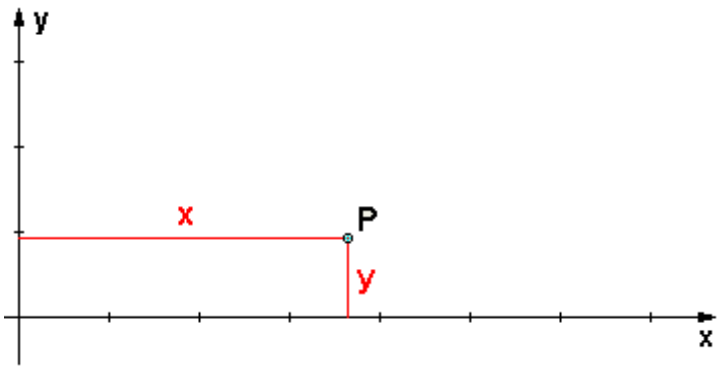
```
<Shape class="..." ... >
    <CartesianGeoPosition id="centerPosition" x='3373000' y='5639000'/>
</Shape>
```

Specifying of concrete geographic unit is not required - used units does not matter. Map Generator only needs to know, how many geographic units is represented by one pixel in rendered map image – see parameter **scale** on **GeographicMap** object.

4.5.1. CartesianGeoPosition

```
<CartesianGeoPosition id="leftBottomCorner" x="3292000" y="5380000"/>
```

Object position, in cartesian geographic coordinates.



Cartesian coordinates are used to specifying the geographic place on the map. Map Generator can convert them to geometric (pixels) positions using map parameters – **scale** and cartesian position of **left bottom corner**.

4.5.2. CartesianGeoDimension

```
<CartesianGeoDimension id="dimension" width="120000" height="80000"/>
```

Size of object in geographic units.

4.5.3. CartesianGeoOffset

```
<CartesianGeoOffset id="offset" x="900" y="548"/>
```

Cartesian relative offset in geographic units.

4.5.4. S42GeoPosition

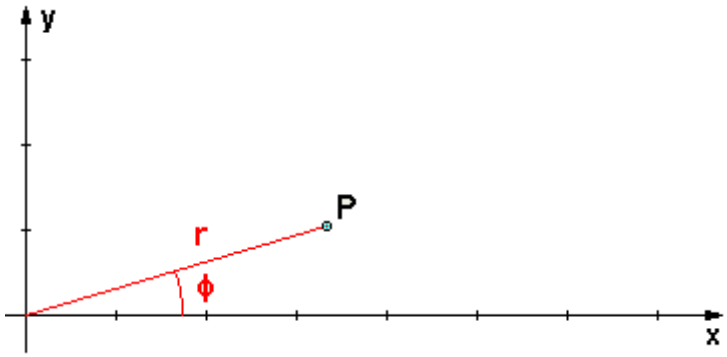
```
<S42GeoPosition id="leftBottomCorner" x="3292000" y="5380000"/>
```

Cartesian position in Gauss-Krüger S-42 coordinates, unit is meter.

4.5.5. PolarGeoPosition

```
<PolarGeoPosition id="leftBottomCorner" angle="30" radius="50000"/>
```

Object position, in polar geographic coordinates. Angular range is in 0 – 360 degrees, radius is in geographic units.



Using polar coordinates is not very obvious in Map Generator.

4.5.6. WGS84GeoPosition

```
<WGS84GeoPosition id="position" longitude='12°05&apos;29&quot;' latitude='50°15&apos;07&quot;'/>
```

Position, in WGS 84 coordinates, specified in latitude and longitude, unit is degreees, minutes and seconds. Using this object requires to have **CoordinateTransformation** used.

4.6. Main objects

4.6.1. MapGenerator

General Map generator from MapGenerator module (MapGenerator.jar). Have no parameters.

```
<MapGenerator>
        <Description></Description>
        <Matrixes rows="279" columns="488" cellSize="1000">
                ...
        </Matrixes>

        <Renderer>
                ...
        </Renderer>

        <MapWriters>
                ...
        </MapWriters>

        <MatrixWriters>
```

```
            ...
        </MatrixWriters>
</MapGenerator>
```

This generator can be launched using command:

```
java -jar MapGenerator.jar ...
```

4.6.1.1 Child objects

- **Description** – Runset description
- **Matrixes** – matrixes list
- **Renderer** – Map renderer
- **MapWriters** – list of 1 – N map writers
- **MatrixWriters** – list of 0 – N matrix writers

Matrixes, Input Data Values and Interpolators

4.6.2. Matrixes

Matrixes list. All matrixes must have this same position of left bottom corner, cell size and numbers rows and colums.

```
<Matrixes rows="279" columns="488" cellSize="1000">
        <CartesianGeoPosition id="leftBottomCorner" x='3292000' y='5380000'/>
                <InputMatrix id="..." .../>
                <ComputedMatrix id="..." ...>
                ...

         <Modificators>
                ...
         </Modificators>
</Matrixes>
```

4.6.2.1 Attributes

- **rows** – Row number of matrixes
- **columns** – Columns number of matrixes
- **cellSize** – Cell size of matrixes in **geographic units**

4.6.2.2 Child objects

- **CartesianGeoPosition**, id="leftBottomCorner", defines left bottom corner of matrixes
- 1 - N objects **InputMatrix** or **ComputedMatrix**, identified by unique **id**.
- **Modificators** – list of 0 – N modification algorithms

4.6.3. InputMatrix InputMatrixFromFile

Input matrix, loaded from file.

```
<InputMatrix class="idea.map.matrix.InputMatrixFromFile" id="HustotaObyvCR"
fileName="./geodata/hustota_obyv.txt"/>
```

4.6.3.1 Attributes

- **fileName** – Path to file containing matrix. Use slash as separator. File have following text structure:

```
ncols         488
nrows         279
xllcorner     3292000
yllcorner     5380000
cellsize      1000
NODATA_value  -9999
-9999 -9999 -9999 539.8067 544.5424 546.6235 552.7978 554.486  ...cut
```

There are matrix parameters in header, then follows cell values. Matrix parameters must be same as properties of **Matrixes** object in Runset.

4.6.4. ComputedMatrix

Computed matrix, it's cell values are computed from input data using interpolation.

```
<ComputedMatrix id="Background">
	<Interpolator class="..." />
	<DataValues class="..." />
</ComputedMatrix>
```

4.6.4.1 Child objects

- **Interpolator** – interpolation algorithm
- **DataValues** – list of input values

4.6.5. DataValues DataValuesFromXmlFile

List of input values, loaded from XML file.

```
<DataValues class="idea.map.data.DataValuesFromXmlFile" fileName="./data/rain2005.xml"/>
```

4.6.5.1 Attributes

- **fileName** – path to XML file containing input values. Use slash as separator.

XML file must have following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<DataValues>
	<row name="..." value="...">
		<CartesianGeoPosition x="..." y="..."/>
	</row>
```

```
        <row name="..." value="...">
                <S42GeoPosition x="..." y="..."/>
        </row>

        <row name="..." value="...">
                <WGS84GeoPosition latitude="..." longitude="..."/>
        </row>
</DataValues>
```

Row element have following attributes:

- name – name of place where value was obtained or measured
- value – the value

Row element have nested element **CartesianGeoPosition** or **S42GeoPosition** or **WGS84GeoPosition** that specified place, where values was obtained.

4.6.6. Interpolator IDWInterpolator

Interpolation algorithm IDW (Inverse Distance Weighting).

```
<Interpolator class="idea.map.interpolator.IDWInterpolator" stations="12" beta="2" threads="2"
excludeAreaMatrix="HustotaObyvCR"/>
```

4.6.6.1 Attributes

- **stations** – number of surrounding input values, whose input values are used for interpolation of one cell.
- **beta** – weight factor
- **excludeAreaMatrix** – id of matrix, whose NODATA cells are used for marking of area, that are skipped by interpolation. Attribute is not mandatory - when missing, whole matrix is computed.

4.6.7. Interpolator IDWCoefWeightInterpolator

Interpolation algorithm IDW, with coeficients and weight factors. Both coeficients and weight factors are loaded from input matrixes. Before interpolation, every input value is corrected by formula:

$$value = value - coefficient * weight$$

Then standard IDW interpolation is performed. Finally, each matrix cell is corrected:

$$value = value + coefficient * weight$$

```
<Interpolator class="idea.map.interpolator.IDWCoefWeigthInterpolator" stations="12" beta="2"
threads="2" excludeAreaMatrix="HustotaObyvCR" coefficientMatrix="coefficients"
weightMatrix="weights"/>
```

4.6.7.1 Attributes

- **stations** – number of surrounding input values, whose input values are used for interpolation of one cell.

- **beta** – weight factor
- **excludeAreaMatrix** – id of matrix, whose NODATA cells are used for marking of area, that are skipped by interpolation. Attribute is not mandatory - when missing, whole matrix is computed.
- **weightMatrix** – id of input matrix, containings weight factors
- **coefficientMatrix** – id of input matrix, containings coeficients

4.6.8. Interpolator IDWConstCoefWeightInterpolator

Interpolation algorithm IDW, with **constant** coeficient and weight factors. Coeficient is common, weight factors are loaded from input matrix. Before interpolation, every input value is corrected by formula:

value = value – coefficient * weight

Then standard IDW interpolation is performed. Finally, each matrix cell is corrected:

value = value +coefficient * weight

```
<Interpolator class="idea.map.interpolator.IDWConstcoefWeigthInterpolator" stations="12" beta="2"
threads="2" excludeAreaMatrix="HustotaObyvCR" coefficient="48.5" weightMatrix="weights"/>
```

4.6.8.1 Attributes

- **stations** – number of surrounding input values, whose input values are used for interpolation of one cell.
- **beta** – weight factor
- **excludeAreaMatrix** – id of matrix, whose NODATA cells are used for marking of area, that are skipped by interpolation. Attribute is not mandatory - when missing, whole matrix is computed.
- **weightMatrix** – id of input matrix, containings weight factors
- **coefficient** – Coeficient value

4.6.9. IDWConstCoefComputedWeightInterpolator

```
<Interpolator class="idea.map.interpolator.IDWConstCoefComputedWeightInterpolator" maxThreads="6"
timeout="60" stations="12" beta="2" excludeAreaMatrix="" coefficient="1" regressionMatrix1="pm10"
regressionMatrix2="pm10_model"/>
```

This interpolator have **two computing stages**: First, weights are computed using regression from two input matrixes.

Then, this same algorithm as in previous interpolator is used.

4.6.9.1 Attributes

- **stations** – number of surrounding input values, whose input values are used for interpolation of one cell.
- **beta** – weight factor
- **excludeAreaMatrix** – id of matrix, whose NODATA cells are used for marking of area, that are skipped by interpolation. Attribute is not mandatory - when missing, whole matrix is computed.
- **regressionMatrix1** – id of input matrix 1, used for interpolation
- **regressionMatrix1** – id of input matrix 2, used for interpolation
- **coefficient** – Coeficient value

4.6.10. Interpolator NoopInterpolator

Interpolator that does nothing. Can be used for situations, where we need only input data, not interpolated values. When it is used for final matrix, the **Renderer** can have set renderBackground="**false**", because there is nothing to render. Input data can be visualized using **Shape** class="idea.graphics.shapes.complex.Stations".

```
<Interpolator class="idea.map.interpolator.NoopInterpolator"/>
```

4.7. Modificators

List 0 – N modification algorithms. Modification algorithms are performed one by one in sequence as defined in list.

4.7.1.1 Child objects

- 0-N objects of type **Modificator**

```
<Modificators>
                <Modificator class="..." />
                <Modificator class="..." />
</Modificators>
```

4.7.2. Modificator Phase1Combinator

Combination algorithm, from input matrixes A, B and C store values into result matrix D by formula:

$$D = (A > C) ? B : A$$

If in A, B, or C is NODATA, then reult is too NODATA.

```
<Combinator class="idea.map.combinator.Phase1Combinator" matrixA="Background" matrixB="All"
matrixC="Urban" matrixD="matrixD" threads="2" timeout="60"/>
```

4.7.2.1 Attributes:

- **matrixA** – id of matrix, that act in formula as variable A
- **matrixB** – id of matrix, that act in formula as variable B
- **matrixC** – id of matrix, that act in formula as variable C
- **matrixD** – id of result of matrix, that act in formula as variable D

4.7.3. Modificator Phase2Combinator

Combination algorithm, from input matrixes C, D and E store values into result matrix F by formula:

$$F = (C * E) + (1 - E) * D$$

If in C, D, or E is NODATA, then reult is too NODATA.

```
<Combinator class="idea.map.combinator.Phase2Combinator" matrixC="Urban" matrixD="matrixD"
matrixE="HustotaObyvCR" matrixF="matrixF" threads="2" timeout="60"/>
```

4.7.3.1 Attributes:

- **matrixC** – id of matrix, that act in formula as variable C
- **matrixD** – id of matrix, that act in formula as variable D
- **matrixE** – id of matrix, that act in formula as variable E
- **matrixF** – id of result of matrix, that act in formula as variable F

4.7.4. Modificator HiPassFilter

Modificator make correction of matrix values by formula:

$$value = value > limit \ ? \ limit : value$$

it means that all values that exceeds the limit replace by limit, rest leave as is. NODATA values remains unchanged.

```
<Modificator class="idea.map.modificator.HiPassFilter" limit="500"  matrix="matrixD"/>
```

4.7.4.1 Attributes

- **limit** – limit value
- **matrix** – id of modified matrix

4.7.5. Modificator LoPassFilter

Modificator make correction of matrix values by formula:

value = value < limit ? limit : value

it means that all values under the limit replace by limit, rest leave as is. NODATA values remains unchanged.

```
<Modificator class="idea.map.modificator.LoPassFilter" limit="0.1"  matrix="matrixD"/>
```

4.7.5.1 Attributes

- **limit** – limit value
- **matrix** – id of modified matrix

4.8. Renderer, Color Scale and Map

Renderer get final matrix and using the Color Scale transform its values to colors and paint them to particular pixels in map.

4.8.1. Renderer GeoImageMapRenderer

Renderer of final matrix to map as bitmap image.

```
<Renderer class="idea.map.renderer.GeoImageMapRenderer" matrix="matrixF">
       <ColorScale class="..." />
       <GeographicMap scale="...">
              ...
       </GeographicMap>
</Renderer>
```

4.8.1.1 Attributes

- **matrix** – matrix id, whose values are used for painting map backgroud
- **renderBackground** – value **false** disable rendering of color background. Not mandatory, default is **true.**

4.8.1.2 Child objects

- **ColorScale** – scale used for transform final matrix values to colors
- **GeographicMap** – map, where matrix values will be painted

4.8.2. ColorScale StandardListColorScale

This color scale is type of **list color scale**. Those scales are defined as **list** of **ColorForValueRange** items. This color scale have parameters defined directly inside the XML

Runset.

```
<ColorScale class="idea.map.colorscale.StandardListColorScale" text="Altitude [m]"
outOfRangeColor="00FFFF">
        <ColorForValueRange name="Lowest" color="#FFFFFF" valueFrom="0" valueTo="150" seq="1" />
        <ColorForValueRange name="Lower" color="#89F964" valueFrom="150" valueTo="200" seq="2"/>
        <ColorForValueRange name="Low" color="#37C507" valueFrom="200" valueTo="300" seq="3"/>
        <ColorForValueRange name="Low-Mid" color="#288F05" valueFrom="300" valueTo="400" seq="4"/>
        <ColorForValueRange name="Mid" color="#C28854" valueFrom="400" valueTo="500" seq="5"/>
        <ColorForValueRange name="High" color="#986536" valueFrom="500" valueTo="600" seq="6"/>
        <ColorForValueRange name="Higher" color="#664324" valueFrom="600" valueTo="700" seq="7"/>
        <ColorForValueRange name="Highest" color="#48301A" valueFrom="700" seq="8"/>
</ColorScale>
```

4.8.2.1 Attributes

- **text** – text, that displays object **ListColorScaleLegend** as legend title.
- **outOfRangeColor** – color, that is used for eventually values out of scale range. If not specified and value out of range is ocurred, it causes error.

4.8.2.2 Child objects

- 1 – N objects of type **ColorForValueRange**

Note: See graphic shape **ListColorScaleLegend** to learn, how-to display legend for this scale on map.

4.8.3. ColorForValueRange

Item for list color scales - defines values range and assign color to it.

```
<ColorForValueRange name="Lowest" color="#FFFFFF" valueFrom="0" valueTo="150" seq="1" />
```

4.8.3.1 Attributes

- **name** – text, that displays object **ListColorScaleLegend** next to item
- **color** – color, defined as three hexadecimal RGB numbers
- **valueFrom** – low range value
- **valueTo** – high range value, last range in scale not have it
- **seq** – sequence of range

4.8.4. ColorScale LinearTwoColorScale

This color scale is type of **linear color scale**. This is created as smooth color transition of one color to another.

```
<ColorScale class="idea.map.colorscale.LinearTwoColorScale" colorNegative="BLUE" colorPositive="RED"
text="Precipitation" unit="mm" minValue="0" maxValue="2000"/>
```

4.8.4.1 Attributes

- **text** – text, that displays object **LinearColorScaleLegend** as legend title.
- **outOfRangeColor** – color, that is used for eventually values out of scale range. If not specified and value out of range is ocurred, it causes error.
- **minValue** – low range value, represented by dark **colorNegative** color
- **maxValue** – high range value, represented by dark **colorPositive** color
- **colorNegative** - basic color used for negative part of scale
- **colorPositive** - basic color used for negative part of scale

Note: Zero value is represented by white color. See graphic shape **LinearColorScaleLegend** to learn, how-to display legend for this scale on map.

Only limited set of colors can be used in **colorNegative** and **colorPositive** attributes, possible values are: BLUE, RED, YELLOW, MAGENTA, CYAN, GREEN, ORANGE, TEAL, PURPLE, OLIVE, SPRINGGREEN, DEEPSKYBLUE, DEEPPINK, BLACK.

4.8.5. GeographicMap

Geographic map. Contain color background, overlapped by vector objects (shapes).

```
<GeographicMap scale="600">
        <CartesianGeoPosition id="leftBottomCorner" x='3267000' y='5370000'/>
        <CartesianDimension id="dimension" width="900" height="548"/>

        <Shape .../>
        <Shape .../>
        ...
</GeographicMap>
```

4.8.5.1 Attributes

- **scale** – defines how many **geographic units** is represented by one pixel in map

4.8.5.2 Child objects

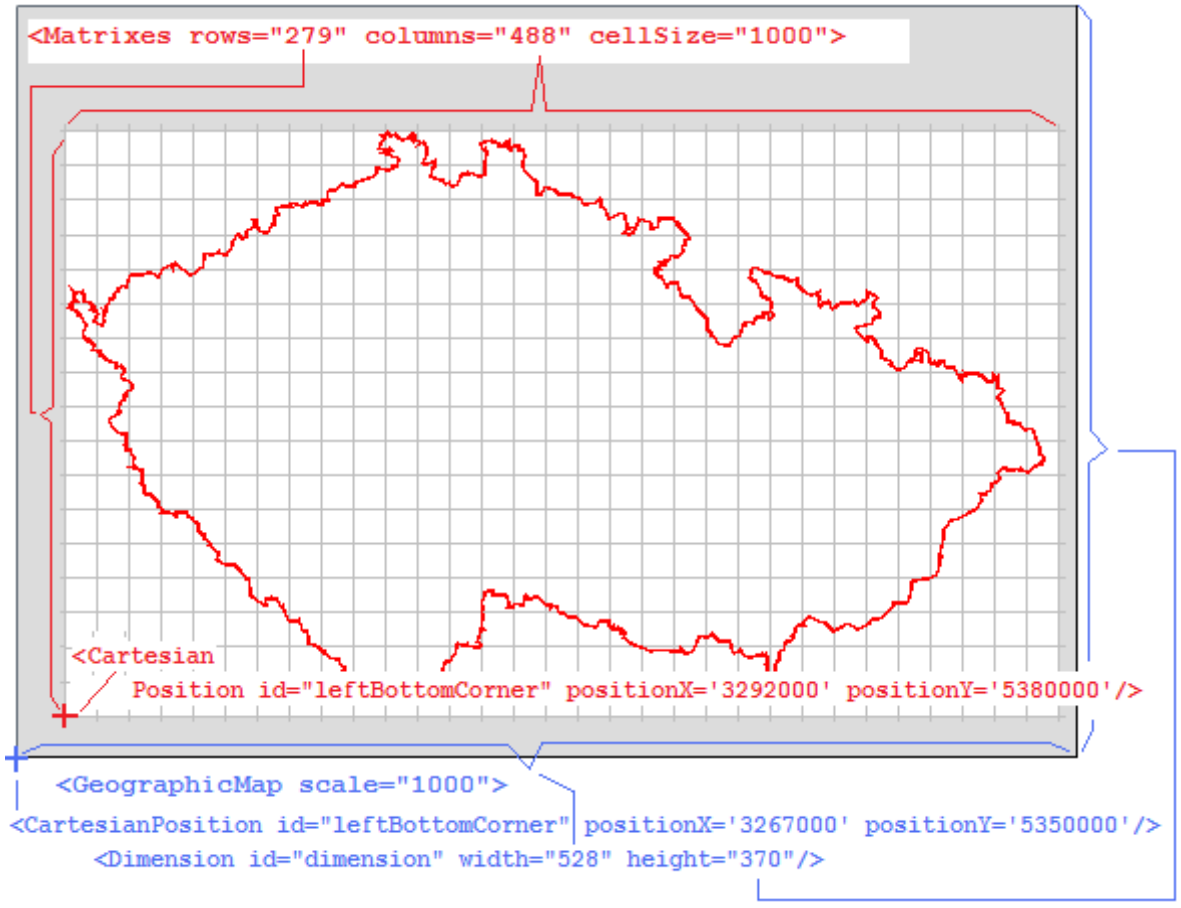
- **CartesianGeoPosition**, id="leftBottomCorner", defines geographic position of left bottom pixel in map
- **Dimension** – size of map in pixels
- **CoordinateTransformation** – non mandatory, required only when **WGS84GeoPosition**s are used
- **HtmlImageMap** – non-mandatory generator of HTML clickable map
- 0 – N objects of type **Shape**

4.8.5.3 Relations between matrixes and map parameters

Both matrixes and map have defined geographic position of left bottom corner. Those positions may be equals. When **maps position of left bottom corner** in X or Y axis is **smaller**, then **left** and **bottom** grow up void space, that can be used for better map look.

Matrixes have defined **cell size**, map have **scale**. Those values may be equals, then cell from final matrix corresponds with one pixel in map. When values differs, it cause zoom in or zoom out of map.

Matrixes have defined **number of rows and columns**, map have **dimension** in pixels. Those values may be equals, or is possible to have map size **larger**, then **right** and **top** grow up void space, that can be used for better map look.



On this map, there is 25 pixels void space on left side: (3292000 / 1000) - (3267000 / 1000) = 25px, and 61 pixels on top: (370 - 279) - ((5380000 / 1000) - (5350000 / 1000)) = 61

4.8.6. CoordinateTransformation CoordinateTransformationWGS84_S42

Coordinate transformation servers for transform WGS 84 positions to two-dimensional cartesian positions.

