

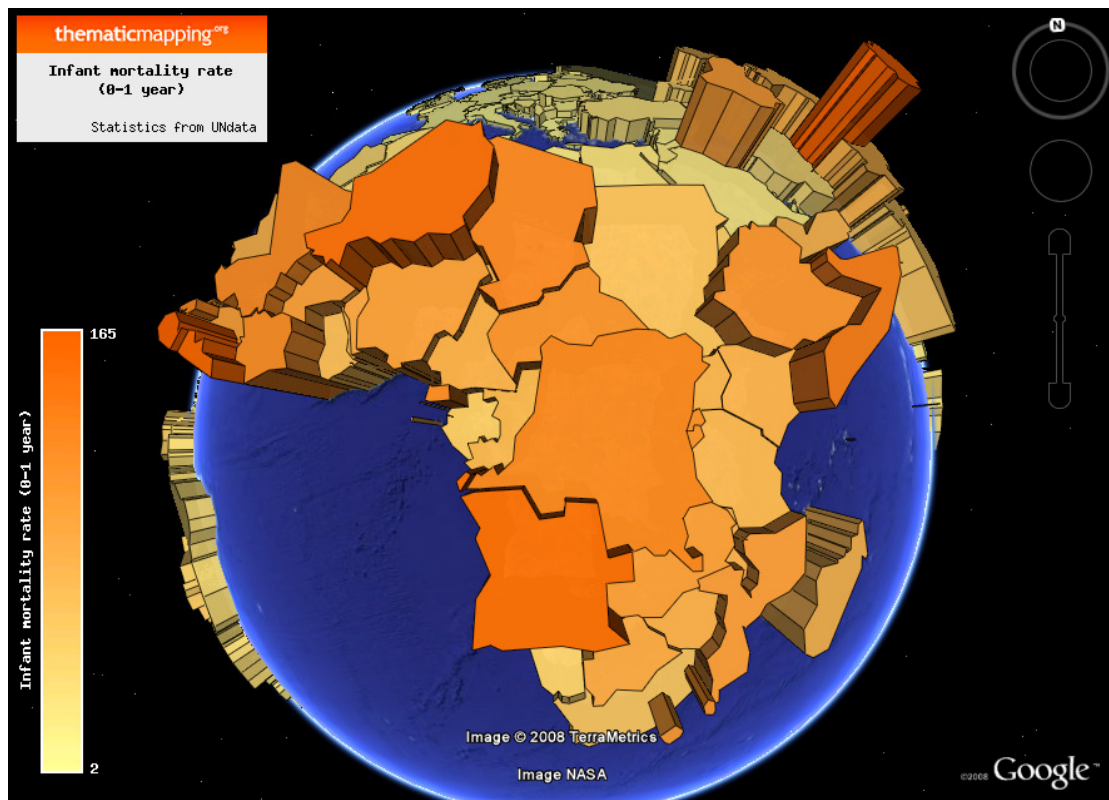
Institute of Geography - School of GeoSciences - University of Edinburgh

MSc in Geographical Information Science 2008

Awarded with Distinction

Part 2: Supporting Document

Thematic Mapping Engine



Bjørn Sandvik

This document is available from thematicmapping.org under a *Creative Commons Attribution-Share Alike 3.0 License*: <http://creativecommons.org/licenses/by-sa/3.0/>

Table of contents

1. Introduction	5
2. The Thematic Mapping Engine	7
2.1 Requirements	7
2.3 The TME web Interface	8
2.3.1 User guide	9
2.3.2 How the web interface works	10
2.4 TME Application Programming Interface (API)	13
2.4.1 TME DataConnector class	14
2.4.2 TME ThematicMap class	15
3. Data preparation	17
3.1 Using open data.....	17
3.2 UN statistics	17
3.3 World borders dataset	18
3.4 International country codes.....	20
4. Database	21
4.1 Uploading spatial data.....	21
4.2 Uploading statistical data.....	22
4.3 Querying and transforming spatial data.....	23
5. Thematic mapping techniques for KML	26
5.1 The KML styling mechanism	26
5.2 Mathematical scaling of point symbols	26
5.3 Proportional symbols	28
5.3.1 Proportional image icons	28
5.3.2 Drawing regular polygons.....	29
5.3.3 Proportional 3-D Collada objects.....	31
5.4 Chart maps	34
5.5 Bar maps	35
5.6 Choropleth maps	36
5.7 Prism maps.....	37
5.8 Temporal maps.....	38
5.9 Map tiles.....	39
5.9.1 GDAL2Tiles	39
5.9.2 KML super-overlay.....	40
5.10 Map colours	41
5.11 Map legend.....	42
5.11.1 Creating colour legends with GD	43
5.12 File compression	44
6. The thematic mapping website	45
7. References	48

List of figures

Figure 1: The interfaces of the Thematic Mapping Engine.	7
Figure 2: The web interface of the Thematic Mapping Engine	8
Figure 3: Prism map shown with the Google Earth Plug-in.	10
Figure 4: The plug-in is currently not supporting the KML time primitives.....	10
Figure 5: AJAX based communication between web browser and web server.....	11
Figure 6: TME web server infrastructure.....	12
Figure 7: TME Application Programming Interface (API)	13
Figure 8: Choropleth map in Google Earth.....	14
Figure 9: Choropleth map in Google Maps.	14
Figure 11: The world borders Shapefile	18
Figure 12: World borders Shapefile: The resolution of the original dataset.....	19
Figure 13: World borders Shapefile: After removing island polygons.....	19
Figure 14: World borders Shapefile: After simplifying borders.....	19
Figure 15: Entity-Relationship (ER) diagram showing the database structure.....	21
Figure 16: KML winding order.....	24
Figure 17: Comparson of 2-D and 3-D symbols.....	27
Figure 18: GDP per capita with 1-D bars, 2-D circles and 3-D spheres.....	27
Figure 19: The KML icon images used to make proportional symbols	28
Figure 20: Proportional symbol map in Google Earth.....	29
Figure 21: Proportional symbol map in Microsoft Virtual Earth.	29
Figure 22: Scaling error in Google Earth.....	29
Figure 23: Various regular polygons	30
Figure 24: Regular polygons in Google Maps.....	31
Figure 25: Regular polygons in Google Earth.	31
Figure 26: How an object should be positioned in Google SketchUp.....	32
Figure 27: The <i>Tape Measure Tool</i> in Google SketchUp.....	32
Figure 28: The <i>Paint Bucket</i> tool in Google SketchUp	32
Figure 29: Collada objects available in the Thematic Mapping Engine	33
Figure 30: Population in Southeast Asia visualised with 3-D domes.....	33
Figure 31: Population in Southeast Asia visualised with 3-D humans.....	33
Figure 32: Pie chart showing age distribution.	34
Figure 33: World population visualised with scaled pie charts in Goole Earth.....	34
Figure 34: World population visualised with scaled pie charts in Goole Maps	34
Figure 35: Bar map in Google Earth.....	35
Figure 36: Bar map in Google Earth (variable diameter)	35
Figure 37: The polygon hole problem	37
Figure 38: The Google Earth time animation.	39
Figure 40: Map tiles in Google Earth	40
Figure 41: The RGB colour cube.....	41
Figure 42: Choropleth map showing life expectancy by using equal intervals	43
Figure 43: Choropleth map showing life expectancy by using qunatiles	43
Figure 44: Colour legend generated by the GD library	43
Figure 46: Number of visitors to thematicmapping.org website	47

List of tables

Table 1: Elements of TME web interface	9
Table 2: UNdata indicators	18
Table 3: KML style elements.....	26
Table 4: Number of map tiles for increasing zoom levels	39
Table 5: Number of visits to thematicmapping.org	47

1. Introduction

The aim of this research is to determine whether *Keyhole Markup Language* (KML) can be used for thematic mapping. As a proof-of-concept, the work culminates in the implementation of a fully functioning open source application, the *Thematic Mapping Engine*¹.