```
<CoordinateTransformation
class="idea.geographic.coordsys.transformation.CoordinateTransformationWGS84_S42" zoneS42="3"
s42zoneWide="dg6">
        <BursaWolf id="bursaWolf" dx="-23" dy="124" dz="84" ex="-0.13" ey="-0.25" ez="0.02" ppm="-
1.1e-6" />
</CoordinateTransformation>
```

4.8.6.1 Attributes

- **zoneS42** – number of S42 zone
- **s42zoneWide** – can be dg3 or dg6 for 3 or 6 degree zones;

4.8.6.2 Child objects

- **BursaWolf** – parameters of Bursa-Wolf transformation

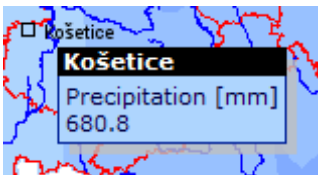
Map generator have implemented CoordinateTransformationSimple, but more sophiscated algorithms can be used by coding own class.

4.8.7. HtmlImageMap

Navigation HTML map for image, that contains HTML **area** tags for input values.

```
<HtmlImageMap>
        <HtmlImageMapItemsFromInputDataValues
class="idea.graphics.map.HtmlImageMapItemsFromInputDataValues" matrix="rain" type="square"
size="6"/>
</HtmlImageMap>
```

When content of this object is used in HTML file containing in **img** tag image with map, after moving mouse over position where input value are obtained, tooltip is showed:



Tooltip contain information about name of the place where input value are obtained and it value. Backgroud color is given from color scale.

4.8.7.1 Child objects

- 0 – N objects of type **HtmlImageMapItemsFromInputDataValues**

Tato funcion is based on library **JavaScript, DHTML Tooltips**, that is found on http://www.walterzorn.com/tooltip/tooltip_e.htm

For proper function is required to have **wz_tooltips.js** placed on web server and refer to it from web page.

4.8.8. HtmlImageMapItemsFromInputDataValues

Creates clickable map items from input values.

4.8.8.1 Attributes

- **matrix** – computed matrix id, whose input values are used for HTML **area** tags in clickable map.
- **type** – object type: CIRCLE, SQUARE, it shall to be same with this parameter in object **Stations**.
- **size –** size of object that show tooltip in pixels

4.9. Writers

Writers are responsible for write output to files, database, send over network, etc. Their can write final map image, enclosing HTML file for it, data values and another informations.

4.9.1. MapWriters

List of map writers. Contain at least one map writer. Map writer is responsible to write map do disc file, database or another place.

```
<MapWriters>
        <MapWriter class="idea.map.writer.MapWriterToFile" fileName="SO2.PNG"/>
        <MapWriter class="idea.map.writer.ImageMapWriterToCanvas"/>
        ...
</MapWriters>
```

4.9.1.1 Child objects

- 0 – N objects of type **MapWriter**

4.9.1.2 Common MapWriter attributes

- **turnOff** – value **true** disable the writer. Not mandatory, default is **false**

4.9.2. MapWriter MapWriterToFile

Write map into disc file.

```
<MapWriter class="idea.map.writer.MapWriterToFile" fileName="c:/so2.png"/>
```

4.9.2.1 Attributes

- **fileName** - file name, eventually including path

4.9.3. MapWriter MapWriterToHtmlFile

Write map into HTML file. It is used with combination in **MapWriterToFile**, that write map into PNG file, and **MapWriterToHtmlFile** write enclosing HTML, that contains image with clickable map created by object **HtmlImageMap**.

```
<MapWriter class="idea.map.writer.MapWriterToHtmlFile" fileName="SO2.html" imageFileName="so2.png"/>
```

4.9.3.1 Attributes

- **fileName** – HTML file name, eventually including path
- **imageFileName –** file name of image with map, eventually including path

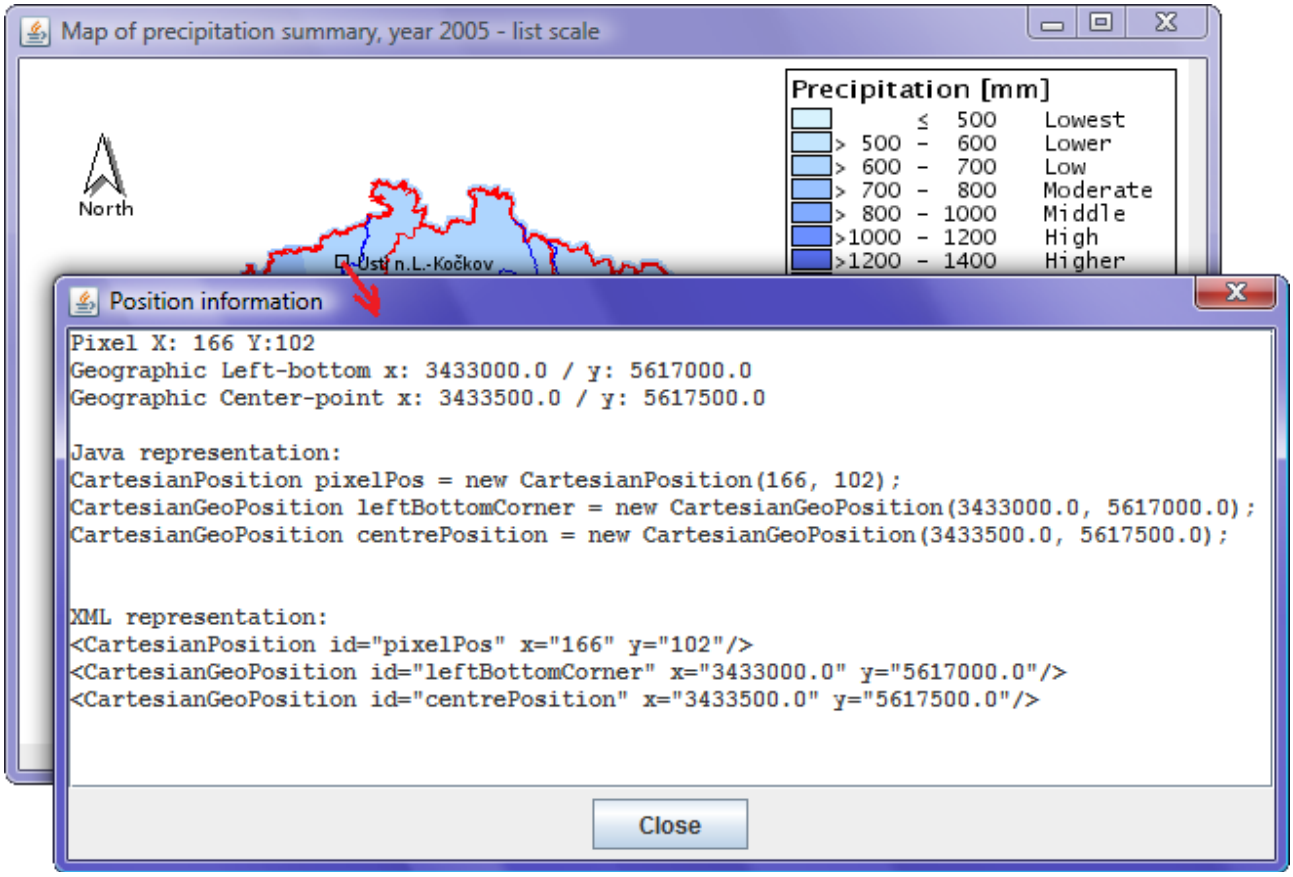
Tooltips requires for function file **wz_tooltip.js**, that is part of library **JavaScript, DHTML Tooltips**.

4.9.4. MapWriter ImageMapWriterToCanvas

Write map into canvas on screen. Causes showing window with map. Do not use this writer, if you process runsets in batch (for exaple from cron), because it causes pause while user closes window.

```
<MapWriter class="idea.map.writer.ImageMapWriterToCanvas"/>
```

When user click mouse over map - dialog with information about positions of this pixel is showed.



Java representation can be used in Java code, when you use API for driving Map Generator, XML code may be used in XML driven processing.

4.9.5. MatrixWriters

List of matrix writers. Matrix writer can be used to write any matrix values to file, database or another place.

```
<MatrixWriters>
        <MatrixWriter class="idea.map.writer.MatrixWriterToFile" matrix="rain"
fileName="rain2005.txt">
                <DataValuesWriter class="idea.map.writer.DataValuesWriterToXmlFile"
fileName="rain2005_out.xml"/>
        </MatrixWriter>
</MatrixWriters>
```

4.9.5.1 Child objects

- 0 – N objects of type **MatrixWriter**

4.9.5.2 Common MatrixWriter attributes

- **turnOff** – value **true** disable the writer. Not mandatory, default is **false**

4.9.6. MatrixWriter MatrixWriterToFile

Write matrix values to file with the same form as input for **InputMatrixFromFile**.

```
<MatrixWriter class="idea.map.writer.MatrixWriterToFile" matrix="rain" fileName="rain2005.txt"/>
```

4.9.6.1 Attributes

- **fileName** – file name, eventually including path
- **matrix** – id of matrix to write

4.9.6.2 Child objects

- **DataValuesWriter –** non-mandatory writer of input values. Can be used only when matrix is input matrix (InputMatrix).

4.9.7. DataValuesWriter DataValuesWriterToXmlFile

Write matrix input values to XML file with the same form as input for **DataValuesFromXmlFile**.

```
<MatrixWriter ...>
        <DataValuesWriter class="idea.map.writer.DataValuesWriterToXmlFile"
fileName="rain2005_out.xml"/>
</MatrixWriter>
```

4.9.7.1 Attributes

- **fileName** - file name of output XML file, eventually including path

4.10. Graphic objects - shapes and filters

When Renderer finish create bottom bitmap layer, next step is to overlap this layer by **shapes** and **filters**.

Shapes serves for adding graphic components to map. Filters serves for processing area – rotate, blur, etc.

Shapes and Filters are included in Runset under **GeographicMap** object.

4.10.1. Common properties

4.10.1.1 Positioning shape or filter

Most of shapes and filters requires to define position on map. It can be geometric **CartesianPosition** (in pixels, where 0-0 is left top) or geographic: **CartesianGeoPosition, PolarGeoPosition**, **S42GeoPosition or WGS84GeoPosition**.

Positions are interchangeable: you can use **CartesianPosition, CartesianGeoPosition, PolarGeoPosition, S42GeoPosition or WGS84GeoPosition** by your choice.

4.10.1.2 Setting dimension of shape or filter

Some shapes and all filters require to specify its dimension. It can be geometric **CartesianDimension** (in pixels) or geographic: **CartesianGeoDimension**, specified in geographic units.

Dimensions objects are interchangeable: you can use **CartesianDimension** or **CartesianGeoDimension** by your choice.

4.10.1.3 Moving shapes using offset

Some shape parts require to specify offset against rest of shape – typically text labels. It can be geometric: **CartesianOffset** (x/y in pixels), **PolarOffset** (radius/angle in pixels/degrees) or geographic: **CartesianGeoOffset**, specified in geographic units.

Offsets are interchangeable: you can use **CartesianOffset, PolarOffset** or **CartesianGeoOffset** by your choice.

4.10.1.4 Turning shape or filter off

All objects of type **Shape** and **Filter** have non-mandatory attribute **rendered**, that can turn off displaying object on map, without deleting it from Runset.

```
<Shape class="..." ... rendered="false"/>
```

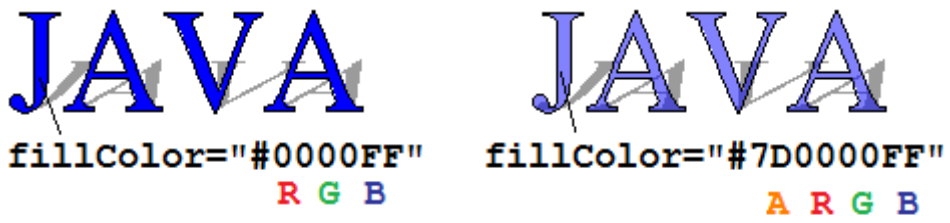
4.10.1.5 Setting colors for shape

Most of shapes requires to specify line color (**drawColor**), or fill color (**fillColor**) for solid shapes. Those values are entered as 6-digits hexadecimal number RGB, or 8-digits number ARGB with # prefix.

- A- alpha channel – defines transcluency, maximum is $7F_H$.
- R – intensity of red component
- G – intensity of green component
- B – intensity of blue component

Example:

```
<Shape class="mycompany.graphics.shapes.simple.TextWithShadow" fillColor="#7D0000FF"
drawColor="#000000" lineWidth="1" text="JAVA" font="Serif-BOLD-70">
    <CartesianPosition id="position" x="10" y="50"/>
</Shape>
```



Another posibility is named HTML colors with **$** prefix like:

```
drawColor="$Indigo"
```

See at http://www.w3schools.com/HTML/html_colornames.asp `for color definitions.`

When **fillColor** is not specified for solid objects (e.g. Rectangle, Oval, ...), object has no body, only outline is painted.

4.10.1.6 Setting line width

It is possible to specify line width in pixels for some objects – **lineWidth**.

Example:

```
lineWidth="2"
```

4.10.1.7 Setting font for shape

Shapes containings text requires to define font. It can be specified by following format:

`"fontfamilyname-style-pointsize"`

in which *style* is one of the four case-insensitive strings: `"PLAIN"`, `"BOLD"`, `"BOLDITALIC"`, or `"ITALIC"`, and pointsize is a decimal representation of the point size.

Examples:

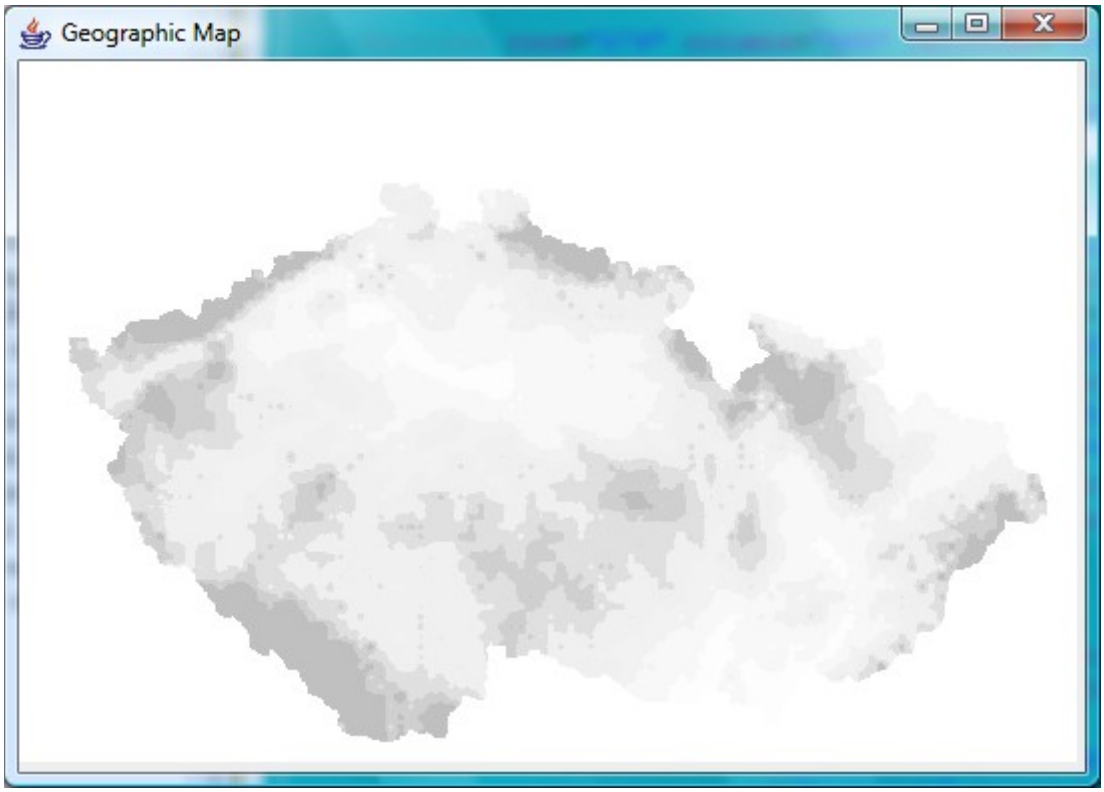
```
font="Lucida Sans Unicode-PLAIN-10"
font="Calibri-BOLD-18"
```

4.10.2. Trial Runset

For clarity is for every object of type Shape of Filter presented part of Runset, where object is used, and image with result. To avoid duplication, whole Runset is presented here. In it, you see green line with uncomplete Shape and Filter object. In following chapters are presented only relevant Runset parts, that refers to explained object. Those parts would be placed to highlighted line:

```
<?xml version="1.0" encoding="UTF-8"?>
<MapGenerator>
    <Description>test</Description>

       <Matrixes rows="279" columns="488" cellSize="1000">
              <CartesianGeoPosition id="leftBottomCorner" x='3292000' y='5380000'/>
              <InputMatrix class="idea.map.matrix.InputMatrixFromFile" id="altitudes"
fileName=".\geodata\vysko_interp.txt"/>
    </Matrixes>

    <Renderer class="idea.map.renderer.GeoImageMapRenderer" matrix="altitudes">
        <ColorScale class="idea.map.colorscale.ColorScale" text="Altitude [m]">
            <ColorForValueRange name="Lowest" color="#FFFFFF" valueFrom="0" valueTo="150" seq="1" />
            <ColorForValueRange name="Lower" color="#FDFDFD" valueFrom="150" valueTo="200" seq="2"/>
            <ColorForValueRange name="Low" color="#F8F8F8" valueFrom="200" valueTo="300" seq="3"/>
            <ColorForValueRange name="Low-Mid" color="#F2F2F2" valueFrom="300" valueTo="400"
seq="4"/>
            <ColorForValueRange name="Mid" color="#F0F0F0" valueFrom="400" valueTo="500" seq="5"/>
            <ColorForValueRange name="High" color="#E0E0E0" valueFrom="500" valueTo="600" seq="6"/>
            <ColorForValueRange name="Higher" color="#D0D0D0" valueFrom="600" valueTo="700"
seq="7"/>
            <ColorForValueRange name="Highest" color="#C0C0C0" valueFrom="700" seq="7"/>
        </ColorScale>

        <GeographicMap scale="1000">
            <CartesianGeoPosition id="leftBottomCorner" x='3267000' y='5370000'/>
            <CartesianDimension id="dimension" width="528" height="350"/>

            <Shape .../>
            <Filter .../>

       </GeographicMap>
    </Renderer>

       <MapWriters>
       <MapWriter class="idea.map.writer.ImageMapWriterToCanvas"/>
    </MapWriters>
</MapGenerator>
```

In Runset, there is loaded and displayed input hypsography matrix of Czech Republic in grey scale, to we have orientation background. This Runset (without one green linese so without graphics objects) would produce map, that looks like this:



4.11. Simple graphics objects

Namespace **idea.graphics.shapes.simple** contains classes for painting graphics primitives.

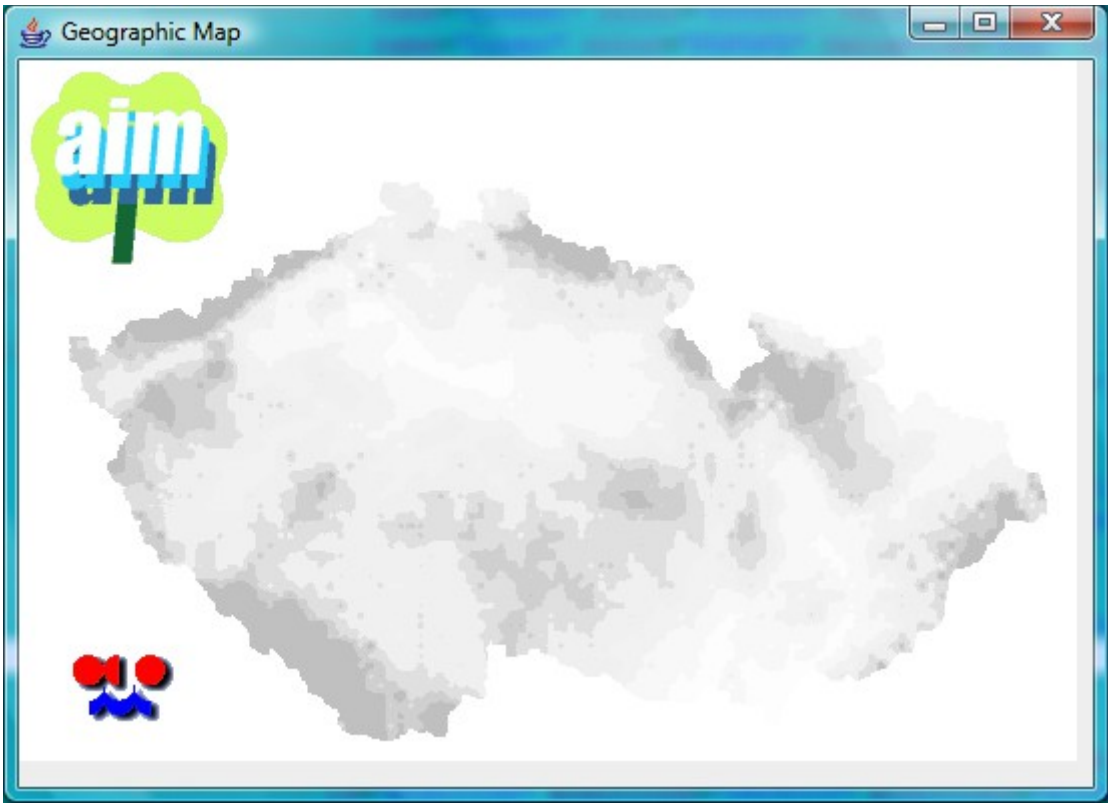
4.11.1. Image

Display bitmap image on defined position. First example defines position in S-42, second one in pixels.

```
<Shape class="idea.graphics.shapes.simple.Image" fileName="./img/logo_chmi.gif">
       <S42GeoPosition id="position" x="3280000" y="5426000"/>
</Shape>

<Shape class="idea.graphics.shapes.simple.Image" fileName="./img/aim-logo.gif">
       <CartesianPosition id="position" x="5" y="5"/>
</Shape>
```

4.11.1.1 Attributes

- **fileName** – path to file containing image



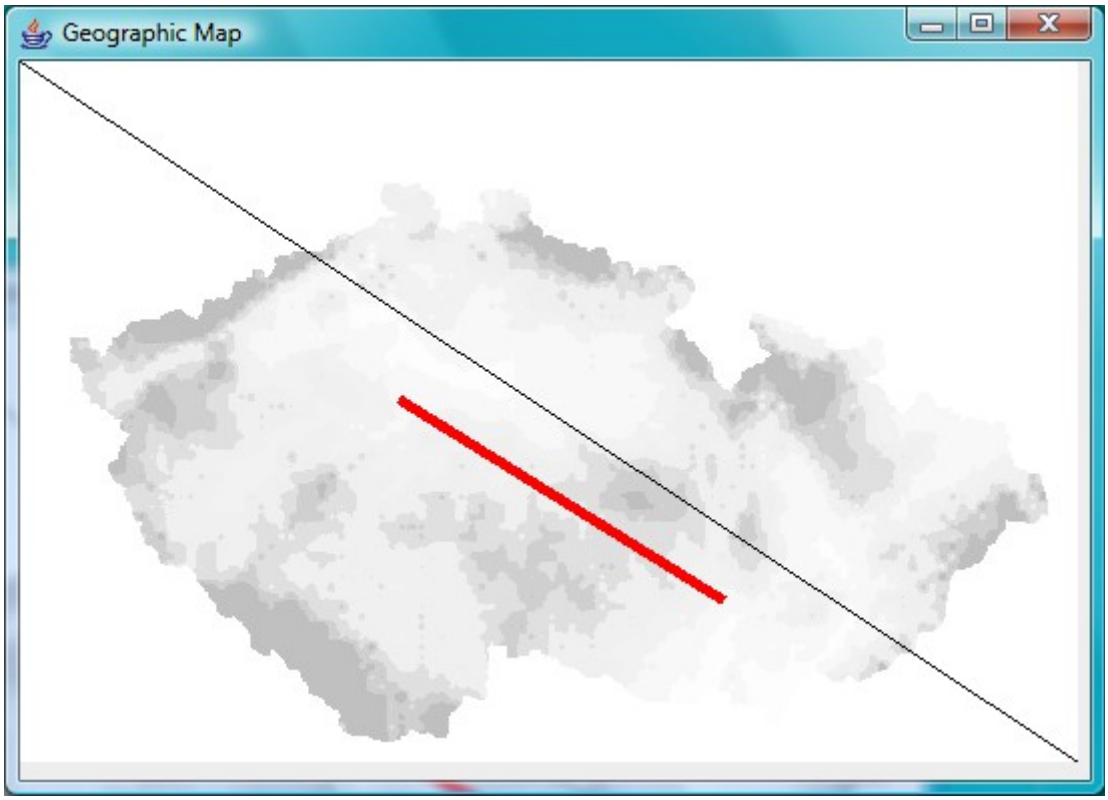
4.11.2. Line

Draw line, between defined begin and end with non-mandatory color and width. Presented example draw bold red line between Prague and Brno (defined in S-42), and diagonal (end position in pixels is same as map size) line with default properties.

```
<Shape class="idea.graphics.shapes.simple.Line" drawColor="#FF0000" lineWidth="5">
       <S42GeoPosition id="positionBegin" x="3458912" y="5550917"/>
       <S42GeoPosition id="positionEnd" x="3616717" y="5452708"/>
</Shape>

<Shape class="idea.graphics.shapes.simple.Line">
       <CartesianPosition id="positionBegin" x="0" y="0"/>
       <CartesianPosition id="positionEnd" x="528" y="350"/>
</Shape>
```

4.11.2.1 Attributes

- **drawColor** – color of line
- **lineWidth** – line width in pixels



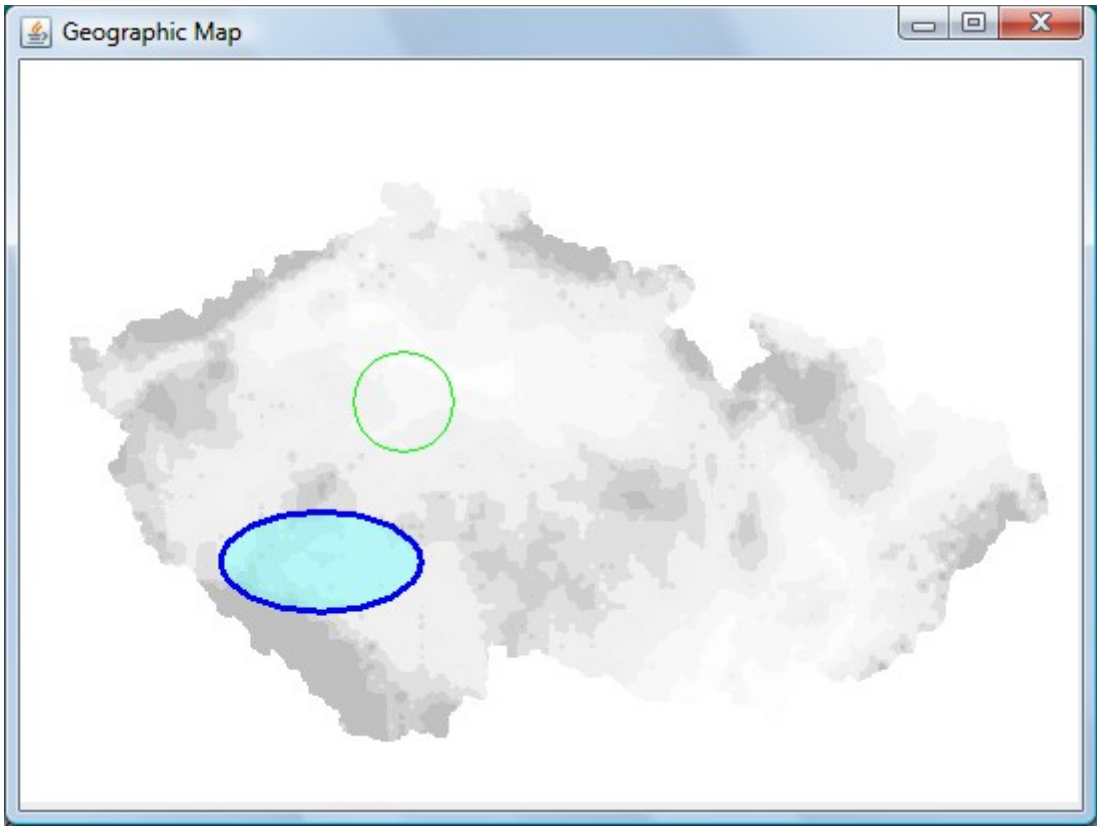
4.11.3. Oval

Draw oval, with defined centre position and size. First example draw circle with 50 km diameter around Prague (defined in S-42), second one paint semi-transcluent ellipse on defined position, parameters are in pixels.

```
<Shape class="idea.graphics.shapes.simple.Oval" drawColor="#00FF00" lineWidth="1" rendered="true">
       <CartesianGeoPosition id="centre" x="3458912" y="5550917"/>
       <CartesianGeoDimension id="dimension" width="50000" height="50000"/>
</Shape>

<Shape class="idea.graphics.shapes.simple.Oval" drawColor="#0000FF" fillColor="#7F80FFFF"
lineWidth="3" rendered="true">
       <CartesianPosition id="centre" x="150" y="250"/>
       <CartesianDimension id="dimension" width="100" height="50"/>
</Shape>
```

4.11.3.1 Attributes

- **drawColor** – color of line
- **fillColor** – non-mantatory fill color, if not present, object body is not filled
- **lineWidth** – line width in pixels



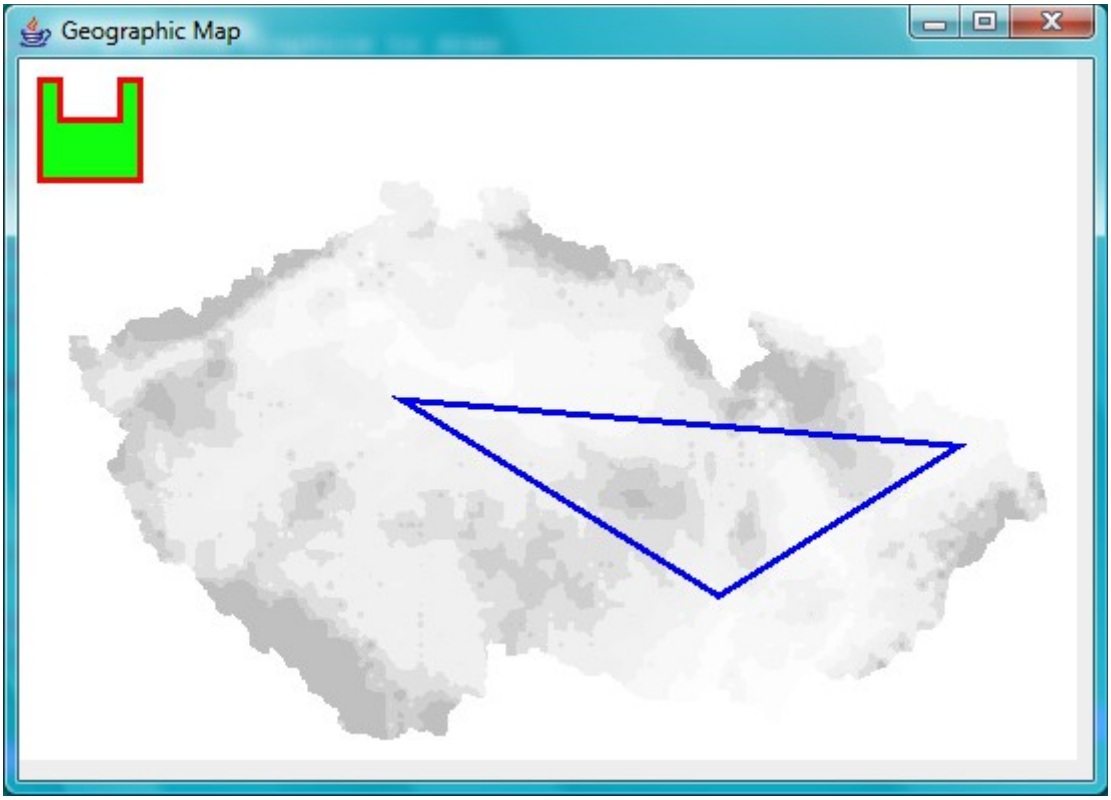
4.11.4. Polygon

Draw segmented **closed** line, specifiedl by points. First example connect cities Prague, Brno and Ostrava with line defined by given parameters (S-42), second one draw image, parameters are in pixels.

```
<Shape class="idea.graphics.shapes.simple.Polygon" drawColor="#0000FF" lineWidth="3">
	<S42GeoPosition x="3458912" y="5550917"/>
	<S42GeoPosition x="3616717" y="5452708"/>
	<S42GeoPosition x="3736220" y="5527766"/>
</Shape>

<Shape class="idea.graphics.shapes.simple.Polygon" drawColor="#FF0000" fillColor="#0FFF0F"
lineWidth="3">
	<CartesianPosition x="10" y="10"/>
	<CartesianPosition x="10" y="60"/>
	<CartesianPosition x="60" y="60"/>
	<CartesianPosition x="60" y="10"/>
	<CartesianPosition x="50" y="10"/>
	<CartesianPosition x="50" y="30"/>
	<CartesianPosition x="20" y="30"/>
	<CartesianPosition x="20" y="10"/>
</Shape>
```

4.11.4.1 Attributes

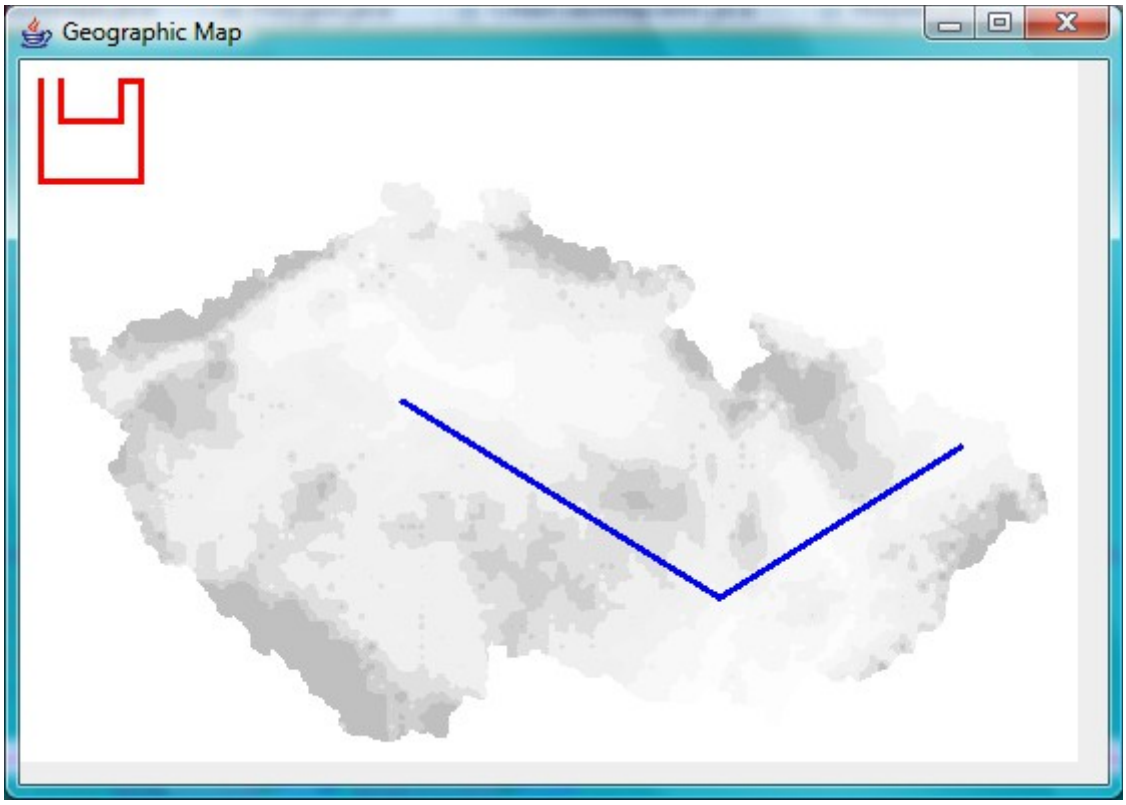
- **drawColor** – color of line
- **fillColor** – non-mantatory fill color, if not present, object body is not filled
- **lineWidth** – line width in pixels



4.11.5. Polyline

Draw segmented **open** line, specifiedl by points. First example connect cities Prague and Brno with line defined by given parameters (S-42), second one draw image, parameters are in pixels.

```
<Shape class="idea.graphics.shapes.simple.Polyline" drawColor="#0000FF" lineWidth="3">
       <S42GeoPosition x="3458912" y="5550917"/>
       <S42GeoPosition x="3616717" y="5452708"/>
       <S42GeoPosition x="3736220" y="5527766"/>
</Shape>

<Shape class="idea.graphics.shapes.simple.Polyline" drawColor="#FF0000" lineWidth="3">
       <CartesianPosition x="10" y="10"/>
       <CartesianPosition x="10" y="60"/>
       <CartesianPosition x="60" y="60"/>
       <CartesianPosition x="60" y="10"/>
       <CartesianPosition x="50" y="10"/>
       <CartesianPosition x="50" y="30"/>
       <CartesianPosition x="20" y="30"/>
       <CartesianPosition x="20" y="10"/>
</Shape>
```



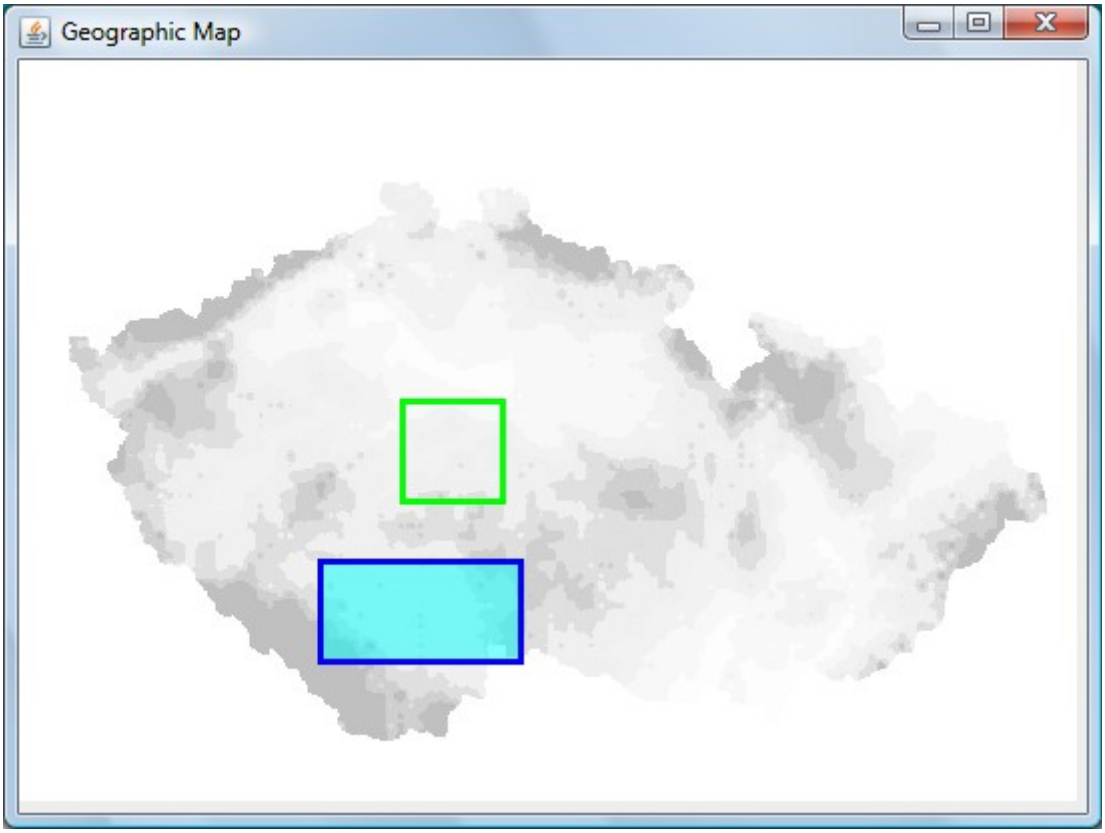
4.11.6. Rectangle

Draw rectangle with defined parameters. Require position of left top corner and dimension. Example show square with position 0-0 in Prague and size 50 km (defined in S-42) and semi-transcluent rectangle (defined in pixels).