The term *neogeography* is commonly applied to the set of technologies and techniques presented in this document. Neogeography combines the complex techniques of cartography and GIS and places them within the reach of users and developers (Turner, 2006; Walsh, 2008; Davis, 2007).

“Every now and again a web based service comes along that takes our breath away, Bjørn Sandvik’s Thematic Mapping Engine is one of those services. (...) His Thematic Mapping Engine enables you to visualise global statistics on Google Earth in a way that only a few years ago would have been a showstopper using high end tools such as ESRI’s ArcGlobe.”

Dr Andrew Hudson-Smith, Digital Urban 1 July 2008²

The Thematic Mapping Engine was developed using open source software, and it is released as an open source project. The goal was to develop a low-cost solution suitable for non-profits and public benefit organisations. The application also demonstrates what it is possible to achieve using open source tools, open standards and datasets in the public domain.

Cartography and GIS have both emerged as major capabilities on the web. *Distributed maps* are different from traditional static maps in that they link information from various sources and provide a user-defined environment (Crampton, 2001).

“Mapping should proceed thorough multiple, competing visualisations which are not created by a cartographer and transmitted to the user but made on the spot by the user acting as his or her own cartographer.”

Jeremy W. Crampton (2001:236)

Such mapping environments are now easier to create with recent advances in web technologies and standards. The user can, to a large extent, determine what information is to be displayed and in what context. This project aimed to provide data, visualisation tools and a cartographic toolbox to the user in a web-based interface.

A step-wise approach was chosen in order to meet this aim:

1. Data preparation

The first step involved finding appropriate statistical and spatial data that could be combined and used for thematic mapping. The data had to be gathered from various sources and stored in a database.

¹ <http://thematicmapping.org/engine>

² <http://digitalurban.blogspot.com/2008/07/google-earth-thematic-mapping-engine.html>

2. **Thematic mapping with KML**

A scripting language (PHP) was used to query the database and transform the result into KML documents representing various thematic mapping techniques.

3. **Thematic Mapping Engine**

The Thematic Mapping Engine was developed to demonstrate how these techniques could be utilised in a web application.

4. **Evaluation**

The various techniques were evaluated after gaining feedback from people using the Thematic Mapping Engine.

The methods of thematic mapping are well described in the cartographic literature, but it was hard to find books and journal articles describing the use of KML and geobrowsers for this purpose. There is a lot of development going on, but little has so far been described in the academic literature. The focus was therefore shifted towards the “blogosphere”³. Blogs are now widely used among “geeks” and professionals alike, to present their own work and perspectives and to comment on other people’s work. Bloggers actively review and comment on the latest trends and developments, and this turned out to be a valuable information source for this project.

A dedicated website⁴ was established to present the various thematic mapping techniques to a wider audience. This website became popular in the blogosphere, and it was also featured by the United Nations and Google. A lot of valuable feedback was received as a result of this publicity.

This document is divided into five sections. The Thematic Mapping Engine is described in the first section. The second and third section describes how spatial and statistical data were collected, optimised, stored and retrieved. The forth section describes in detail the thematic mapping techniques introduced in the Research Paper (Sandvik, 2008), using code samples from the Thematic Mapping Engine. The last section shows how the thematicmapping.org website was used to exchange ideas with a wider community.

³ Blogosphere is a collective term encompassing blogs and their interconnections.

⁴ <http://thematicmapping.org>

2. The Thematic Mapping Engine

The Research Paper explains how *Keyhole Markup Language* (KML) and geobrowsers can be used for thematic mapping (Sandvik, 2008). The experiments show that KML has a great potential for thematic mapping, even though the techniques are complicated to use for non-programmers. By embedding the techniques in a *Thematic Mapping Engine*, it was possible to hide this complexity. TME allows the user to create thematic maps through an easy to use web interface, or by writing a few lines of code.

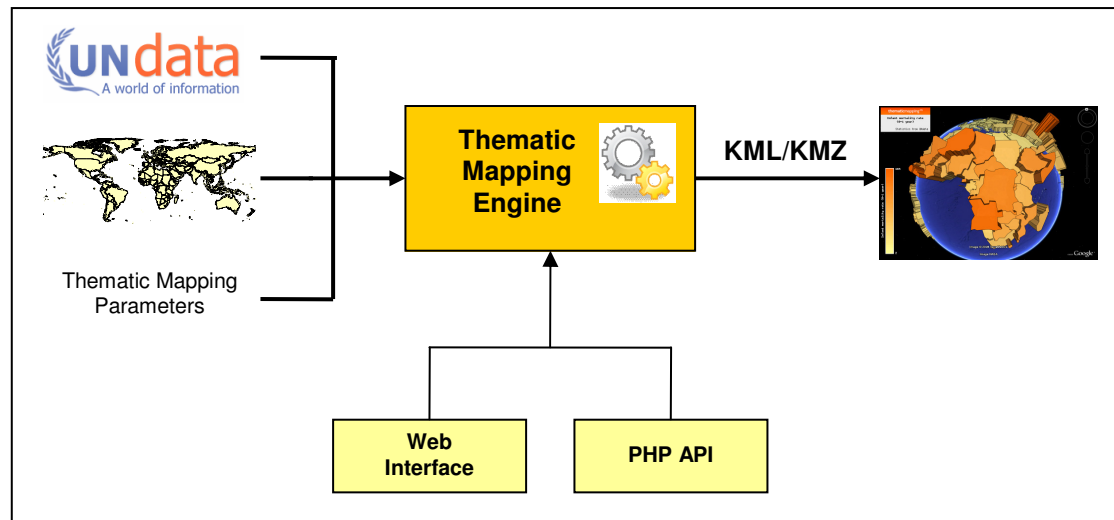


Figure 1: The interfaces of the Thematic Mapping Engine.

Figure 1 shows how the Thematic Mapping Engine works on a high level. The engine takes statistical data (attributes), spatial features and thematic mapping parameters as input and returns a KML/KMZ file. This file can be viewed in Google Earth, or other geobrowsers supporting the KML standard. TME can be accessed from a web interface (section 2.3) or from a PHP script (section 2.4).

2.1 Requirements

The Thematic Mapping Engine requires the following software (all are open source and available free of charge):

- **PHP (Version 5)**
PHP is a computer scripting language originally designed for producing dynamic web pages⁵.
- **MySQL (Version 5)**
MySQL is a relational database management system (RDBMS), especially popular for web applications⁶.