```
<Shape class="idea.graphics.shapes.simple.Rectangle" drawColor="#00FF00" lineWidth="3"
rendered="true">
        <CartesianGeoPosition id="position" x="3458912" y="5550917"/>
        <CartesianGeoDimension id="dimension" width="50000" height="50000"/>
</Shape>

<Shape class="idea.graphics.shapes.simple.Rectangle" drawColor="#0000FF" fillColor="#7F00FFFF"
lineWidth="3" rendered="true">
        <CartesianPosition id="position" x="150" y="250"/>
        <CartesianDimension id="dimension" width="100" height="50"/>
</Shape>
```

4.11.6.1 Attributes

- **drawColor** – color of line
- **fillColor** – non-mantatory fill color, if not present, object body is not filled
- **lineWidth** – line width in pixels



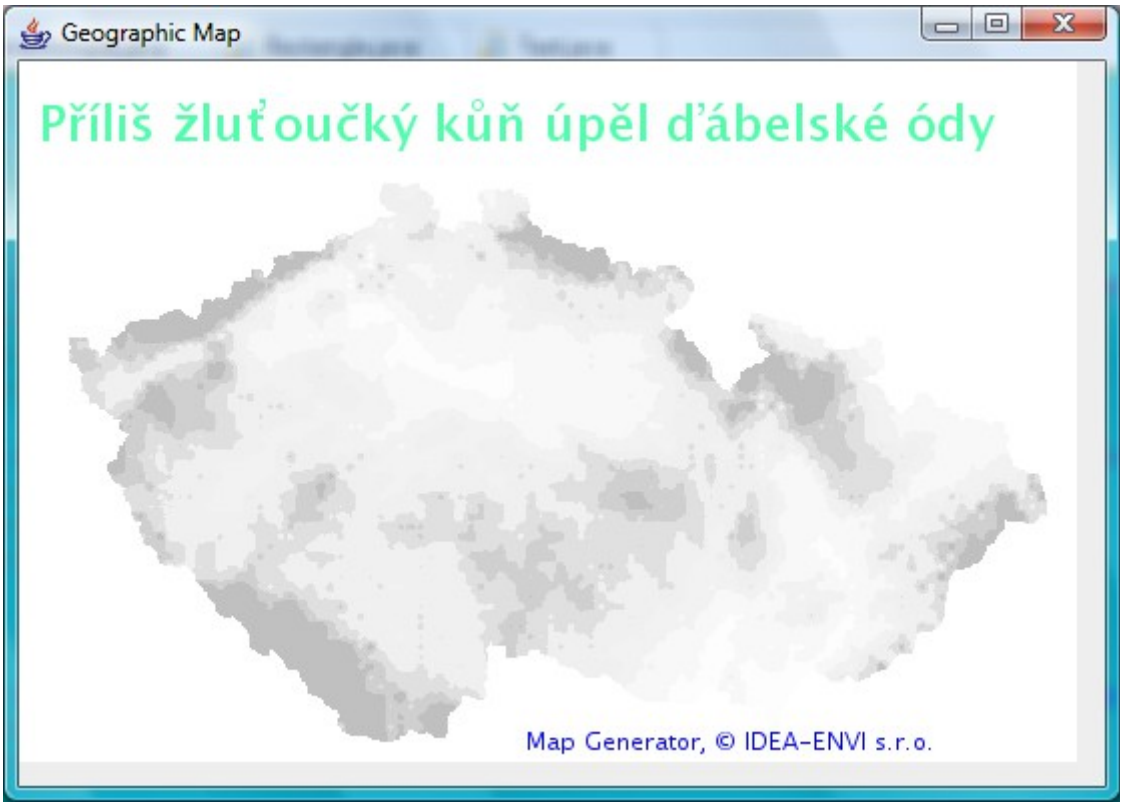
4.11.7. Text

Draw specified text on defined position.

```
<Shape class="idea.graphics.shapes.simple.Text" drawColor="#0000FF" text="Map Generator, © IDEA-ENVI s.r.o." font="Lucida Sans Unicode-PLAIN-12">
        <S42GeoPosition id="position" x='3520000' y='5376000'/>
</Shape>

<Shape class="idea.graphics.shapes.simple.Text" drawColor="#50FFAB" text="Příliš žluťoučký kůň úpěl ďábelské ódy" font="Lucida Sans Unicode-BOLD-25">
        <CartesianPosition id="position" x='10' y='40'/>
</Shape>
```

4.11.7.1 Attributes

- **drawColor** – text color
- **text** – Text string
- **font** – font definition



4.12. Compound graphics objects - general

Namespace **idea.graphics.shapes.complex** contain classes for drawing objects, that are composed from other primitive objects.

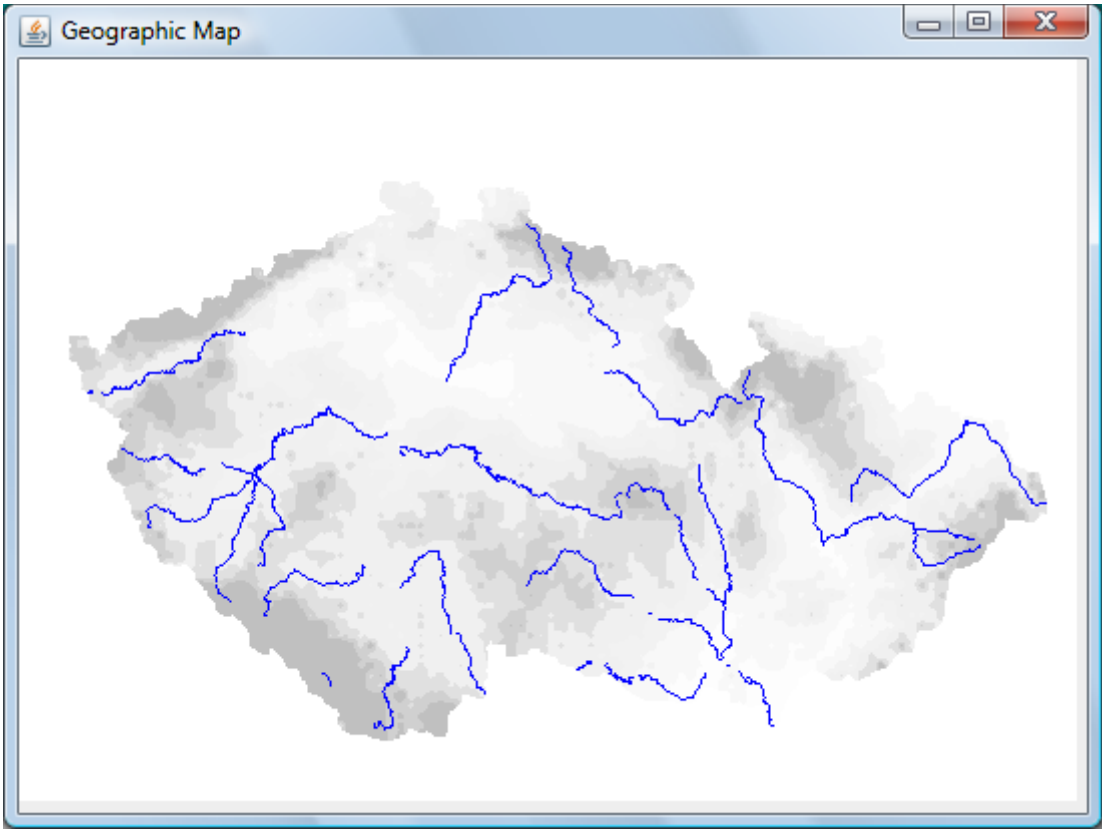
4.12.1. PolylinesFromGeoDataFile

Draw segmented lines, with points defined in text file (export format of ArcView).

```
<Shape class="idea.graphics.shapes.complex.PolylinesFromGeoDataFile"
fileName="./geodata/cz_rivers.txt" drawColor="#0000FF" lineWidth="1" rendered="true"/>
```

4.12.1.1 Attributes

- **fileName** – path to file with data
- **drawColor** – line color
- **lineWidth** – line width in pixels



This shape can be used for drawing borders, rivers, ways and anothers geographic objects.

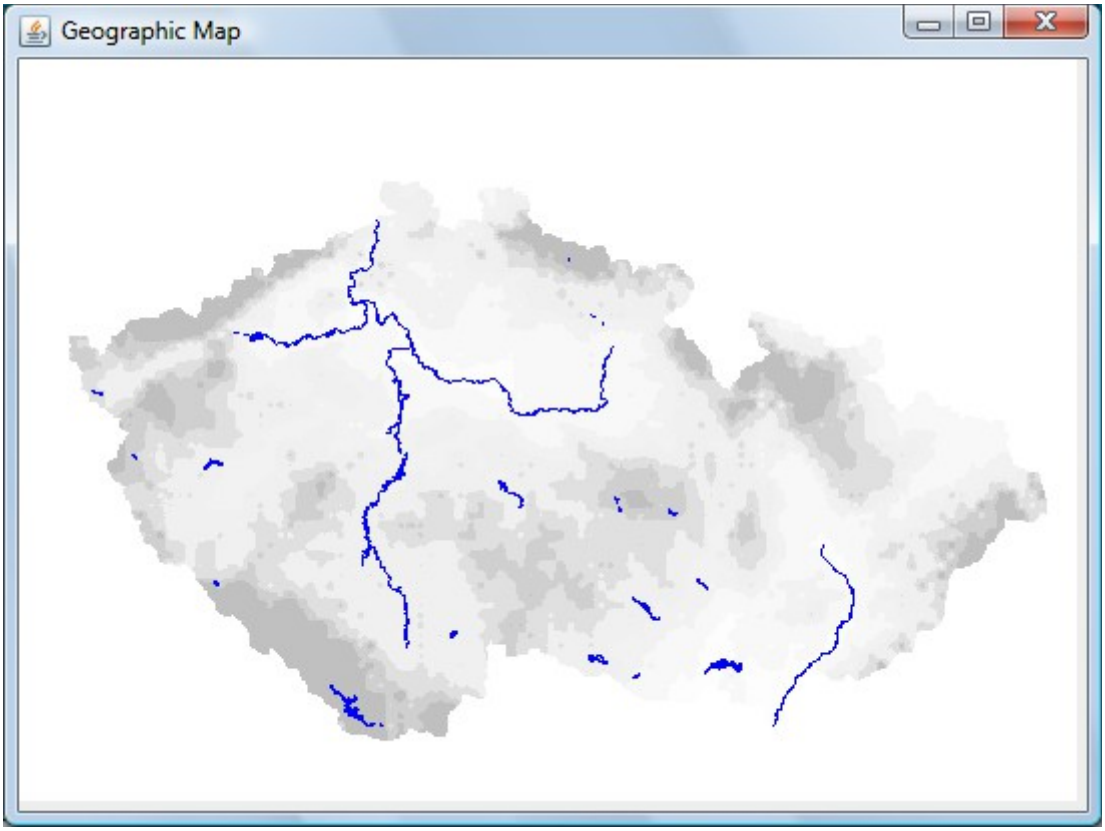
4.12.2. PolygonsFromGeoDataFile

Draw closed lines, with points defined in text file (export format of ArcView).

```
<Shape class="idea.graphics.shapes.complex.PolygonsFromGeoDataFile"
fileName="./geodata/cz_lakes.txt" drawColor="#0000FF" lineWidth="1" rendered="true"/>
```

4.12.2.1 Attributes

- **fileName** – path to file with data
- **drawColor** – line and body color
- **fillColor** – non-mantatory fill color, if not present, object body is not filled
- **lineWidth** – line width in pixels



This shape can be used for drawing lakes or other geographic areas.

4.12.3. ListColorScaleLegend

Draw legend for list color scale on defined position and size. See sample runset **runsets\Rain2005_a.xml** to learn how to use list color scale and legend for it.

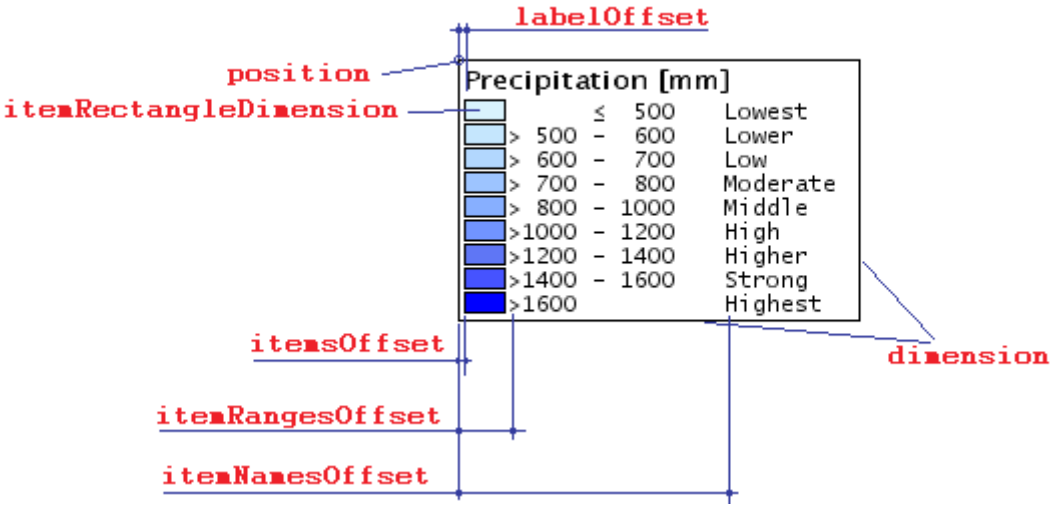
```
<Shape class="idea.graphics.shapes.complex.ListColorScaleLegend" frame="false"
numberFormatPattern="####" mainLabelFont="Lucida Sans Unicode-BOLD-13" itemLabelsFont="Lucida
Console-PLAIN-12">
       <CartesianGeoPosition id="position" x="3660000" y="5715000"/>
       <CartesianGeoDimension id="dimension" width="200000" height="130000"/>
       <CartesianDimension id="itemRectangleDimension" width="20" height="10"/>
       <CartesianOffset id="labelOffset" x="3" y="15"/>
       <CartesianOffset id="itemsOffset" x="3" y="20"/>
       <CartesianOffset id="itemRangesOffset" x="22" y="10"/>
       <CartesianOffset id="itemNamesOffset" x="130" y="10"/>
</Shape>
```

4.12.3.1 Attributes

- **mainLabelFont -** font for main label
- **itemLabelsFont** - font for item labels
- **frame** - non mandatory, true if required frame around legend, default is false.
- **numberFormatPattern** – pattern for formating numbers in scale items. If not specified, pattern for default locale is used.
- **noDataItem** - non mandatory, true if item for NODATA required, default is false.
- **itemNames** - non mandatory, true if required to draw item names, default is true.

4.12.3.2 Child objects

- **Position id="position"** – legend position
- **Dimension id="dimension"** – legend size
- **Dimension id="itemRectangleDimension"**– items rectangle size
- **Offset id="labelOffset"** – offset for positioning main label
- **Offset id="itemsOffset"** – offset for positioning items
- **Offset id="itemRangesOffset"** – offset for positioning labels with ranges
- **Offset id="itemNamesOffset"** – offset for positioning labels with names


labelOffset
position
itemRectangleDimension
Precipitation [mm]
≤ 500 Lowest
> 500 - 600 Lower
> 600 - 700 Low
> 700 - 800 Moderate
> 800 - 1000 Middle
>1000 - 1200 High
>1200 - 1400 Higher
>1400 - 1600 Strong
>1600 Highest
itemsOffset
dimension
itemRangesOffset
itemNamesOffset

4.12.4. LinearColorScaleLegend

Draw legend for linear color scale on defined position and size. See sample runset **runsets\Rain2005_b.xml** to learn how to use linear color scale and legend for it.

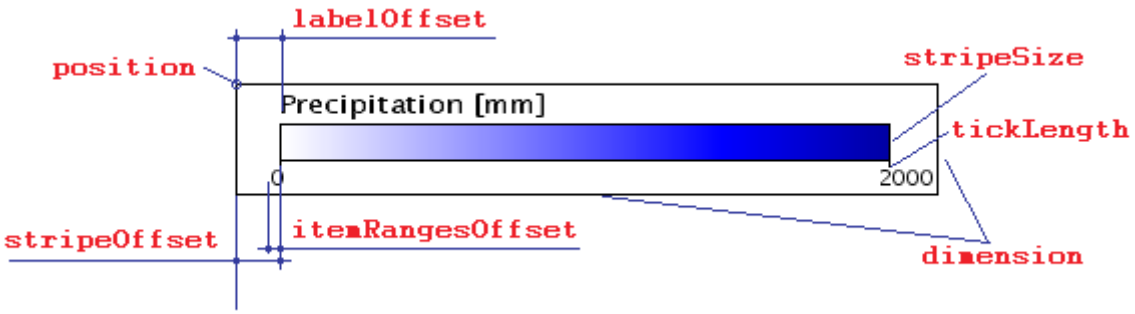
```
<Shape class="idea.graphics.shapes.complex.LinearColorScaleLegend" frame="true" stripeSize="16"
orientation="HORIZONTAL" numberFormatPattern="####" mainLabelFont="Lucida Sans Unicode-BOLD-13"
itemLabelsFont="Lucida Console-PLAIN-12">
       <CartesianGeoPosition id="position" x="3500000" y="5715000"/>
       <CartesianGeoDimension id="dimension" width="350000" height="055000"/>
       <CartesianOffset id="labelOffset" x="22" y="15"/>
       <CartesianOffset id="stripeOffset" x="22" y="20"/>
       <CartesianOffset id="itemRangesOffset" x="-5" y="30"/>
</Shape>
```

4.12.4.1 Attributes

- **orientation** – can be HORIZONTAL or VERTICAL
- **mainLabelFont -** font for main label
- **itemLabelsFont** - font for range labels
- **frame** - non mandatory, true if required frame around legend, default is false.
- **numberFormatPattern** – pattern for formating numbers in scale items. If not specified, pattern for default locale is used.

4.12.4.2 Child objects

- **Position id="position"** – legend position
- **Dimension id="dimension"** – legend dimension
- **Offset id="labelOffset"** – offset for positioning main label
- **Offset id="stripeOffset"** – offset for positioning color stripe
- **Offset id="itemRangesOffset"** – offset for positioning labels with ranges



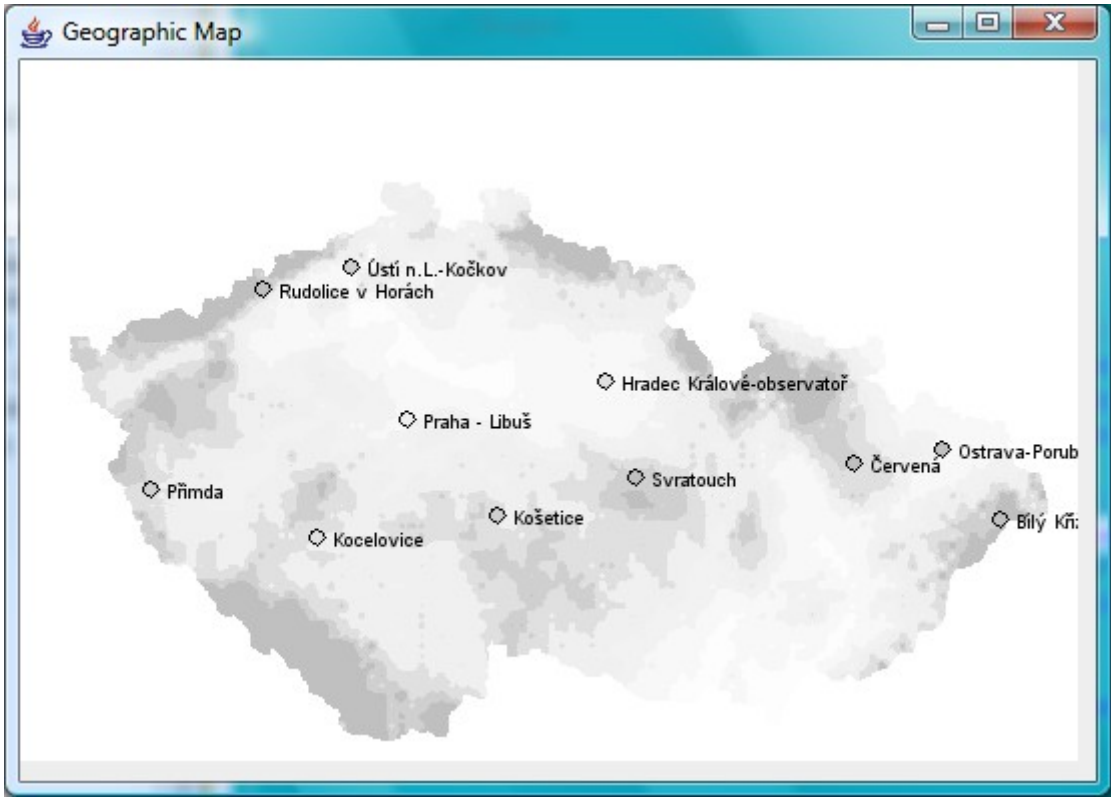
4.12.5. Stations

Draw measuring stations (places where are obtained input values) from defined input matrix. Marks are filled with color that depends on value, what station obtained. This object can be used only if we have at least one computed matrix in Runset.