⁵ <http://www.php.net>

⁶ <http://www.mysql.com>

- **Apache HTTP Server**

The Apache HTTP Server is an open source web server, developed and maintained by the Apache Software Foundation⁷.

Both Linux and the Windows operating systems can be used. This software combination is often referred to as the LAMP⁸ or WAMP⁹ stack. This software combination is popular because of its low acquisition costs and because of the ubiquity of its components. It is the standard software package installed on many web servers (Kay, 2006).

The easiest option is to use a hosting provider whereby this software is pre-installed on the web server. The Thematic Mapping Engine was successfully installed using a web hosting solution from Bluehost¹⁰. The software stack can also be installed on a PC by using the XAMPP¹¹ distribution. XAMPP is an easy to install Apache distribution containing MySQL, PHP and Perl. Only the XAMPP Windows distribution has been tested.

The TME web interface requires the Ext JS JavaScript library. To be able to preview thematic maps in the web browser, the Google Earth Plug-in is required. These components can be installed from the following web sites:

- Ext JS (Version 2.1): <http://extjs.com/products/extjs>
- Google Earth Plug-in: <http://code.google.com/apis/earth>

2.3 The TME web Interface

With the TME web interface, thematic maps can be created in a web browser, without a single line of code. This is achieved through an interactive web form where the user can select between statistical indicators and various thematic mapping techniques. Mapping parameters, like the colour and size, can be readily changed. The form returns a KMZ file which can be visualised directly in the web browser using the new Google Earth plug-in, or downloaded to a computer.

The web interface supports all major web browsers, including Internet Explorer 6+, Firefox 1.5+, Safari 3+ and Opera 9+. The Google Earth plug-in currently only supports web browsers on the Windows operating system.

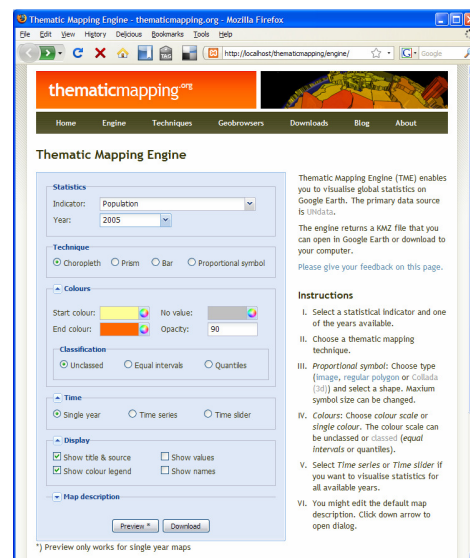


Figure 2: The web interface of the Thematic Mapping Engine

⁷ <http://www.apache.org>

⁸ LAMP is an acronym for Linux, Apache, MySQL and PHP.

⁹ WAMP is an acronym for Windows, Apache, MySQL and PHP.

¹⁰ <http://www.bluehost.com>

¹¹ <http://www.apachefriends.org/en/xampp.html>

2.3.1 User guide

This guide explains how to use the web interface to create a thematic map.

Table 1: Elements of TME web interface

	<p>Select the statistical indicator and one of the years available.</p>
	<p>Select one of the thematic mapping techniques (Choropleth, Prism, Bar, Proportional symbol)</p>
	<p>Prism map: Maximum height (in metres) of the prisms can be changed.</p>
	<p>Bar map: Maximum height (in metres) and bar radius can be changed.</p>
	<p>Proportional symbol: Select symbol type (Image, Regular polygon or 3-D object) and symbol shape. Maximum size can be changed.</p>
	<p>Choose colour scale or single colour map.</p> <p>The colour scale can be unclassified or classed (<i>equal intervals</i> or <i>quantiles</i>). The number of classes can be changed (2-9 classes).</p>
	<p>Select <i>Time series</i> or <i>Time slider</i> to visualise statistics for all available years.</p>
	<p>Select information elements that should be displayed on the map.</p>
	<p>The default title, description and source of the map can be changed.</p>
	<p>Click the <i>Preview</i> button to view map in the web browser (requires Google Earth plug-in) or the <i>Download</i> button to download the KMZ file.</p>

2.3.2 How the web interface works

The TME web interface can be characterised as a *Rich Internet Application* (RIA), a web application that has the features and functionality of traditional desktop applications (Loosley, 2006). In a traditional web application, all processing is done on the web server and a new web page is downloaded each time the user clicks. RIAs transfer the processing necessary for the user interface to the web browser, but keep the bulk of the data back on the web server.

The web interface was developed using HTML, JavaScript, Ext JS, Google Earth API and AJAX¹² techniques. By combining these technologies, it was possible to create a responsive user experience.

Ext JS is a cross-browser JavaScript library for building desktop-like web applications. Ext is dual licensed under the General Public License (GPL), which TME uses, and a commercial license. By using Ext JS, it was possible to build interactive form elements which responded to various events initialised by the user or the program flow. The validity check of the user inputs is performed in the web browser and not on the server. Ext JS also controls the AJAX based communication between the web browser and the web server.

AJAX is a group of interrelated web development techniques used for creating interactive web applications. By using AJAX, it is possible to retrieve data from the web server asynchronously in the background without interfering with the display and behaviour of the existing web page. Despite its name, XML is not required as the data-interchange format. TME uses the JSON¹³ encoding to transfer data from the web server to the web browser, as it is more readily generated and parsed by programming languages.

In May 2008, Google launched the Google Earth Plug-in together with a free JavaScript API (Rademacher, 2008). This made it possible to embed Google Earth in a web interface, instead of having to switch between two applications (the web browser and Google Earth).

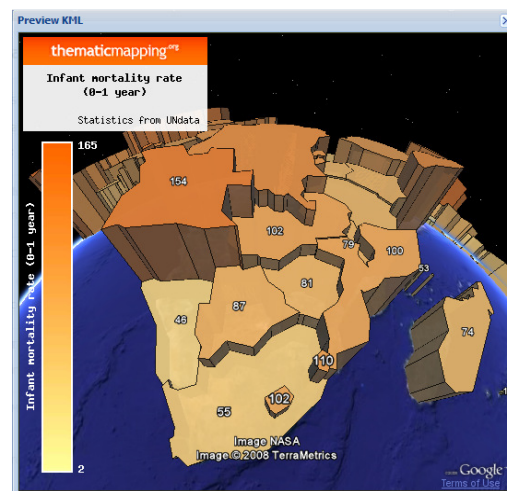


Figure 3: Prism map shown with the Google Earth Plug-in.

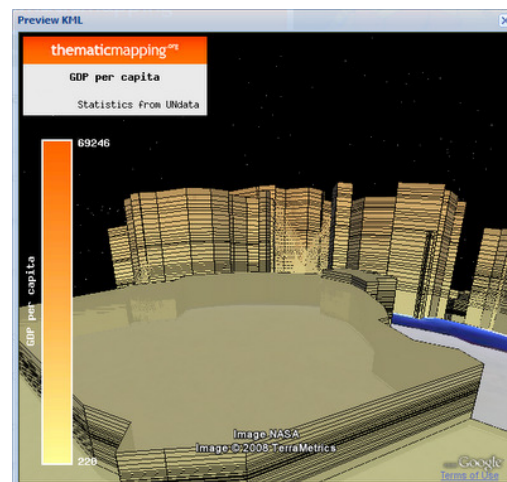


Figure 4: The plug-in is currently not supporting the KML time primitives. All prisms are rendered on top of each other, regardless of the time specified.

¹² AJAX is an acronym for Asynchronous JavaScript and XML.

¹³ JSON is an acronym for JavaScript Object Notation.

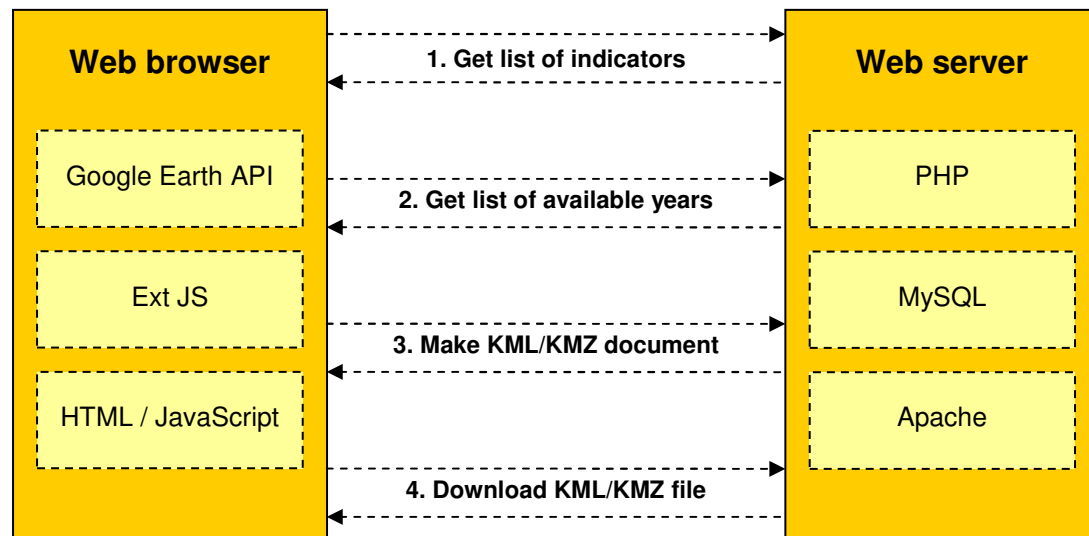


Figure 5: AJAX based communication between web browser and web server.

Figure 5 shows the communication occurring between the web browser and the web server when the user is creating a thematic map:

1. When the web interface is loaded by the web browser, an AJAX request is automatically fired to the web browser¹⁴. The browser is asking for a list of all available statistical indicators, and this list is returned by the web server. This information is added to the first drop-down box in the web form.
2. When the user selects one of the indicators in the list, a new request is sent, asking for a list of years where data is provided. The years are added to the second drop-down box.
3. The third request is sent when the user clicks the *Preview* or *Download* button. All of the map parameters, representing the choices made by the user, are then transferred to the web server. The parameters are used to generate a KML document, which are zipped into a KMZ file and stored on the server. A reference (URL) to the KMZ file is returned to the browser.
4. If the *Preview* button was clicked, the KMZ file is automatically loaded by the Google Earth Plug-in. If the *Download* button was clicked, a link to a KMZ file is displayed. The user can download the file by clicking on this link.

¹⁴ This happens to be the same web browser as the web interface was loaded from, but it could also be a different web server.

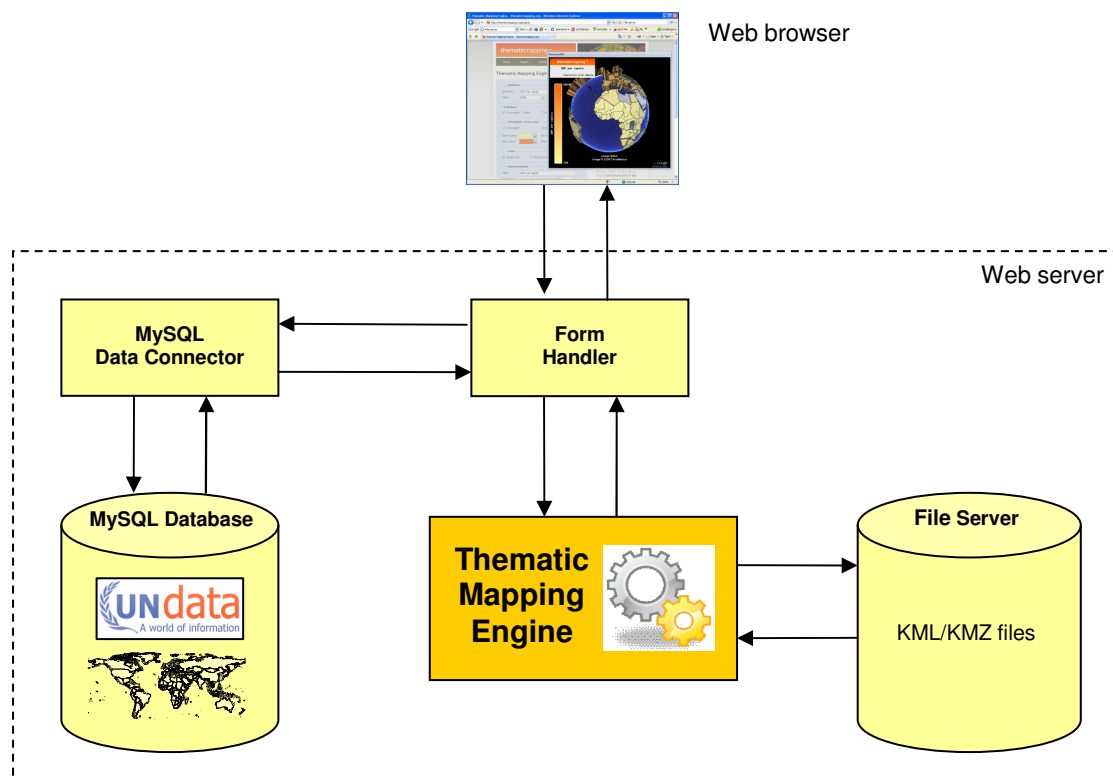


Figure 6: TME web server infrastructure

Figure 6 shows how the requests from the web browser are handled by the web server. All communication goes through a single PHP script. This *Form Handler* script makes use of two additional PHP classes:

- *MySQL Data Connector* class
This class retrieves data from the MySQL database (indicators, years or data used to make a thematic map). This class is further explained in section 2.4.1.
- *Thematic Mapping* class
The data from the MySQL Data Connector class are passed on to this class, together with the map parameters sent from the web browser. This information is used to generate the KML document. This class is further explained in section 2.4.2.

2.4 TME Application Programming Interface (API)

The Thematic Mapping Engine can also be used as an application programming interface (API). This allows thematic maps to be created with a few lines of PHP code. Existing or new applications can use this API to add thematic mapping functionality.