```
<Shape class="idea.graphics.shapes.complex.Stations" matrix="rain" type="circle"/>
```

4.12.5.1 Attributes

- **matrix** – computed matrix id, whose input values will be painted as measuring stations.
- **type** – mark type: CIRCLE, SQUARE
- **stationNames** – true or false, if you wich to render names of stations. Nonmandatory, default is true.
- **size** – size extension of circle or rectangle, nonmandatory, user may use it to resize default mark dimension.



4.13. Compound graphics objects - geographic

Namespace **idea.graphics.shapes.complex.geo** contains classes for drawing objects from geographics area.

4.13.1. City

Draw city with specified size and name to defined position. Aditionally define offset for text with name against city position.

```
<Shape class="idea.graphics.shapes.complex.geo.City" name="Zubří" size="SMALL" markColor="#0000FF"
textColor="#FF0000">
       <S42GeoPosition id="position" x='3723334' y='5485467'/>
       <CartesianOffset id="labelOffset" x="5" y="5"/>
</Shape>

<Shape class="idea.graphics.shapes.complex.geo.City" name="Schwartenberg" size="MEDIUM"
markColor="#0F0000" textColor="#0000FF">
       <S42GeoPosition id="position" x='3391578' y='5615421'/>
       <CartesianOffset id="labelOffset" x="8" y="8"/>
</Shape>
```

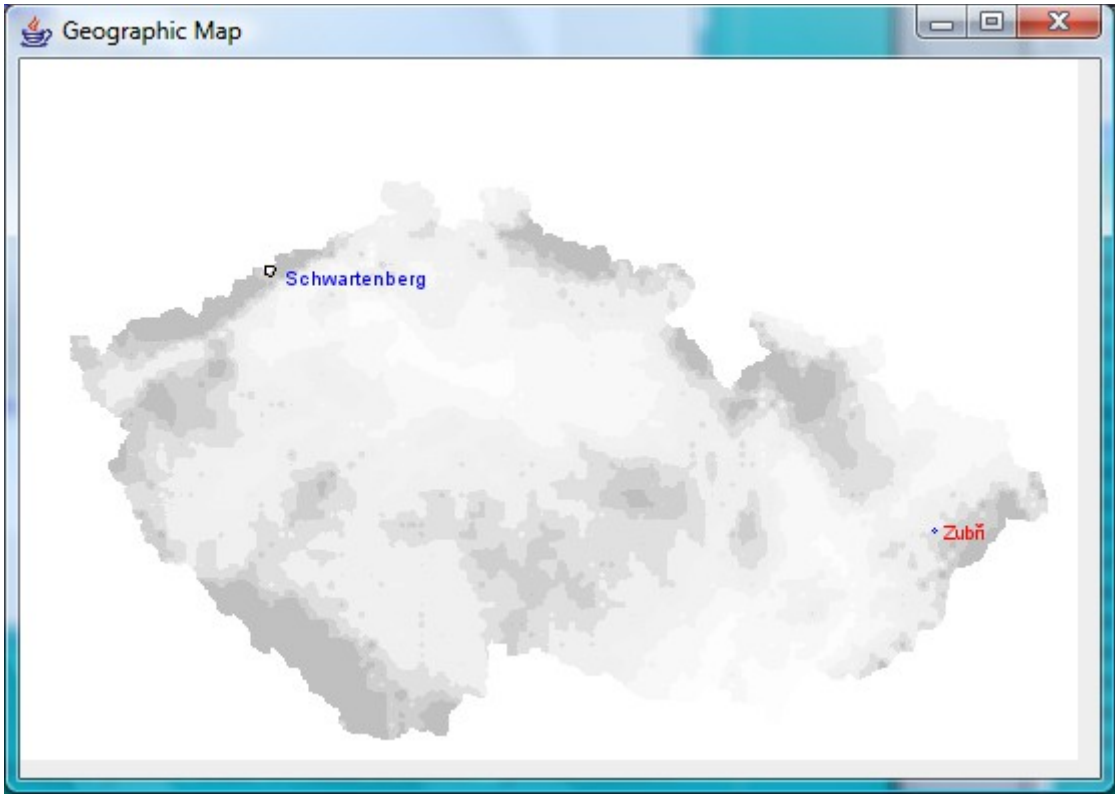
4.13.1.1 Attributes

- **name** – city name
- **size** – city size. Values: LARGE, MEDIUM, SMALL
- **markColor** – color of city mark - circle (non-mandatory)
- **textColor** – text color (non-mandatory)

4.13.1.2 Child objects

- **Position id="position"** – shape position
- **Offset id="labelOffset"** – offset for positioning label

Map Generator



4.13.2. NorthArrow

The purpose of the north arrow is for orientation. This allows the viewer to determine the direction of the map as it relates to due north.

```
<Shape class="idea.graphics.shapes.complex.geo.NorthArrow" lineWidth="1" text="North" font="Lucida
Sans Unicode-PLAIN-10" markFillColor="#FFFFFF" rendered="true">
     <CartesianPosition id="position" x='20' y='20'/>
     <CartesianDimension id="dimension" width="18" height="30"/>
     <CartesianOffset id="labelOffset" x="-5" y="40"/>
     <CartesianOffset id="arrowsOffset" x="-3" y="-2"/>
</Shape>

<Shape class="idea.graphics.shapes.complex.geo.NorthArrow" lineWidth="1" text="South-West"
angle="225" font="Lucida Sans Unicode-PLAIN-10" markFillColor="#FFFFFF" rendered="true">
     <CartesianPosition id="position" x='100' y='50'/>
     <CartesianDimension id="dimension" width="18" height="30"/>
     <CartesianOffset id="labelOffset" x="-10" y="10"/>
     <CartesianOffset id="arrowsOffset" x="-3" y="-5"/>
</Shape>
```

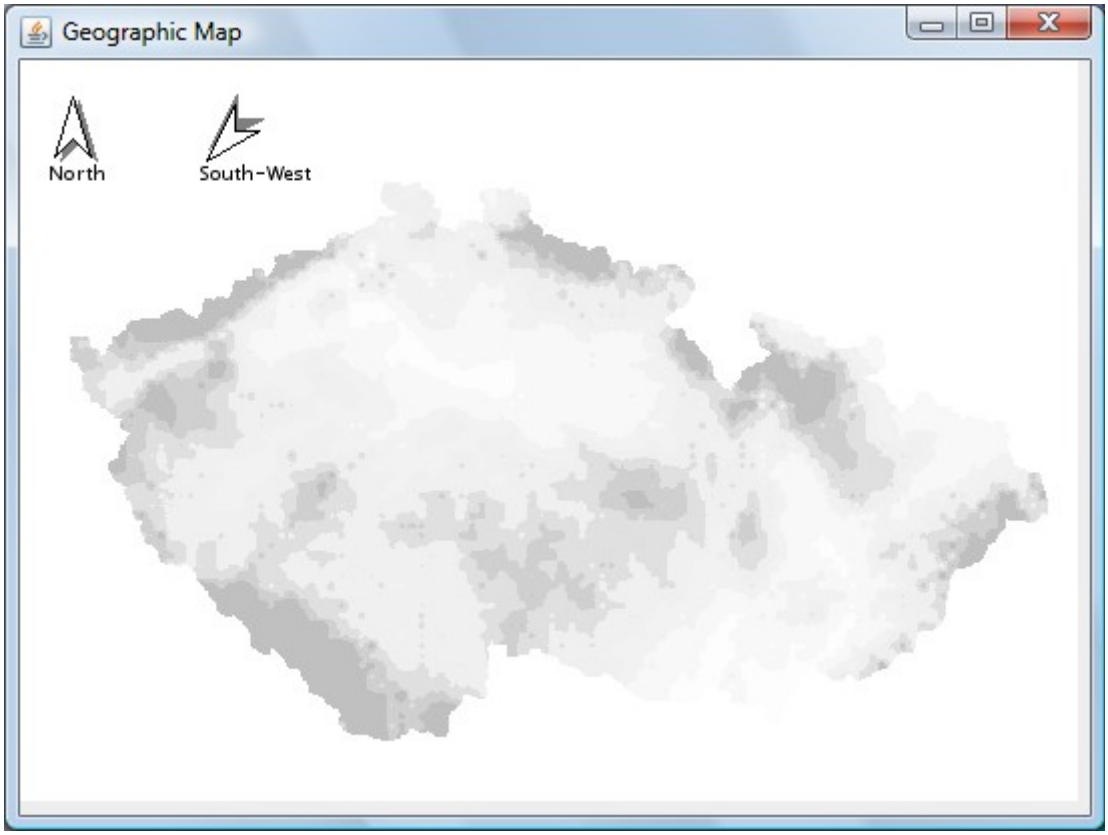
4.13.2.1 Atributes

- **lineWidth** – outline line width
- **text –** label text, e.g. 'North'
- **angle** – rotation angle, default is 0 degrees
- **font –** font used for text
- **markDrawColor –** color used for arrow outline, default is black
- **markFillColor –** color used for arrow body, default is white
- **markShadowColor –** color osed for shadow arrow, default is gray

4.13.2.2 Child objects

- **Position, id=“position“ -** defines position of shape
- **Dimension, id=“dimension“ -** defines size of shape
- **Offset, id=“arrowsOffset“ -** offset between main and shadow arrows
- **Offset, id=“labelOffset“ -** offset for precise positioning label with text, against rest of shape

Map Generator



4.13.3. WayMarker

The purpose of the way marker is to sign motorways or highways.

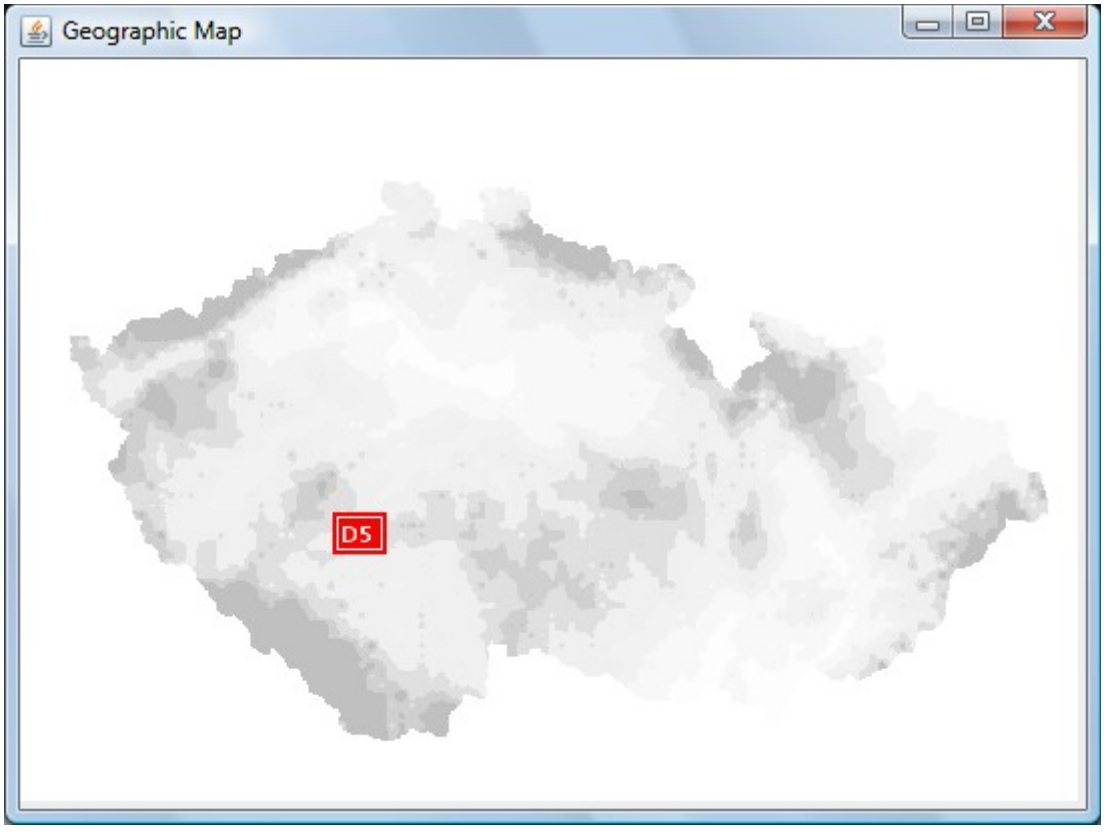
```
<Shape class="idea.graphics.shapes.complex.geo.WayMarker" drawColor="#FF0000" fillColor="#FF0000"
outlineColor="#FFFFFF" text="D5" font="Lucida Console-BOLD-12">
        <CartesianPosition id="position" x="156" y="226"/>
        <CartesianDimension id="dimension" width="26" height="20"/>
        <CartesianOffset id="labelOffset" x="5" y="15"/>
</Shape>
```

4.13.3.1 Atributes

- **font –** font used for text label
- **outlineColor –** color used for rectangle inside mark and label
- **fillColor –** color used for body

4.13.3.2 Child objects

- **Position id="position"** – legend position
- **Dimension id="dimension"** – legend size
- **Offset, id="labelOffset" -** offset for precise positioning label with text, against rest of shape



4.13.4. MapResolutionScale

The scale explains the relationship of the map dimension to the real world.

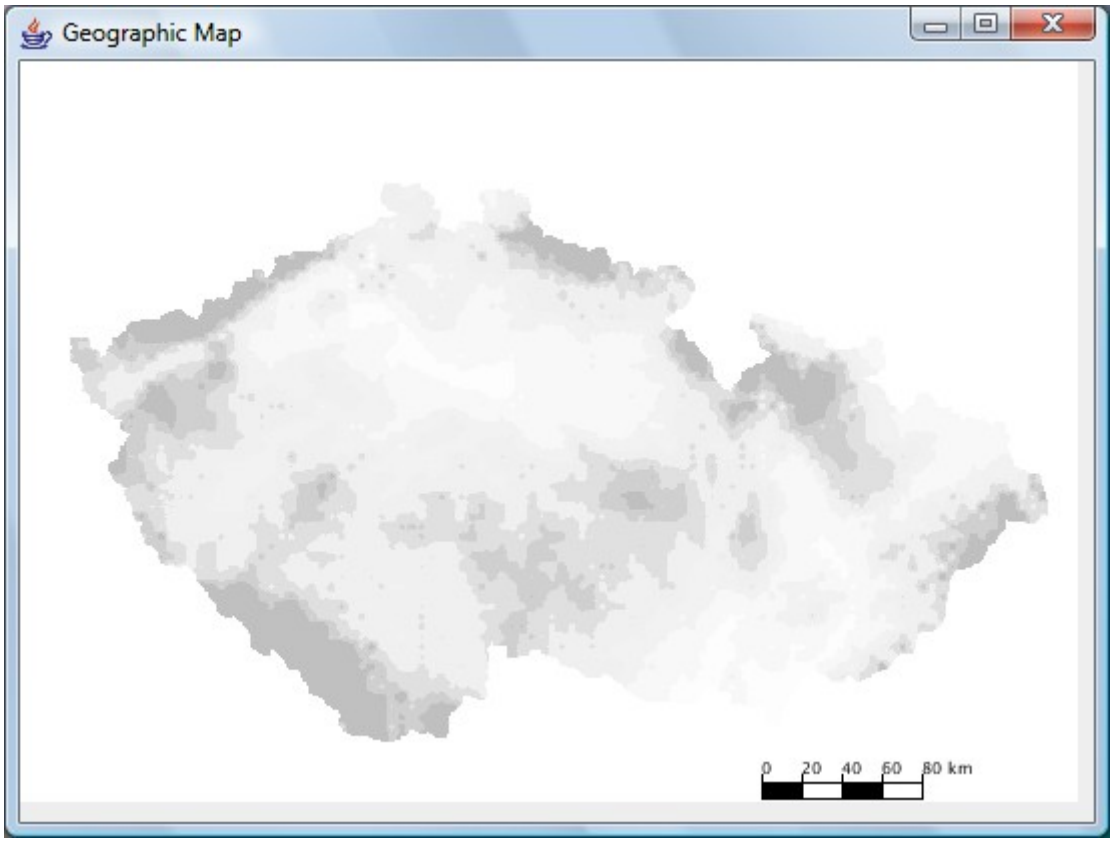
```
<Shape class="idea.graphics.shapes.complex.geo.MapResolutionScale" boxHeight="8" segments="4"
resolution="_20km" font="Lucida Sans Unicode-PLAIN-8">
       <CartesianPosition id="position" x='370' y='360'/>
</Shape>
```

4.13.4.1 Attributes

- **boxHeight** – defines segments height, default is 8 pixels
- **tickHeight** – height of vertical lines between segments, default is 4 pixels
- **segments** – number of scale segments
- **resolution** – segment width in meters – predefined values: „_1(or 2 or 5)00m“ - „_1(or 2 or 5)0000km“
- **font** – font definition, non mandatory

4.13.4.2 Child objects

- **Position, id=“position“ -** defines position of shape



4.14. Compound graphics objects - meteorological

Namespace **idea.graphics.shapes.complex.meteo** contains classes for drawing objects from meteorological area.

4.14.1. WindBarb

The wind barb indicates the wind direction and wind speed.

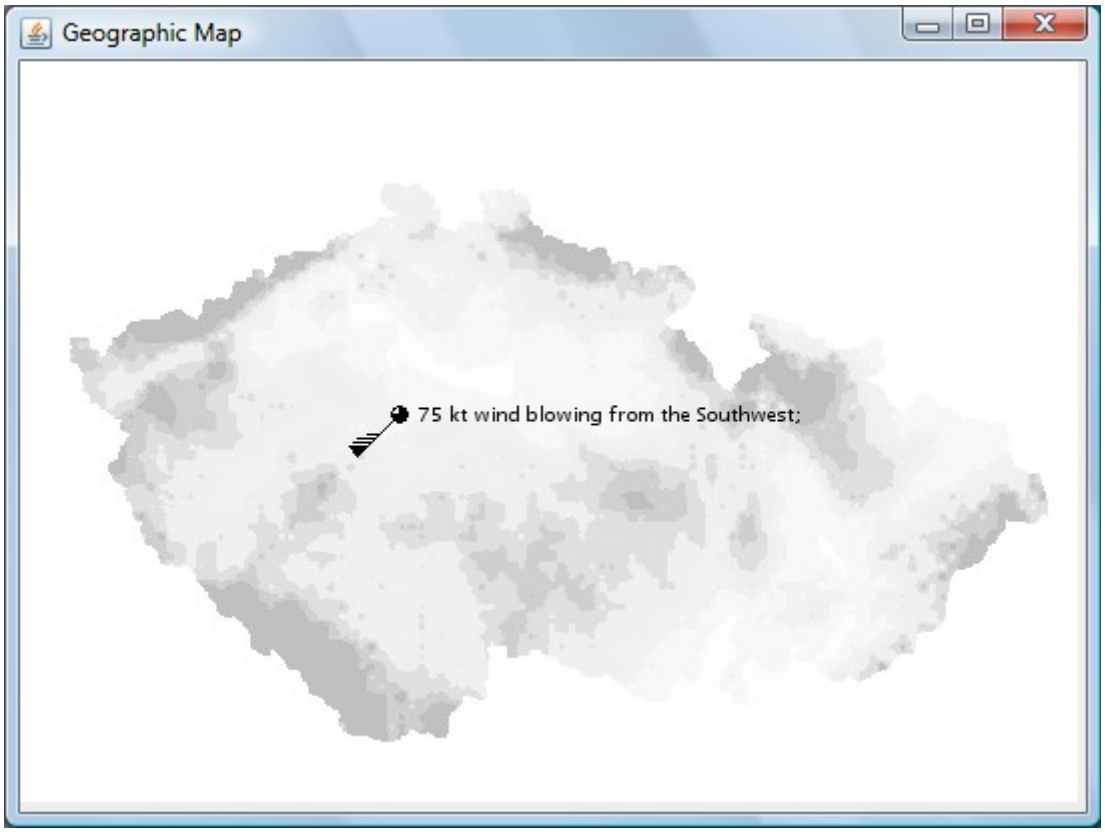
```
<Shape class="idea.graphics.shapes.complex.meteo.WindBarb" type="CLASSIC" skyCoverage="BROKEN"
windSpeed="75" speedUnit="KNOTS" windDirection="225" barbLength="6" lineWidth="1" text="75 kt wind
blowing from the Southwest;" font="Lucida Sans Unicode-PLAIN-10" outlineColor="#000000"
rendered="true">
    <CartesianPosition id="position" x='189' y='176'/>
    <CartesianDimension id="dimension" width="8" height="8"/>
    <CartesianOffset id="labelOffset" x="10" y="4"/>
</Shape>
```

4.14.1.1 Atributes

- **type** – at this time only CLASSIC type is supported
- **skyCoverage** – sky coverage symbol, one of: NONE, CLEAR, FEW, SCATTERED, BROKEN, OVERCAST
- **windSpeed** – speed of the wind
- **speedUnit** – Unit of the wind speed, one of: KNOTS, KM_PER_HOUR, M_PER_SECOND
- **windDirection** - direction (in degrees) that the wind is blowing **from**. This means that 45 degrees is Northeast wind
- **barbLength** - length of long barb in pixels (long barb is 10kt symbol)
- **font –** font used for text label
- **outlineColor –** color used for rectangle inside mark and label
- **fillColor –** color used for body
- **textColor –** color used for text label

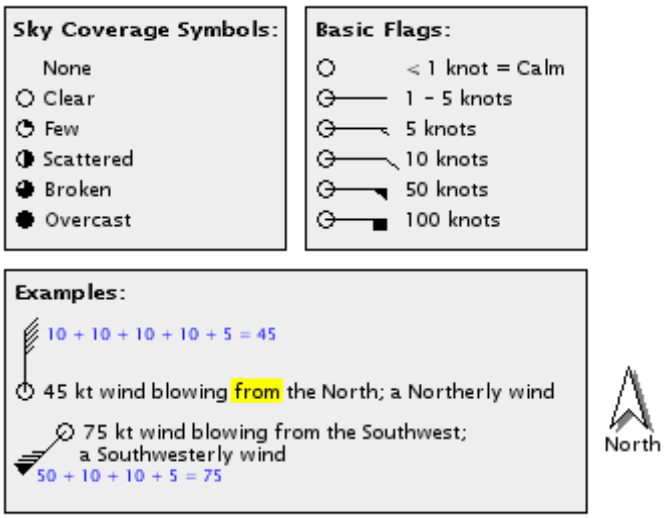
4.14.1.2 Child objects

- **Position id="position"** – legend position
- **Dimension id="dimension"** – size of sky coverage symbol
- **Offset, id="labelOffset" -** offset for precise positioning label with text, against rest of shape



4.14.1.3 The Wind Barb explained

Wind Barbs point in the direction **from** which the wind is blowing. Short barb represents 5 knots, each long barb 10 knots. Triangle pennant is 50 knots, square pennant is 100 knots.



4.15. Compound graphics objects - miscellaneous

Namespace **idea.graphics.shapes.complex.misc** contains classes for drawing miscellaneous objects.

4.15.1. AnalogClock

Displays simple clock with actual time.

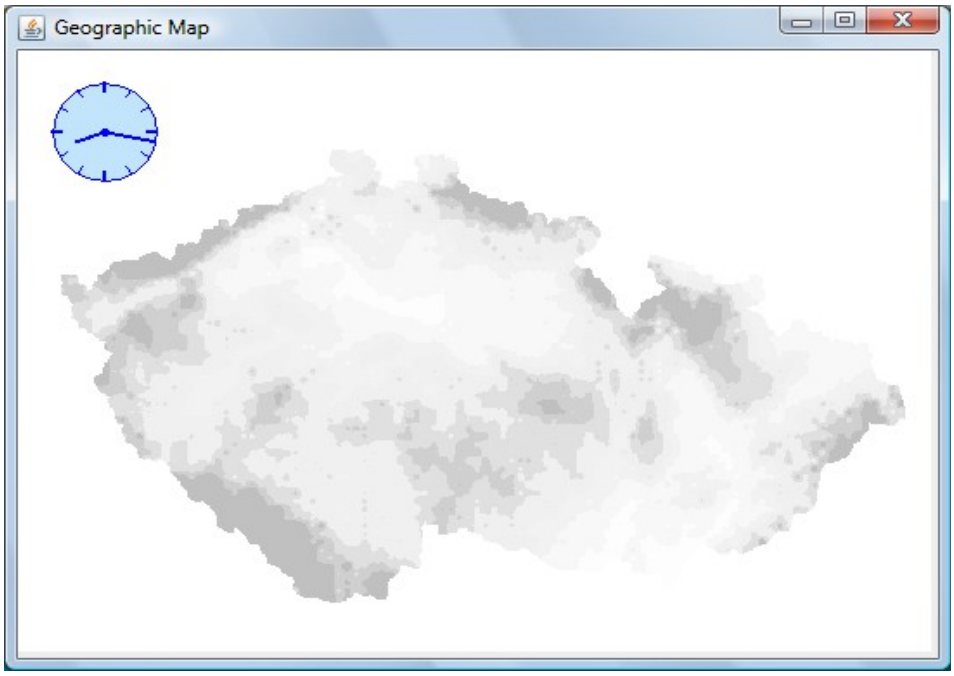
```
<Shape class="idea.graphics.shapes.complex.misc.AnalogClock" displayTime="CURRENT" timeZone="CET"
fillColor="#C2E4FD" outlineColor="#0000FF" minuteHandColor="#0000FF" hourHandColor="#0000FF"
rendered="true">
    <CartesianPosition id="position" x='20' y='20'/>
    <CartesianDimension id="dimension" width="60" height="60"/>
</Shape>
```

4.15.1.1 Atributes

- **displayTime -** Values: CURRENT or DATE_VALUES
- **timeZone** – time zone used for clock
- **font –** font used for text label
- **outlineColor –** color used for circle aroud clock
- **fillColor –** color used for body
- **minuteHandColor –** color used for minute hand
- **hourHandColor –** color used for hour hand

4.15.1.2 Child objects

- **Position id="position"** – clock position
- **Dimension id="dimension"** – clock size



4.16. Compound graphics objects – geographic, Czech Republic

Namespace **idea.graphics.shapes.complex.geo.czech** contains classes for drawing objects from geographics area of the Czech Republic.

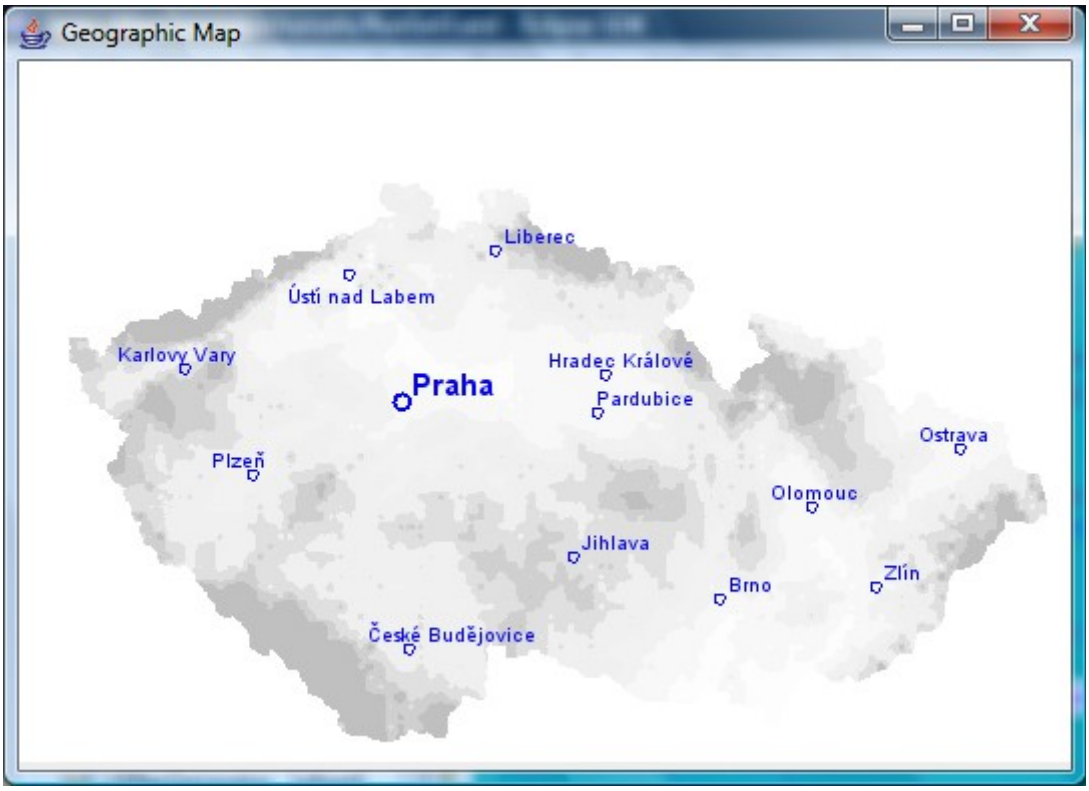
4.16.1. CitiesCzechRepublic

Draw significant cities in Czech Republic. Cities are divided into categories LARGE – Prague, MEDIUM – regional, SMALL - others