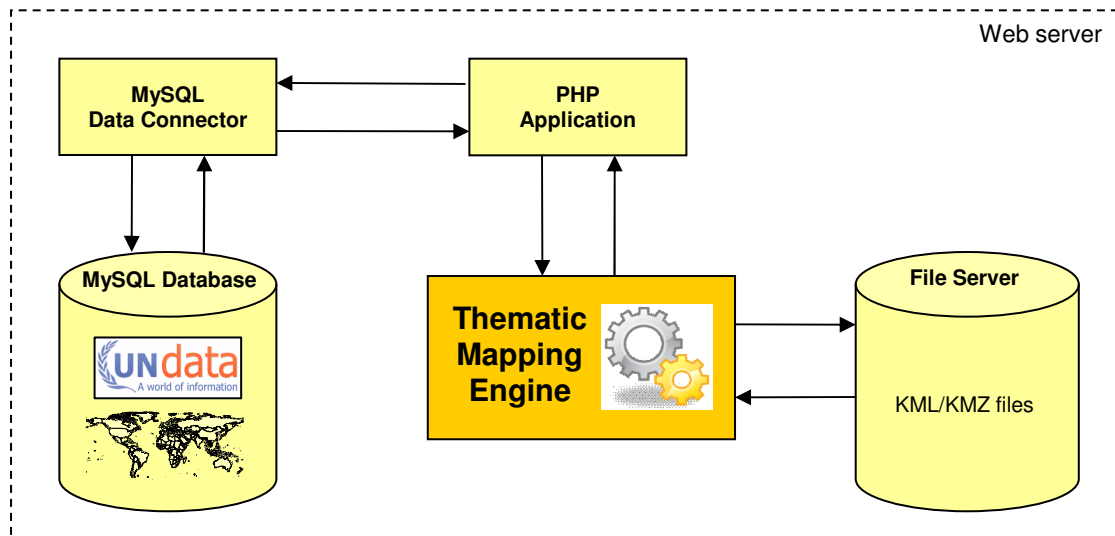


Figure 7: TME Application Programming Interface (API). The Form Handler script is here replaced by a custom PHP script.

The TME API works by including the two classes introduced above in a PHP application. This is the code required to create a choropleth map:

```
01 include ('TME_MySQL_DataConnector.php');
02 include ('TME_Engine.php');
03
04 $dataConnector = new DataConnector();
05 $dataStore = $dataConnector->getDataStore(68, 2005, 0);
06
07 $parameters = array( 'mapType'      => 'choropleth',
08                     'indicator'    => 68,
09                     'year'         => 2005,
10                     'classification' => 'equal' );
11
12 $map = new ThematicMap($dataStore, $parameters);
13 $file = $map->getKML();
```

The two classes are included in line 1 and 2. A new *dataConnector* object is created from the *DataConnector* class (line 4). The *getDataStore* method is called to retrieve the data required to create a thematic map (line 5). The parameters refer to the *indicator id*, *year* and a *region code*. The region code is used only to select the values and features from one continental region (e.g. Africa)¹⁵. All map parameters are added

¹⁵ Region codes are from United Nations Statistical Division:
<http://unstats.un.org/unsd/methods/m49/m49regin.htm>

to an array¹⁶ (line 7-10). A new *map* object is created from the *ThematicMap* class by adding the data array and the map parameters (line 12). Finally, a reference (URL) to the KMZ file is retrieved by calling the *getKML* method of the *map* object.

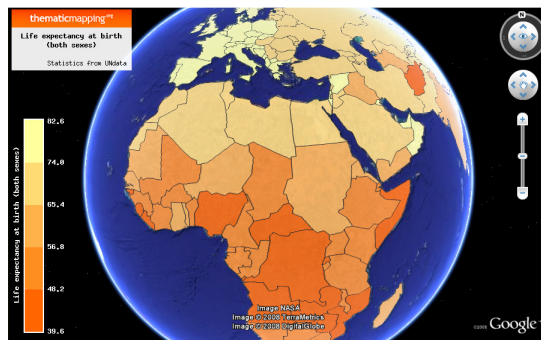


Figure 8: The KMZ file, returned from the PHP script above, displayed in Google Earth.

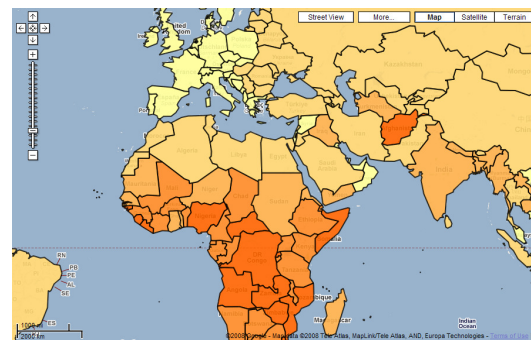


Figure 9: The same KMZ file displayed in Google Maps.

All valid map parameters are listed in appendix C1. An additional example for proportional symbol maps is included in appendix C2.

2.4.1 TME DataConnector class

The object-oriented programming (OOP) features introduced in PHP 5 were used to create an application that could be readily extended (Ullman, 2007). All queries to the MySQL database are kept in a separate class (TME_MySQL_DataConnector.php). A different database can be used by adding a new PHP data connector class¹⁷.

The DataConnector class contains three methods:

- *getIndicators ()*
This method returns a list of the statistical indicators in the database.
- *getIndicatorYears (\$indicatorID)*
This method returns a list of available years for one indicator.
- *getDataStore (\$indicatorID, \$year, \$region)*
This method retrieves the spatial features and statistical values required to create a thematic map. The data is returned in a multidimensional array:

¹⁶ A data structure that associates values with keys is also called a *hash table*.

¹⁷ Instead of creating a class interacting with a MySQL database, one could make one that interacts with a non-specific database. By using the OOP principles of inheritance and overriding, a more particular class for MySQL could be defined. This approach would enforce new data classes to implement the same functionality as the general class.

```

dataStore => Array
  [features] => Array
    [feature id] => Array
      [name] => name
      [lon] => longitude
      [lat] => latitude
      [wkt] => WKT geometry
      [...]
    [indicators]
      [indicator id]
        [name]
        [description]
        [source]
        [decimals]
        [max]
        [min]
        [values] => Array
          [2005] => Array
            [feature id] => value
            [...]
          [...]
        [years] => Array
        [...]

```

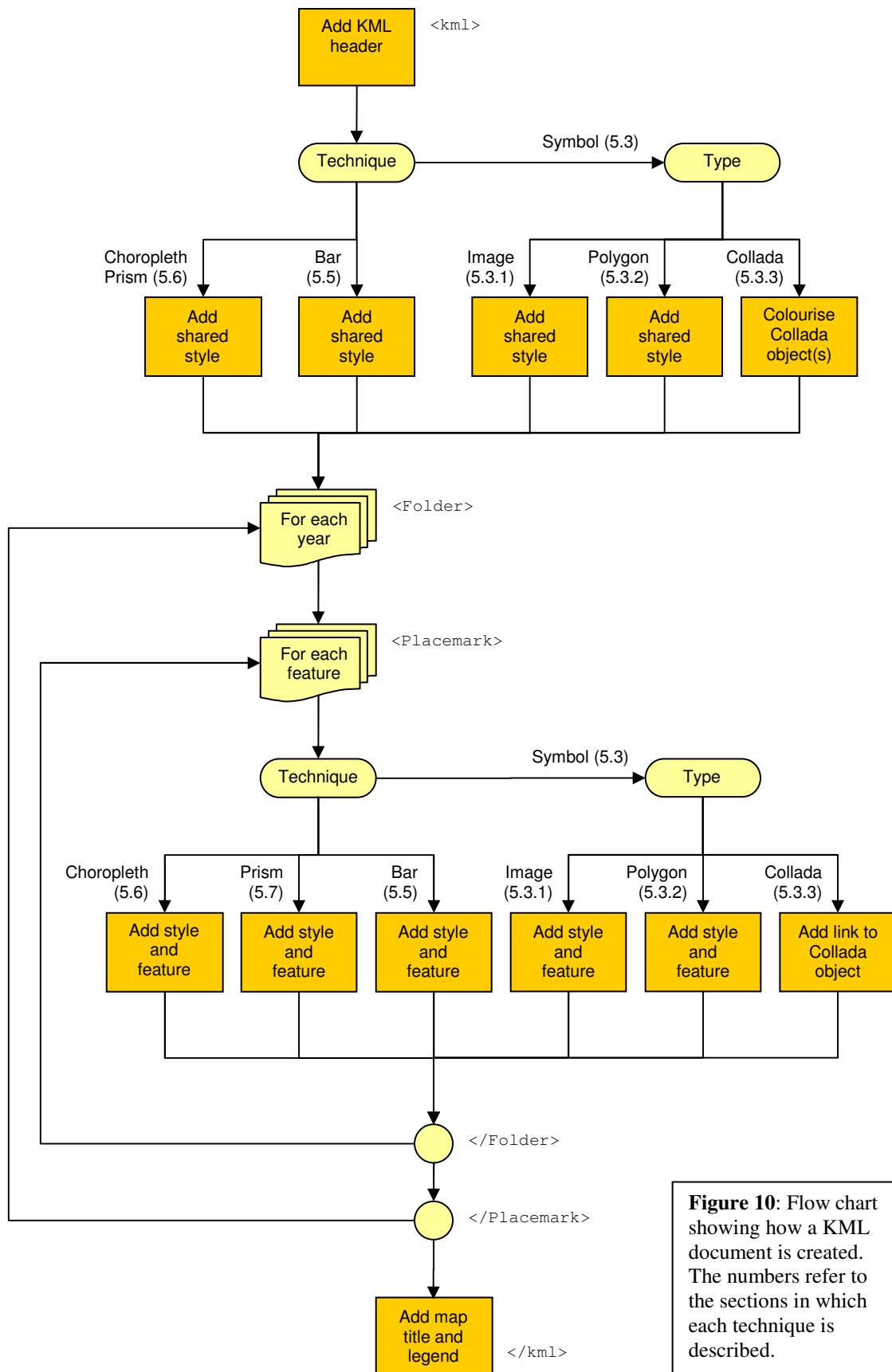
The spatial features (e.g. countries) and the statistical values are kept separately in the array, as not all features have values, and not all values have associated features. Name, longitude, latitude and geometry (i.e. border) is added for each feature. Various metadata are added, together with the statistical values.

2.4.2 TME ThematicMap class

The multidimensional array retrieved from the *DataConnector* class is passed on to the *ThematicMap* class together with the mapping parameters. Figure 10 shows a flow chart of how the KML document is created when the *getKML* method is called.

Firstly, the shared styles are defined. The shared styles will be different for each thematic mapping technique. An exception is choropleth and prism maps, which have the same shared styles. There are two *for-each* loops. The outermost loop runs through each of the years. A new KML *Folder* element is added for each year. The inner loop runs throughout all the features (e.g. countries) present in the data store. A KML *Placemark* element is added for each feature. Within this element, feature specific styles and the feature itself are added. How this is done for each thematic mapping technique is explained in section 5.

This is a simplified view, as there are many more conditions for handling all the map parameters.



3. Data preparation

Due to special interest and availability, international statistics were used for the thematic mapping experiments. Three types of data were gathered: a spatial dataset containing world borders, statistical data from the United Nations, and international country codes. The latter was used to link statistical values to the spatial features. Although this is principally global data, the techniques developed here are scale-independent.

3.1 Using open data

Since KML is an open, human-readable format, it was required to use data provided under open and non-restrictive licensing terms. The following requirements had to be met:

- *Access for all.* Anyone with an internet connection should be able to access the mapping examples. The *terms of use* of the Google mapping technology also states that “your service must be freely accessible to end users”¹⁸.
- *Using vector data.* Many data licenses do not readily permit use in an open interface as data providers are concerned about backward engineering. Using an open standard like KML would violate such licensing terms.
- *Allow redistribution.* Many data providers forbid redistribution of their data, which is basically what this project is doing by using KML and publicly available geobrowsers.

An important goal of this project was also to show the benefits of using public domain (“open”) data.

3.2 UN statistics

UNdata¹⁹ is a new internet-based data service which offers free access to a wide range of global statistics through a single entry point. The database service is offered by the United Nations Statistics Division (UNSD). UNdata enables users to access a large number of UN databases, either by browsing the data series, or through a keyword search. The non-restrictive licensing terms made this data service an ideal source for this project.

The following statistical indicators were downloaded from UNdata in an XML format:

¹⁸ <http://code.google.com/apis/maps/terms.html>

¹⁹ <http://data.un.org>

Table 2: UNdata indicators

Indicator	Years
AIDS estimated deaths, aged 0-49	2001 - 2005
Children under five mortality rate	1960 - 2005
CO2 emissions	1980 - 2004
CO2 emissions per capita	1980 - 2004
GDP per capita	1975 - 2006
Infant mortality rate (0-1 year)	1960 - 2005
Internet users	1990 - 2005
Internet users per 100 population	1990 - 2005
Life expectancy at birth (both sexes)	1950 - 2045
Life expectancy (females)	1950 - 2045
Life expectancy (males)	1950 - 2045
Mobile phone subscribers	1980 - 2004
Mobile phone subscribers per 100 inhabitants	1980 - 2006
Patent applications	1984 - 2002
Percentage aged 0-14	1950 - 2050
Percentage aged 15-59	1950 - 2050
Percentage aged 60 or over	1950 - 2050
Population	1950 - 2050
Tuberculosis death rate	1990 - 2005

The table shows the long time series for some of the indicators - up to 100 years, including projections. The XML format did not include metadata like indicator name and description, and this information had to be copied manually.

3.3 World borders dataset

A world borders dataset was needed for choropleth mapping and to calculate the positions (centroids) of the proportional symbols. Ideally, border data and country statistics should be from the same source, to ensure that the areas are identical. The *United Nations International Boundaries Project*²⁰ offers a 1:1 and 1:15 million-scale dataset, which reflects the cartography practice of the UN Cartographic Section. Unfortunately, this dataset is only accessible to the UN community, due to sensitivity linked to international boundaries.