```
<Shape class="idea.graphics.shapes.complex.geo.czech.CitiesCzechRepublic" sizeLimit="MEDIUM"
markColor="#0000FF"/>
```

4.16.1.1 Attributes

- **sizeLimit** – draw only in this category and larger. Values: LARGE, MEDIUM, SMALL
- **markColor** – color of city mark - circle (non-mandatory)
- **textColor** – text color (non-mandatory)



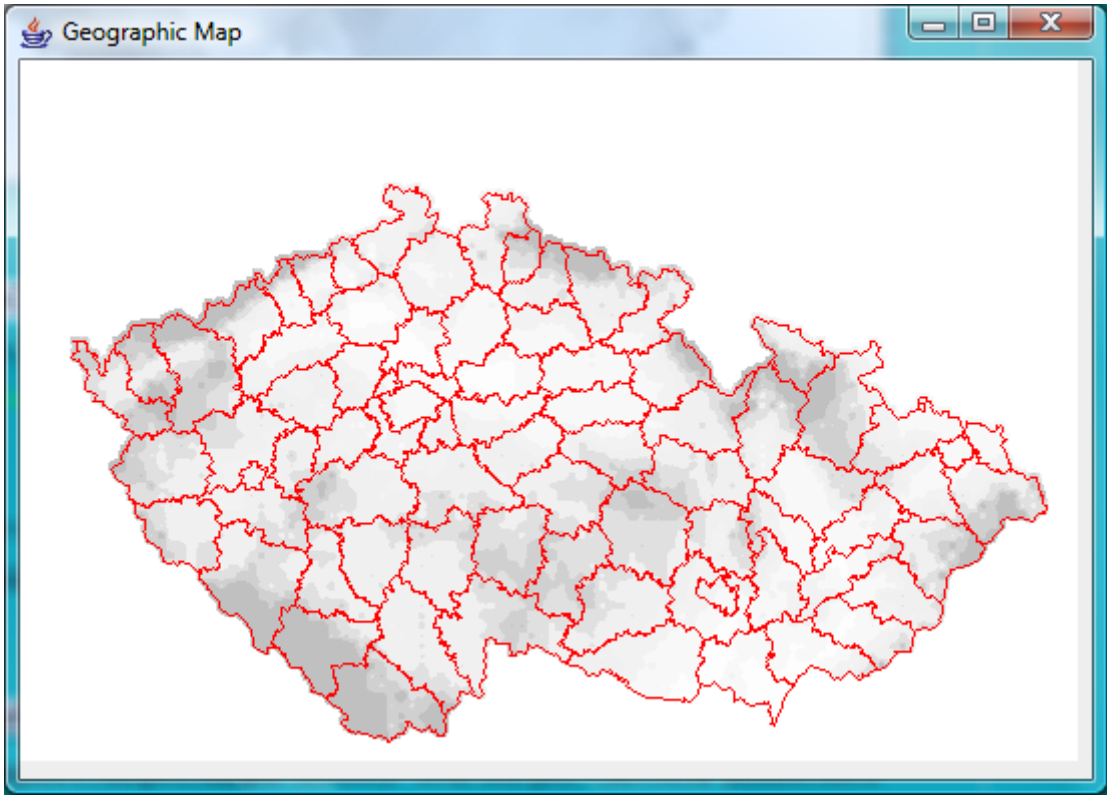
4.16.2. DistrictsCzechRepublic

Draw districts borders in Czech Republic.

```
<Shape class="idea.graphics.shapes.complex.geo.czech.DistrictsCzechRepublic" drawColor="#FF0000"/>
```

4.16.2.1 Attributes

- **drawColor** – line color
- **lineWidth** – line width in pixels



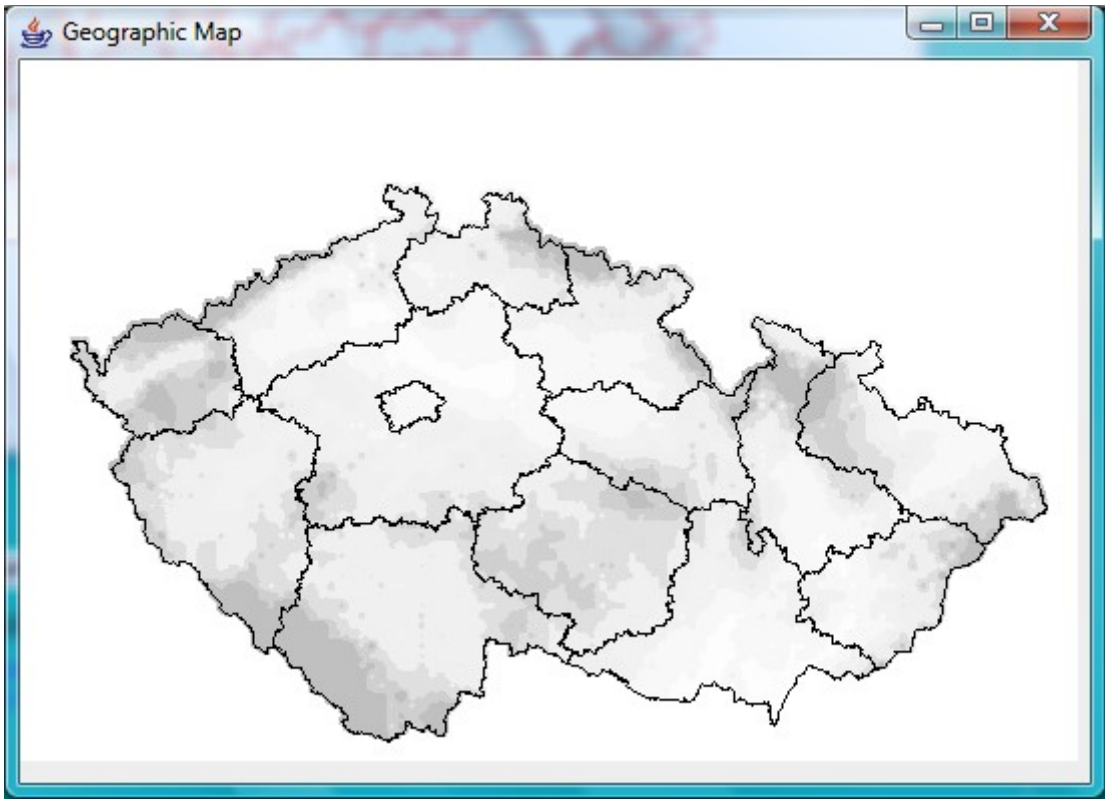
4.16.3. RegionsCzechRepublic

Draw regions borders in Czech Republic.

```
<Shape class="idea.graphics.shapes.complex.geo.czech.RegionsCzechRepublic" drawColor="#000000"
lineWidth="3" />
```

4.16.3.1 Attributes

- **drawColor** – line color
- **lineWidth** – line width in pixels



4.17. Filters

Filters must have defined **Position** of left-top corner and **Dimension** for their working area.

Filters and Shapes can be either ordered, so it's possible to paint shape, apply filter and then repaint this area.

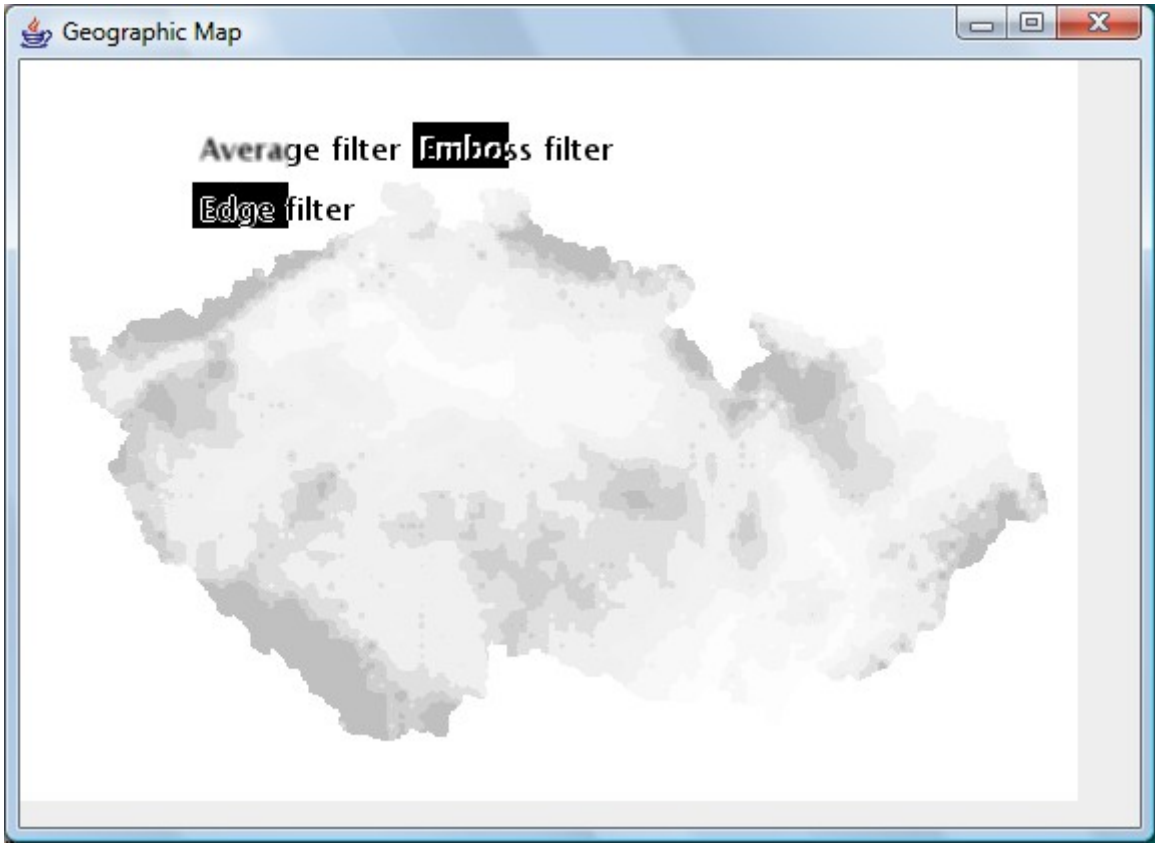
4.17.1. Convolution

This filter can be used for blur or another operations.

```
<Shape class="idea.graphics.shapes.simple.Text" drawColor="#000000" text="Average filter"
font="Lucida Sans Unicode-BOLD-15">
     <CartesianPosition id="position" x="90" y="50"/>
</Shape>

<Filter class="idea.graphics.filter.Convolution" filter="AVERAGE">
    <CartesianPosition id="position" x='85' y='30'/>
    <CartesianDimension id="dimension" width="50" height="25"/>
</Filter>
```

4.17.1.1 Attributes

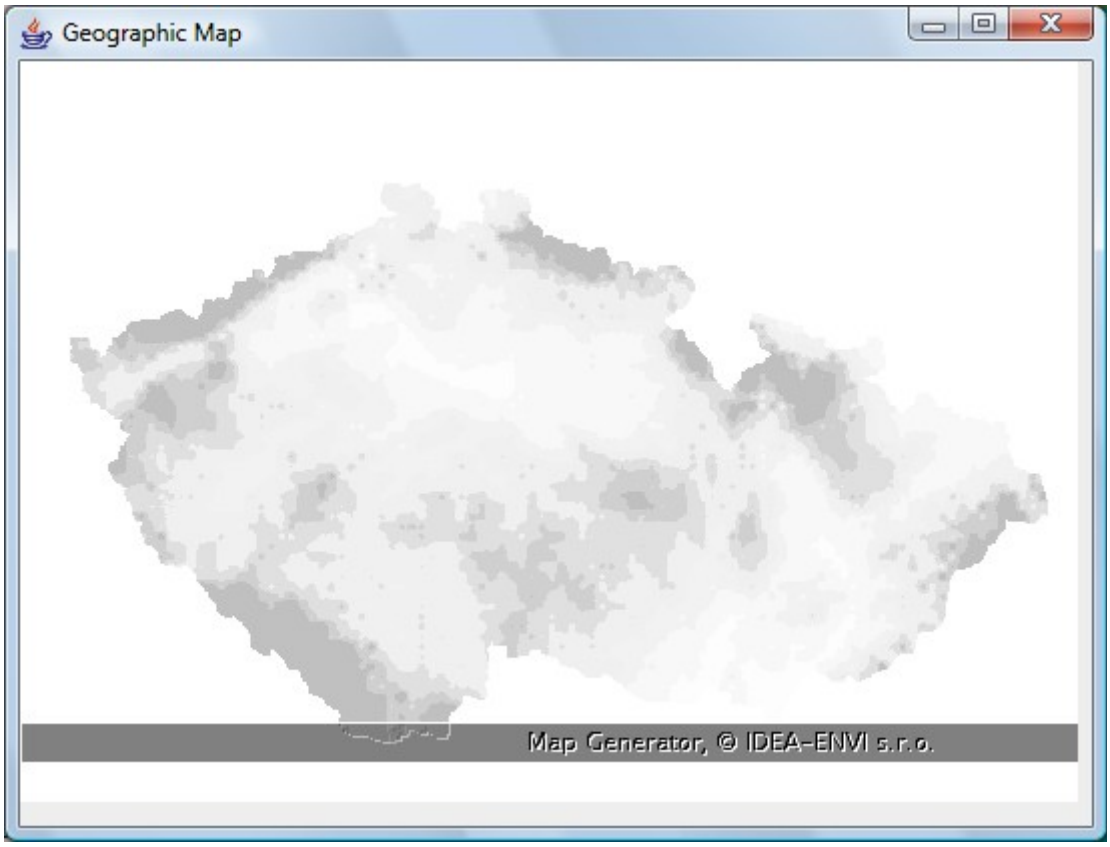
- **filter** – filter pattern, one of: EDGE, SHARPEN, AVERAGE, EMBOSS.



4.17.2. Emboss

This filter create gray-scale embossment.

```
<Shape class="idea.graphics.shapes.simple.Text" drawColor="#0000FF" text="Map Generator, © IDEA-ENVI s.r.o." font="Lucida Sans Unicode-PLAIN-12">
     <S42GeoPosition id="position" x='3520000' y='5376000'/>
</Shape>

<Filter class="idea.graphics.filter.Emboss">
     <CartesianPosition id="position" x='0' y='330'/>
     <CartesianDimension id="dimension" width="528" height="20"/>
</Filter>
```



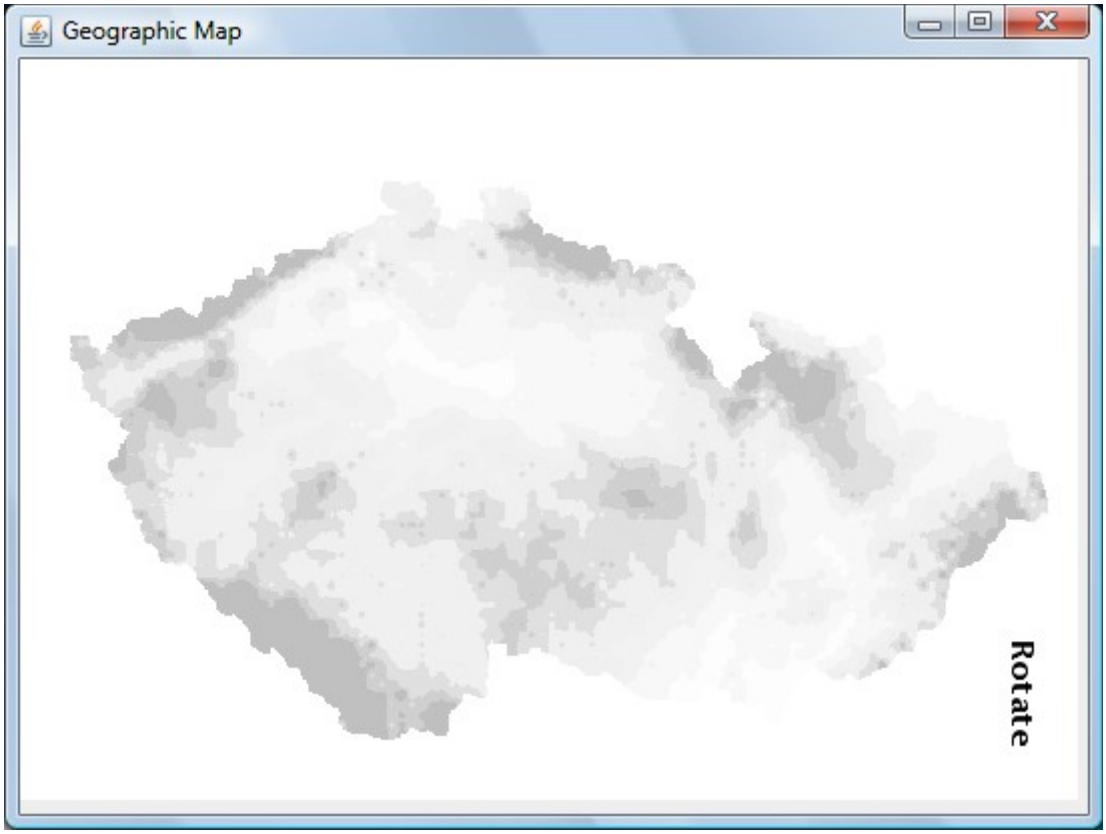
4.17.3. Rotate

This filter rotate it's area by specified angle.

```
<Shape class="idea.graphics.shapes.simple.Text" drawColor="#000000" text="Rotate" font="Lucida Sans Unicode-BOLD-15">
        <CartesianPosition id="position" x="450" y="305"/>
</Shape>

<Filter class="idea.graphics.filter.Rotate" rendered="true">
        <CartesianPosition id="position" x='450' y='290'/>
        <CartesianDimension id="dimension" width="60" height="60"/>
</Filter>
```

4.17.3.1 Attributes

- **angle** – rotation angle, if not specified, default is 90 degrees



5. Runset samples

This part contain explanation of two Runsets. You may find them in subdirectory **runsets** of module **MapGenerator.**

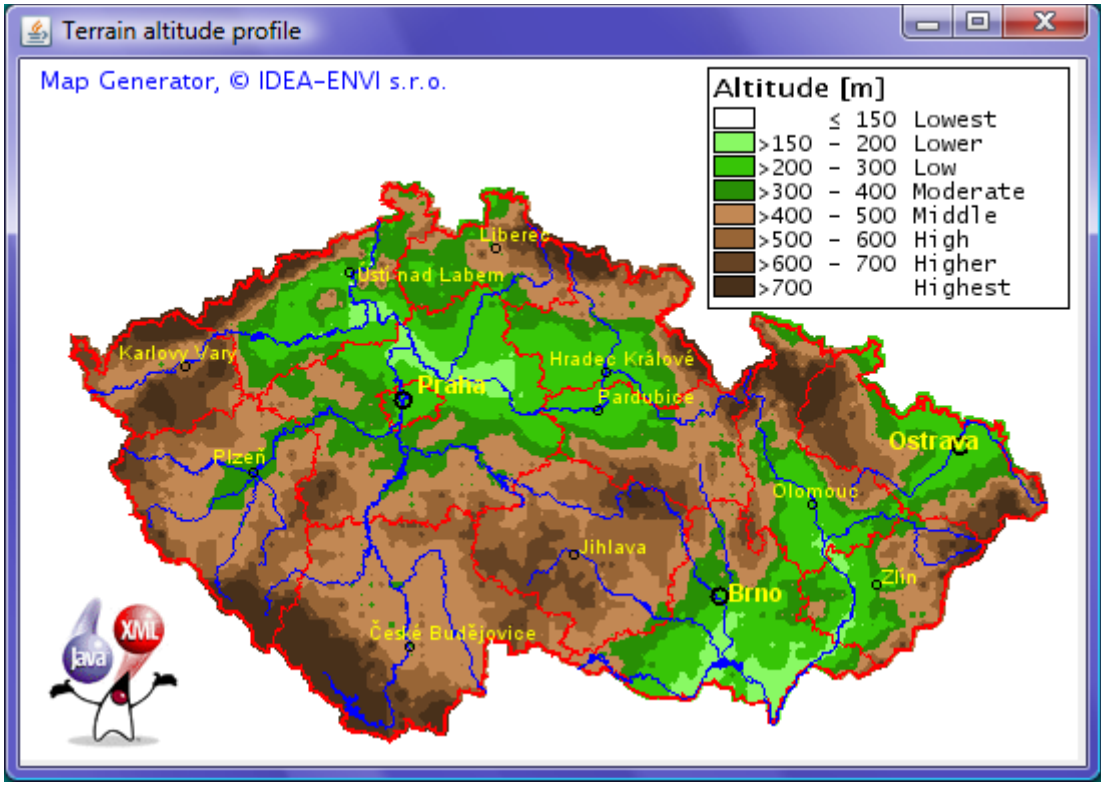
5.1. Generate map with terrain modeling

In subdirectory **runsets** there is file **Altitudes.xml** and in subdirectory **matrixes** there is **czech_hypsography.txt** – input matrix with fine smoothed terrain model of Czech Republic. In subdirectory **img** there are images.

Launch Runset from command line ...

```
java -jar MapGenerator.jar /F 3 log.txt .\runsets\Altitudes.xml
```

… and see this result:



In subdirectory output, there is created **Altitudes.png** with this map.

Note: this Runset **not have input values**, it only load input matrix and some shapes and render it. It contains no computations (interpolation or modification), so we may omit optimalization parameters for JVM and use directly jar file name.

5.1.1. Runset Altitudes.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<MapGenerator>
    <Description>Display terrain model</Description>
        <Matrixes rows="279" columns="488" cellSize="1000">
        <CartesianGeoPosition id="leftBottomCorner" x='3292000' y='5380000'/>

        <InputMatrix class="idea.map.matrix.InputMatrixFromFile" id="hypsography"
fileName="./matrixes/czech_hypsography.txt"/>
    </Matrixes>

    <Renderer class="idea.map.renderer.GeoImageMapRenderer" matrix="altitudes">
        <Shape class="idea.graphics.shapes.complex.ListColorScaleLegend" frame="true"
mainLabelFont="Lucida Sans Unicode-BOLD-13" itemLabelsFont="Lucida Console-PLAIN-12">
          <ColorForValueRange name="Lowest" color="#FFFFFF" valueFrom="0" valueTo="150" seq="1" />
          <ColorForValueRange name="Lower" color="#89F964" valueFrom="150" valueTo="200" seq="2"/>
          <ColorForValueRange name="Low" color="#37C507" valueFrom="200" valueTo="300" seq="3"/>
          <ColorForValueRange name="Low-Mid" color="#288F05" valueFrom="300" valueTo="400" seq="4"/>
          <ColorForValueRange name="Mid" color="#C28854" valueFrom="400" valueTo="500" seq="5"/>
          <ColorForValueRange name="High" color="#986536" valueFrom="500" valueTo="600" seq="6"/>
          <ColorForValueRange name="Higher" color="#664324" valueFrom="600" valueTo="700" seq="7"/>
          <ColorForValueRange name="Highest" color="#48301A" valueFrom="700" seq="8"/>
        </ColorScale>

        <GeographicMap scale="1000">
            <CartesianGeoPosition id="leftBottomCorner" x='3267000' y='5370000'/>
            <CartesianDimension id="dimension" width="528" height="350"/>

            <Shape class="idea.graphics.shapes.complex.ScaleLegend" font="Lucida Console-PLAIN-12"
frame="true">
                <S42GeoPosition id="position" x="3626000" y="5715000"/>
                <S42Dimension id="dimension" width="165000" height="120000"/>
                <CartesianDimension id="itemRectangleDimension" width="20" height="10"/>
                <CartesianOffset id="textOffset" x="22" y="10"/>
                <CartesianOffset id="nameOffset" x="110" y="10"/>
            </Shape>

             <Shape class="idea.graphics.shapes.complex.geo.czech.RegionsCzechRepublic"
drawColor="#FF0000" lineWidth="1" rendered="true"/>
            <Shape class="idea.graphics.shapes.complex.PolylinesFromGeoDataFile"
fileName="./geodata/czech_border.txt" lineWidth="1.5" drawColor="#FF0000" rendered="true"/>

            <Shape class="idea.graphics.shapes.complex.PolygonsFromGeoDataFile"
fileName="./geodata/cz_lakes.txt" drawColor="#0000FF" rendered="true"/>
            <Shape class="idea.graphics.shapes.complex.PolylinesFromGeoDataFile"
fileName="./geodata/cz_rivers.txt" drawColor="#0000FF" rendered="true"/>

            <Shape class="idea.graphics.shapes.simple.Text" drawColor="#0000FF" text="Map Generator,
© IDEA-ENVI s.r.o." font="Lucida Sans Unicode-PLAIN-12">
              <CartesianPosition id="position" x='10' y='15'/>
            </Shape>

            <Shape class="idea.graphics.shapes.complex.geo.czech.CitiesCzechRepublic"
sizeLimit="MEDIUM" textColor="#FFFF00" rendered="true"/>

            <Shape class="idea.graphics.shapes.simple.Image" fileName="./img/javaxml-duke.png"
rendered="true">
              <S42GeoPosition id="position" x="3280000" y="5460000"/>
             </Shape>

        </GeographicMap>
    </Renderer>

    <MapWriters>
        <MapWriter class="idea.map.writer.MapWriterToFile" fileName="./output/altitudes.png"/>
        <MapWriter class="idea.map.writer.ImageMapWriterToCanvas"/>
    </MapWriters>
</MapGenerator>
```

5.1.2. Runset analyse

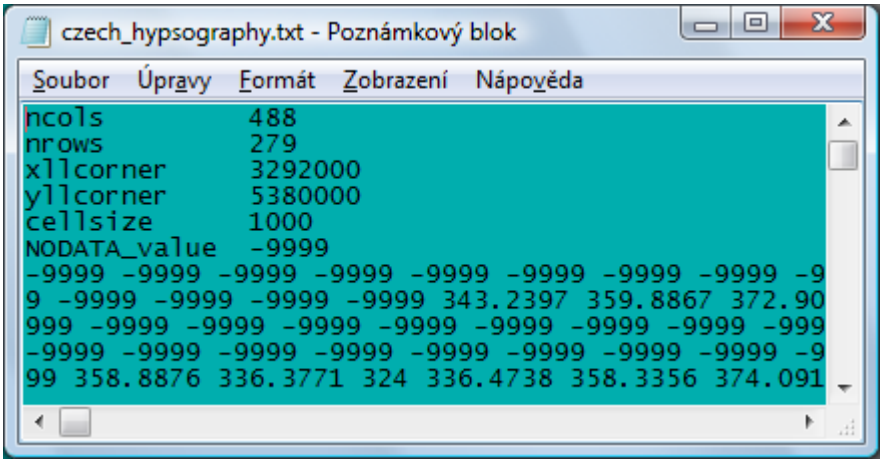
XML element **MapGenerator** is root element, that contains all child objects.

XML element Description contain Runset description as plain text.

XML element **Matrixes** contains matrixes parameters – rows/columns numbers, cell size 1000 m and **CartesianGeoPosition** of left bottom cell. Next, there is input matrix **InputMatrixFromFile** with attribute **id="hypsography"** and **fileName** that load terrain model from specified file.

There is not any computed matrix.

Input matrix **czech_hypsography.txt** must have same position of left bottom corner, number of cells and cell size as defined in atributes of element **Matrixes**.



XML element **Renderer** contains attribute **matrix**="**hypsography**", so it will render loaded terrain model into map. Next, there is element **ColorScale** with color scale definition.

Another child element of **Renderer** is **GeographicMap** with atribute **scale="1000"**. One pixel on map represents 1 km^2. Position of left bottom corner on map is shifted against matrix parameters more left bottom and map size is larger than numbers of matrix cells – this cause white frame around map. One matrix cell is corresponds to one pixel on map.

Next you see list if XML elements **Shape**:

- **ScaleLegend** draw legend on defined position
- **RegionsCzechRepublic** draw regions borers
- **PolygonsFromGeoDataFile** with attribute **fileName="./geodata/cz_lakes.txt"** draw lakes.
- **PolylinesFromGeoDataFile** with attribute **fileName="./geodata/cz_rivers.txt"** draw rivers.

- **Text** draw given string to defined position
- **CitiesCzechRepublic** with attribute **sizeLimit="MEDIUM"** draw main cities in Czech Republic.
- **Image** draw image from file on defined position

XML element **MapWriters** contains two map writers – **MapWriterToFile** save image to file (./output/Altitudes.png), **ImageMapWriterToCanvas** is responsible for display window.

5.2. Generate map from values in XML file

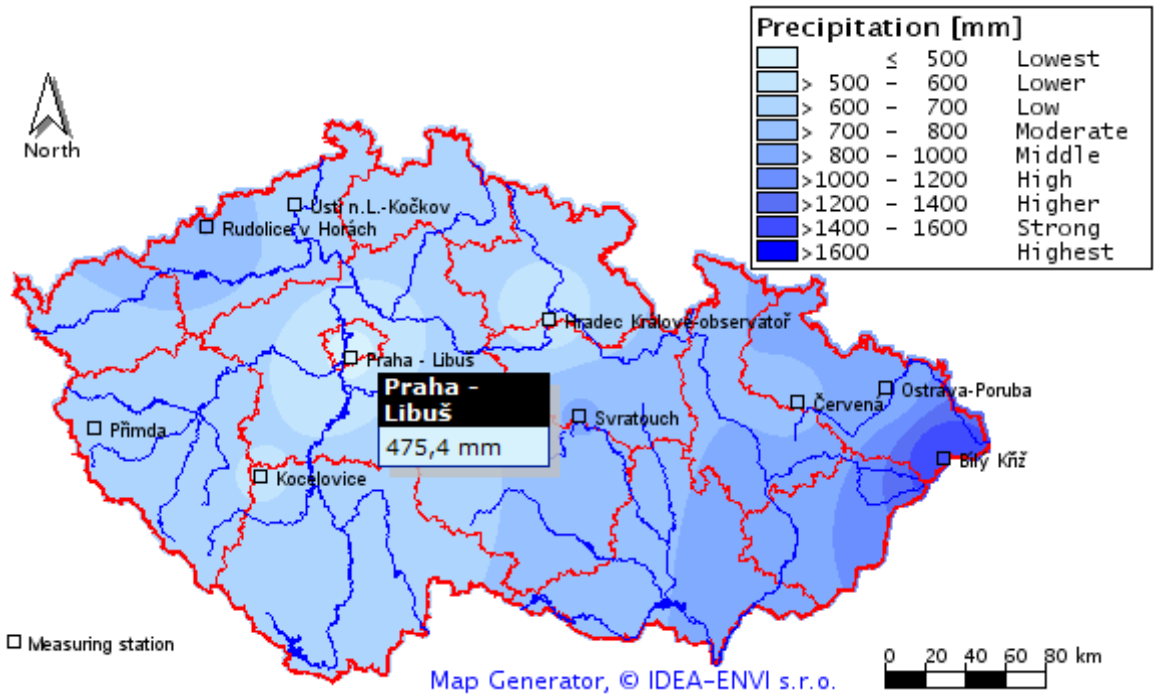
In subdirectory **runsets** there is files **Rain2005_a.xml, Rain2005_b.xml** and in subdirectory **matrixes** there is **czech_hypsography.txt -** input matrix with fine smoothed terrain model of Czech Republic. In subdirectory **inputdata** there is file **rain2005.xml** where are input values.

5.2.1. Input file rain2005.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<DataValues>
	<row name="Kocelovice" value="594.4">
		<S42GeoPosition x="3415977" y="5482301"/>
	</row>
	<row name="Hradec Králové-observatoř" value="542.3">
		<S42GeoPosition x="3559964" y="5560964"/>
	</row>
	<row name="Svratouch" value="811.7">
		<S42GeoPosition x="3574692" y="5512015"/>
	</row>
	<row name="Přimda" value="664.3">
		<S42GeoPosition x="3332572" y="5506761"/>
	</row>
	<row name="Bílý Kříž" value="1581.7">
		<S42GeoPosition x="3756433" y="5491607"/>
	</row>
	<row name="Červená" value="762.2">
		<S42GeoPosition x="3683161" y="5519238"/>
	</row>
	<row name="Ostrava-Poruba / ČHMÚ" value="845.8">
		<S42GeoPosition x="3727479" y="5526401"/>
	</row>
	<row name="Praha - Libuš" value="475.4">
		<S42GeoPosition x="3460493" y="5541924"/>
	</row>
	<row name="Košetice" value="680.8">
		<S42GeoPosition x="3505806" y="5493429"/>
	</row>
	<row name="Rudolice v Horách" value="804">
		<S42GeoPosition x="3388213" y="5606867"/>
	</row>
	<row name="Ústí n.L.-Kočkov" value="655.5">
		<S42GeoPosition x="3432330" y="5617518"/>
	</row>
</DataValues>
```

Launch Runset from command line ...