A world borders Shapefile meeting the above requirements was downloaded from the Mapping Hacks website²¹. This dataset was originally derived by Schyler Erle from public domain sources.

Various changes were made to make the dataset more suitable for thematic mapping (all changes are listed in appendix A):

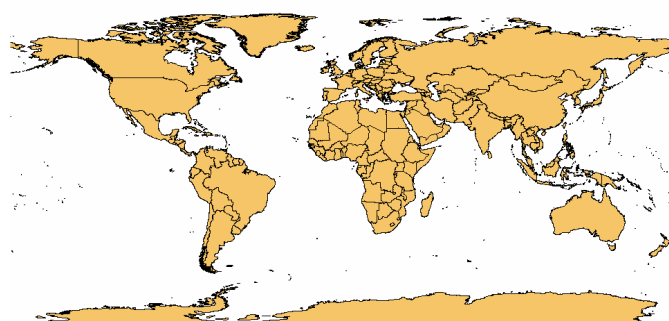


Figure 11: The world borders Shapefile displayed with an Equirectangular (Plate Carrée) projection.

²⁰ <http://boundaries.ungis.org>

²¹ <http://mappinghacks.com/data>

- Added geographic coordinate system: GCS_WGS_1984²²
- Polygons representing one country/area were merged into one feature.
- Various feature changes to make the dataset more compatible with the ISO 3611-1 country codes used by the United Nations.
- Region and sub-region codes from the UN Statistical Division were added.
- Longitude/latitude values for each country were added.

Generalisation is the process of reducing the information content of maps due to scale change, map purpose, intended audience, and/or technical constraints (Slocum et al. 2005). When doing thematic mapping on the web, it is important to limit the amount of data transferred between the server and the mapping client. In addition, modern web browsers are only capable of displaying low-resolution vector data.

The original world borders dataset consisted of 3775 polygons. By deleting “small” island polygons, this number was reduced to 463. This was done manually to avoid the deletion of valid island states. Figure 12 and 13 show the effect on the Caribbean. All countries in the region are still represented.

The remaining polygons were simplified using the *Simplify Polygon* tool in the ArcToolbox. The polygon simplification algorithm (POINT_REMOVE) keeps the so-called critical points that depict the essential shape of the polygon and removes all other points. The tolerance, that determines the degree of simplification, was set to 0.3 decimals degrees. Figure 14 shows the effect of the simplification algorithm. Even though the borders are greatly simplified, the main country shapes are preserved.

To be able to generate proportional symbol maps, a latitude/longitude position was needed for each country. GeoDa²³ was used to compute the centroids of each country. The positions had to be manually adjusted



Figure 12: The resolution of the original dataset.



Figure 13: After removing island polygons.

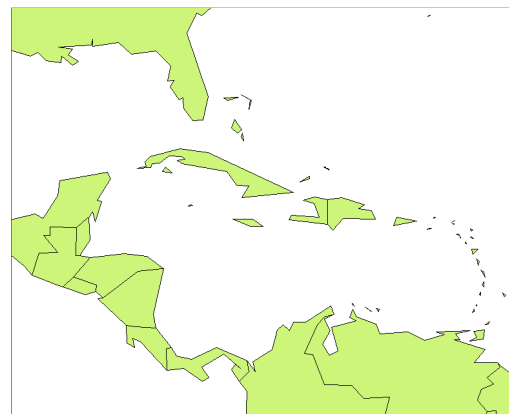


Figure 14: After simplifying borders.

²² The reference system used by KML is based on geographic coordinates (latitude and longitude) in WGS84 (World Geodetic System, 1984). This is the only reference system currently supported by KML (Wilson, 2008).

²³ <https://www.geoda.uiuc.edu>

due to the uneven shape of many countries. The longitude/latitude values were added by inserting two extra columns in the attribute table. Appendix A lists all of the attribute columns in the world borders Shapefile.

3.4 International country codes

A common identifier was needed to be able to link the statistical values to spatial features. The original world borders dataset contained FIPS codes, while UNdata uses ISO 3166-1.

Codes for the Representation of Countries (ISO 3166-1)²⁴ is maintained by the International Organisation for Standardisation (ISO). The standard includes two- and three-character alphabetic codes (alpha-2 and alpha-3) and three-digit numerical codes (numeric-3). *Countries, Dependencies, Areas of Special Sovereignty, and Their Principal Administrative Divisions* (FIPS 10)²⁵ is maintained by the United States National Geospatial-Intelligence Agency, and intended for general use throughout the US Government. The country codes were downloaded from the World Factbook²⁶ and checked against the listing from UN Statistical Division²⁷.

All ISO 3166-1 country codes (alpha-2, alpha-3 and numeric-3) and the FIPS codes (when applicable) were added to the attribute table of the world borders Shapefile. Only ISO 3166-1 numeric-3 was used for this project, but the other codes make it easier to link in other data sources if this project were to be extended.

²⁴ http://www.iso.org/iso/country_codes.htm

²⁵ <http://www.itl.nist.gov/fipspubs/fip10-4.htm>

²⁶ <https://www.cia.gov/library/publications/the-world-factbook/appendix/appendix-d.html>

²⁷ <http://unstats.un.org/unsd/methods/m49/m49alpha.htm>

4. Database

The world borders dataset and statistics from UNdata were stored in a MySQL²⁸ database. MySQL was chosen more because of its availability than its spatial capabilities. MySQL has limited support for spatial data, but it was sufficient for this project. An alternative would be to use the PostgreSQL²⁹ database with the PostGIS³⁰ spatial extension, but this would exclude many potential users since PostGIS is seldom pre-installed by hosting providers.

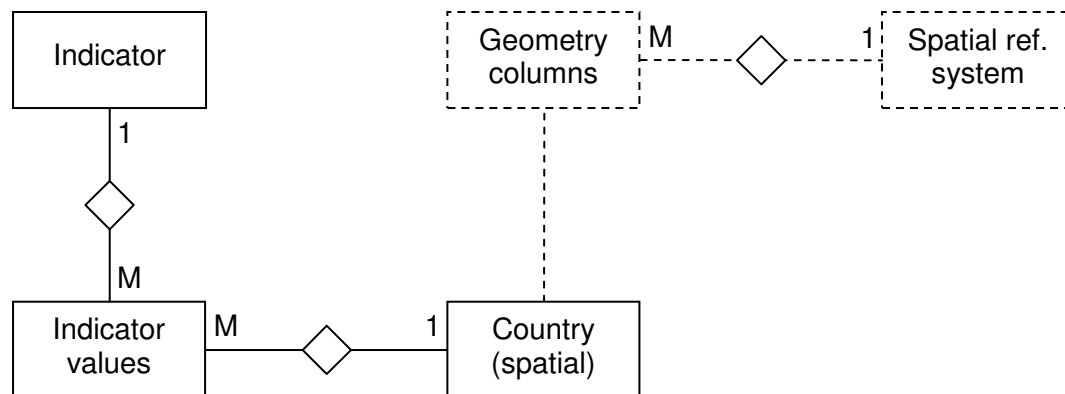


Figure 15: Entity-Relationship (ER) diagram showing the database structure. All table columns are listed in appendix B.