```
java -XX:+UseParallelGC -XX:+UseAdaptiveSizePolicy -jar MapGenerator.jar /F
3 log.txt .\runsets\Rain2005_a.xml
```



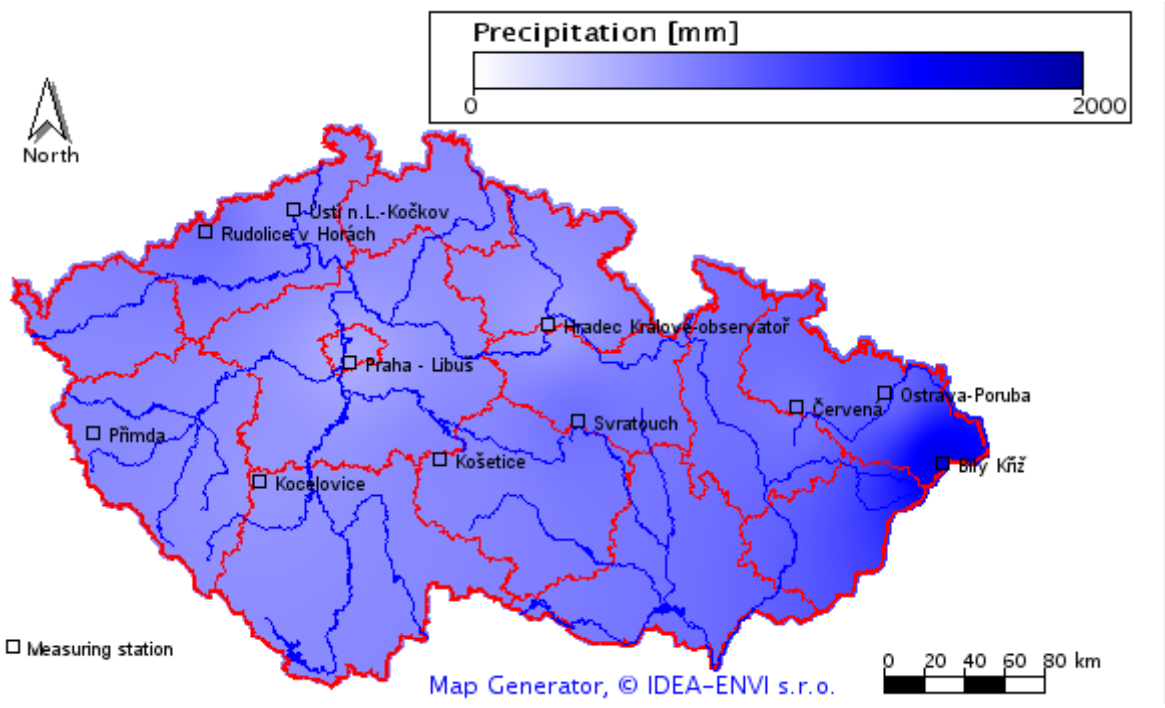
… and open file **./output/rain2005_a.html** using HTML browser:

Tooltips shows when user place mouse over station. When not works, file **wz_tooltip.js**, part of library **JavaScript, DHTML Tooltips** is missing in output directory.

Launch another Runset from command line ...

```
java -XX:+UseParallelGC -XX:+UseAdaptiveSizePolicy -jar MapGenerator.jar /F
3 log.txt .\runsets\Rain2005_b.xml
```

… and open file **./output/rain2005_b.html** using HTML browser:



5.2.2. Runsets Rain2005_a.xml and Rain2005_b.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<MapGenerator>
    <Description>Display precipitation map, year 2005</Description>
        <Matrixes rows="279" columns="488" cellSize="1000">
        <CartesianGeoPosition id="leftBottomCorner" x='3292000' y='5380000'/>

        <InputMatrix class="idea.map.matrix.InputMatrixFromFile" id="altitudes"
fileName="./geodata/vysko_interp.txt"/>

        <ComputedMatrix id="rain">
            <DataValues class="idea.map.data.DataValuesFromXmlFile"
fileName="./inputdata/rain2005.xml"/>
            <Interpolator class="idea.map.interpolator.IDWInterpolator" stations="12" beta="2"
threads="2" timeout="60" excludeAreaMatrix="altitudes"/>
        </ComputedMatrix>

    </Matrixes>
    <Renderer class="idea.map.renderer.GeoImageMapRenderer" matrix="rain">
        <ColorScale class="idea.map.colorscale.StandardListColorScale" text="Precipitation"
unit="mm">
          <ColorForValueRange name="Lowest" color="#D7F2FD" valueFrom="0" valueTo="500" seq="1" />
          <ColorForValueRange name="Lower" color="#C2E4FD" valueFrom="500" valueTo="600" seq="2"/>
          <ColorForValueRange name="Low" color="#AED5FE" valueFrom="600" valueTo="700" seq="3"/>
          <ColorForValueRange name="Low-Mid" color="#98C1FF" valueFrom="700" valueTo="800" seq="4"/>
          <ColorForValueRange name="Mid" color="#81ABFF" valueFrom="800" valueTo="1000" seq="5"/>
          <ColorForValueRange name="High" color="#6B8FFF" valueFrom="1000" valueTo="1200" seq="6"/>
          <ColorForValueRange name="Higher" color="#5970F4" valueFrom="1200" valueTo="1400"
seq="7"/>
          <ColorForValueRange name="Strong" color="#404DFF" valueFrom="1400" valueTo="1600"
seq="8"/>
          <ColorForValueRange name="Highest" color="#0000FF" valueFrom="1600" seq="9"/>
        </ColorScale>
        <GeographicMap scale="1000">
              <CartesianGeoPosition id="leftBottomCorner" x='3267000' y='5370000'/>
              <CartesianDimension id="dimension" width="600" height="350"/>
              <HtmlImageMap>
                    <HtmlImageMapItemsFromInputDataValues
class="idea.graphics.map.HtmlImageMapItemsFromInputDataValues" matrix="rain" type="square"
size="6"/>
              </HtmlImageMap>

              <Shape class="idea.graphics.shapes.complex.ScaleLegend" font="Lucida Console-PLAIN-
12" frame="false">
                    <S42GeoPosition id="position" x="3666000" y="5715000"/>
                    <S42Dimension id="dimension" width="120000" height="80000"/>
              </Shape>

              <Shape class="idea.graphics.shapes.complex.geo.czech.RegionsCzechRepublic"
drawColor="#FF0000" rendered="true"/>

              <Shape class="idea.graphics.shapes.complex.PolylinesFromGeoDataFile"
fileName="./geodata/toky.txt" drawColor="#0000FF" rendered="true"/>

              <Shape class="idea.graphics.shapes.simple.Text" drawColor="#0000FF" text="Map
Generator, © IDEA-ENVI s.r.o." fontName="Lucida Sans Unicode-PLAIN-12">
                    <S42GeoPosition id="position" x='3520000' y='5376000'/>
              </Shape>

              <Shape class="idea.graphics.shapes.complex.Stations" rendered="true" matrix="rain"
type="circle"/>
        </GeographicMap>
        </Renderer>
        <MapWriters>
              <MapWriter class="idea.map.writer.MapWriterToFile"
fileName="./output/rain2005_a.png"/>
              <MapWriter class="idea.map.writer.MapWriterToHtmlFile"
fileName="./output/rain2005_a.html" imageFileName="rain2005_a.png"/>
              <MapWriter class="idea.map.writer.ImageMapWriterToCanvas"/>
```

```
        </MapWriters>
</MapGenerator>
```

5.2.2.1 Runsets analyse

We explain only new things, against previous sample. Both runsets are similar, first use **list color scale**, second one use **linear color scale**.

Matrixes list contains as in previous sample **input matrix** with terrain model. But it serves only for following: its **id** is defined in Interpolator in **excludeAreaMatrix="hypsography".** It means that where this matrix have NODATA values (outside of Czech Republic), interpolation skip those cells – so area outside of Czech Republic remains white.

Next there is computed matrix **ComputedMatrix** with attribute **id="rain"**. It have as input data object **DataValuesFromXmlFile** with parameter **fileName** where is our file with measured data.

Computed matrix have Interpolation method **IDWInterpolator,** that interpolate input values into cells of matrix. Interpolator attributes:

- **stations="12"** – number of surrounding input values, that come into interpolation of cell
- **beta="2"** – weight factor
- **threads="2"** – number of threads for interpolation
- **timeout="60"** – number of second, after exceeding itinterpollation will be terminated, if not finish sooner
- **excludeAreaMatrix="hypsography"** – matrix id for cut off

Renderer draw matrix with id **rain**

Object **HtmlImageMapItemsFromInputDataValues** create clickable map from input values of matrix **rain**.

In list of graphics objects there is Shape **Stations** with atribute **matrix**="rain". It is responsible for drawing measuring stations into map from input values of matrix **rain**.

MapWriterToFile and **MapWriterToHtmlFile** writes to disk PNG with map and enclosing HTML file.

5.2.2.2 Output log analyse

While Map Generator working, it write log entries to screen and log file. Their quantity depends from parameter logLevel.

```
5.10.2007 8:34:43 idea.map.interpolator.IDWInterpolator interpolate
FINE: IDW Interpolation of matrix rain finished in 0.249 sec.
```

Information about time needed for interpolation.

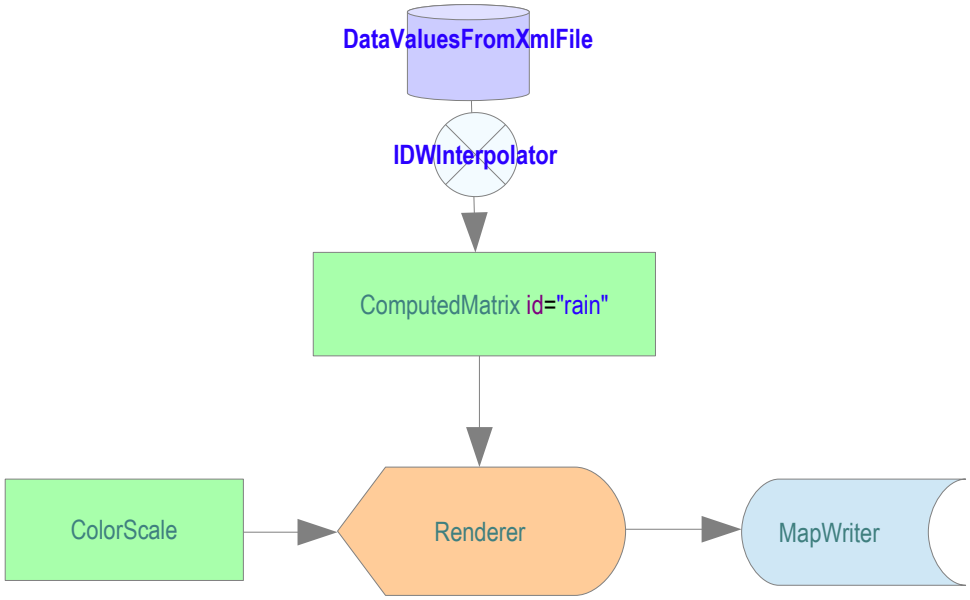
```
5.10.2007 8:34:43 idea.map.matrix.Matrixes modify
FINE: No modificator is set
```

Inform that modificator list is void.

If we change parameter logLevel to 2, then see more detailed list. We see, that interpolation was divided into two threads and also see their times. Summary time is time of slower thread.

```
FINER: IDW Interpolating Thread IDWInterpolatingThread of matrix rain finished in 0.203 sec.
5.10.2007 9:11:42 idea.map.interpolator.IDWInterpolatingThread run
FINER: IDW Interpolating Thread IDWInterpolatingThread of matrix rain finished in 0.234 sec.
5.10.2007 9:11:42 idea.map.interpolator.IDWInterpolator interpolate
FINE: IDW Interpolation of matrix rain finished in 0.234 sec.
```

5.2.2.3 Graphic representation of Runset processing



5.3. Runset TestGraphicElements.xml

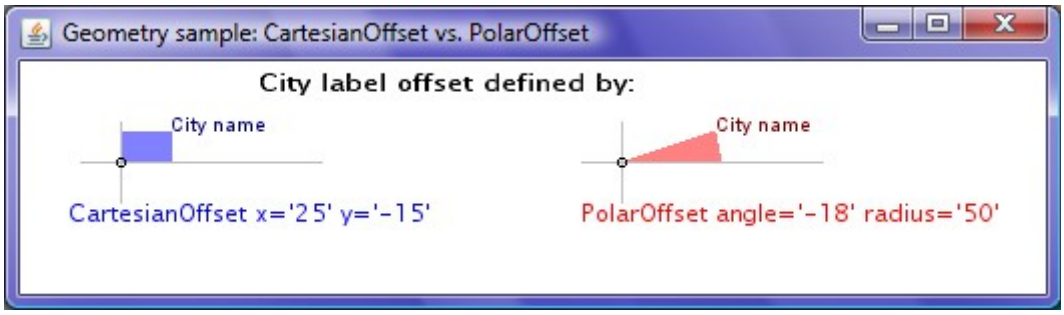
This Runset demonstrates using most of shapes and filters. Shapes are commented out using:

```
rendered="false"
```

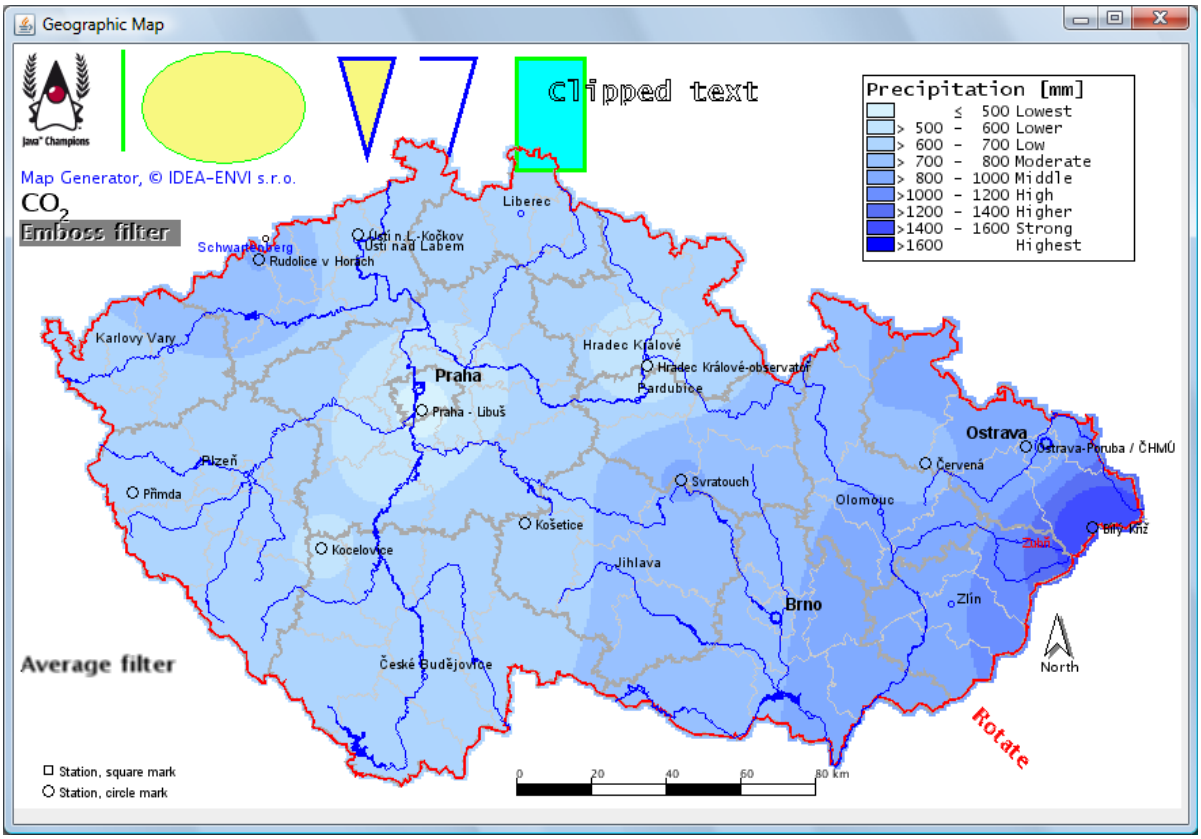
You can change it to **true** in particular XML element and see result.

5.4. Runset Geometry.xml

This runset shows difference between Cartesian and Polar Offsets:



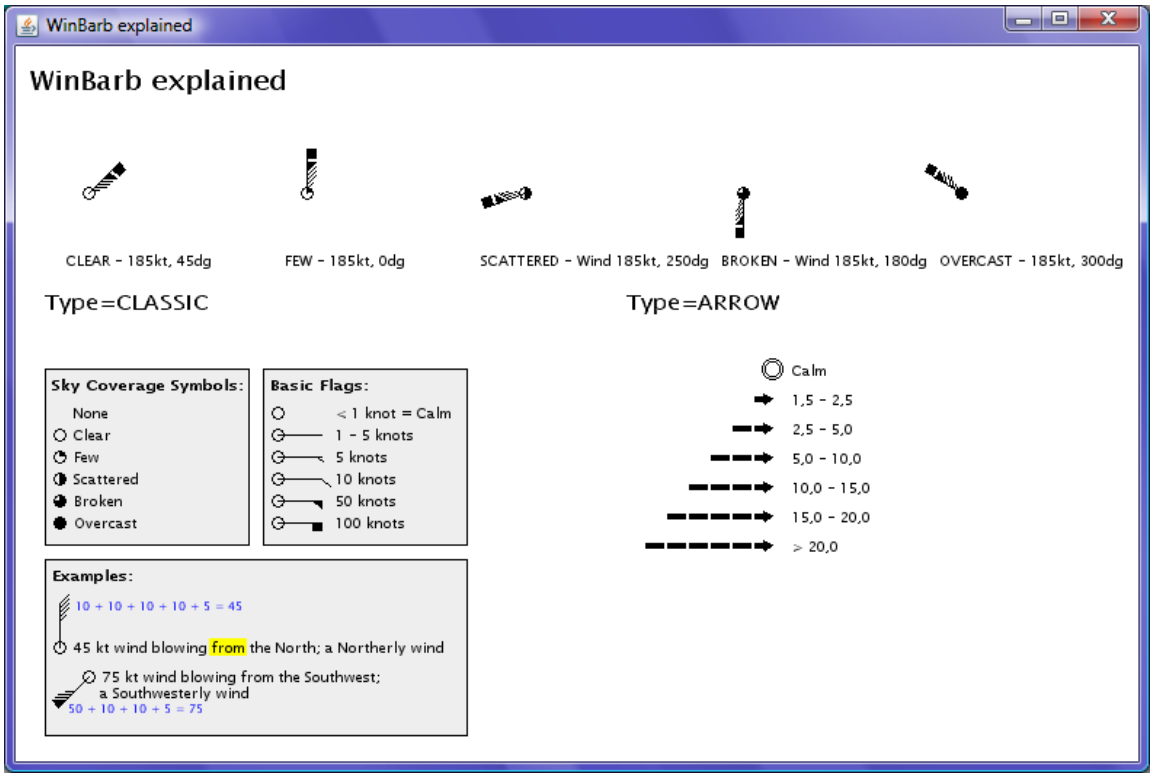
5.5. Runset AnotherTestGraphicElements.xml



This Runset demonstrates using most of shapes and filters. Also, it creates larger map image.

5.6. Runset WinBarbExplained.xml

This Runset demonstrates using WindBarb shape.


WinBarb explained
WinBarb explained
CLEAR - 185kt, 45dg
FEW - 185kt, 0dg
SCATTERED - Wind 185kt, 250dg
BROKEN - Wind 185kt, 180dg
OVERCAST - 185kt, 300dg
Type=CLASSIC
Type=ARROW
Sky Coverage Symbols:
None
Clear
Few
Scattered
Broken
Overcast
Basic Flags:
< 1 knot = Calm
1 - 5 knots
5 knots
10 knots
50 knots
100 knots
Calm
1,5 - 2,5
2,5 - 5,0
5,0 - 10,0
10,0 - 15,0
15,0 - 20,0
> 20,0
Examples:
10 + 10 + 10 + 10 + 5 = 45
45 kt wind blowing from the North; a Northerly wind
75 kt wind blowing from the Southwest;
a Southwesterly wind
50 + 10 + 10 + 5 = 75