Figure 15 shows an Entity-Relationship (ER) diagram of the database structure. One indicator can have several indicator values, and one country (feature) can have several indicator values. All indicator values are therefore kept in a separate table, with indicator id and country id as foreign keys.

4.1 Uploading spatial data

The GDAL/OGR library³¹ was used to upload the world borders Shapefile to the MySQL database. This library contains the *ogr2ogr* utility program which converts simple features data between various file and database formats. The following command was used to upload the features (including the attribute table):

```
ogr2ogr -f MySQL MySQL:myDatabase, user=root,password=myPassword
TM_WORLD_BORDERS_SIMPL-0.1.shp -nln Country -nlt MULTIPOLYGON -
update -overwrite -lco GEOMETRY_NAME=geom
```

The country borders were stored in the database using the *MultiPolygon* datatype (MySQL, 2008). This datatype was needed because many countries consist of several polygons (i.e. land areas/islands). Some of the polygons are complex (i.e. contains

²⁸ <http://www.mysql.com>

²⁹ <http://www.postgresql.org>

³⁰ <http://postgis.refrations.net>

³¹ Geospatial Data Abstraction Library (GDAL/OGR) is a cross platform translator library for raster and vector geospatial data formats that is released under an X/MIT style open source license by the Open Source Geospatial Foundation (OSGeo). <http://www.gdal.org>

holes) because of enclaves. The longitude/latitude position for each country was stored in two table columns. An alternative would be to use a spatial datatype (GEOMETRY POINT). The above command also created two extra tables (*geometry_columns* and *spatial_ref_system*). These tables are required according to the *OpenGIS Simple Features Implementation Specification for SQL* (OGC, 1999; Butler, 2006).

4.2 Uploading statistical data

The statistical indicators were downloaded from UNdata in an XML format, and uploaded to the MySQL database through a tailored PHP script:

```
// Check if file exists
if (file_exists($file)) {
    // Load XML file
    $xml = simplexml_load_file($file);

    // Loop through all records
    foreach ($xml->data->record as $record) {
        // Loop through all fields in a record
        foreach ($record->field as $field) {
            // Store field values
            switch ($field['name']) {
                case 'Variable Code':
                case 'Series Code':
                    $indicator = (int)$field;
                    break;
                case 'Country or Area Code':
                    $country = (int)$field;
                    break;
                case 'Year':
                case 'Year(s)':
                    $year = (int)$field;
                    break;
                case 'Value':
                    $value = (float)$field;
                    break;
            }
        }
        // Add record to database table
        $sql = "INSERT INTO indicator_values
            (indicator, country, year, value)
            VALUES ($indicator, $country, $year, $value);";

        if (@mysqli_query($dbc, $sql) === TRUE)
        {
            printf("Query: $sql\n");
        }
    }
} else { exit("Failed to open $file."); }
```

The script above reads and parses an XML file. Since it was impossible to download indicator metadata from UNdata, this information had to be entered manually in *phpMyAdmin*³².

³² <http://www.phpmyadmin.net>

4.3 Querying and transforming spatial data

MySQL stores spatial data in an internal geometry format (MySQL, 2008). The data can only be retrieved in this internal format or as *Well-Known Text* (WKT) or *Well-Known Binary* (WKB) representations (OGC, 1999). WKT/WKB predates the Geographical Markup Language (GML), which is now a more commonly used format.

By using the *AsText* function, MySQL retrieves geometry in WKT format. This SQL query returns the simplified border of Italy:

```
SELECT AsText(geom) FROM `country_simpl` WHERE un=380
```

Result:

```
MULTIPOLYGON(((15.528889 38.13694,15.081388 36.649162,12.422222
37.796104,13.316666 38.21833,15.528889 38.13694)),((9.513332
41.14666,9.5691660000002 39.150551,8.406944 38.958611,8.192499
40.913605,9.513332 41.14666)),((12.127777 47.001663,13.718655
46.526611,13.383055 46.297218,13.716944 45.596107,12.281387
45.468048,12.368332 44.246666,13.621666 43.553886,14.739721
42.085548,16.141941 41.914162,15.932499 41.47805,18.512218
40.136665,18.349442 39.791939,16.913609 40.445549,16.486664
39.767494,17.169167 38.963333,16.062496 37.924164,15.631666
38.011665,16.2225 38.910828,15.666666 40.03083,11.098888
42.393051,10.107498 44.0075,8.7488880000001 44.429161,7.528055
43.788605,7.662222 44.17083,6.9763880000001 44.284164,6.61976
45.110138,7.1277770000001 45.257774,7.038054 45.931938,7.855742
45.919052,8.4363880000001 46.463333,9.036665 45.837776,9.2819440000001
46.495827,10.129999 46.227219,10.471235 46.871353,12.127777
47.001663),(12.459166 43.896111,12.509998 43.986938,12.415798
43.957954,12.459166 43.896111),(12.445090330889
41.903117521785,12.456660170954 41.901426024699,12.451653339581
41.907989033391,12.445090330889 41.903117521785)))
```

Since the WKT and KML formats are very different, it was not straightforward to achieve a conversion of complex polygons. This function was made for the conversion:

```
public function wkt2kml($wkt) {
    // Change coordinate format
    $wkt = preg_replace( "/([0-9\\.\\-\\+]) ([0-9\\.\\-\\+]),*/e",
        "round('$1',2).','round('$2',2).','0 '",
        $wkt );

    $wkt = substr($wkt, 15); // Remove 'MULTIPOLYGON(((
    $wkt = substr($wkt, 0, -3); // Remove ')))'
    $polygons = explode(')'),(($wkt); // Find all polygons
    $kml = '<MultiGeometry>';

    foreach ($polygons as $polygon) {
        $kml .= '<Polygon>';
        $boundary = explode('),($', $polygon); // Find all polygon boundaries
        $kml .= '<outerBoundaryIs>'
            . '<LinearRing>'
            . '<coordinates>'
            . self::kmlReverseCoordinates($boundary[0])
    }
}
```

```

        . '</coordinates>'
        . '</LinearRing>'
        . '</outerBoundaryIs>';

    for ($i=1; $i < count($boundary); $i++) { // If inner boundaries
        $kml .= '<innerBoundaryIs>'
            . '<LinearRing>'
            . '<coordinates>'
            . self::kmlReverseCoordinates($boundary[$i])
            . '</coordinates>'
            . '</LinearRing>'
            . '</innerBoundaryIs>';
        }
        $kml .= '</Polygon>';
    }
    $kml .= '</MultiGeometry>';

    return $kml;
}
}
}

```

The coordinates returned from the MySQL database had 6 or more decimal points. This is far more than needed, and does not reflect the precision of the simplified borders. The function above reduces the number of decimal points to 2.

When the KML polygons were used to create a 3-D prism map (see section 5.7), the prisms were not shaded properly (figure 16). The reason for this turned out to be the clockwise orientation of the polygon vertices (winding order). 3-D implementations of KML use the vertex winding order to determine the direction in which it faces³³. This is necessary to display the correct lighting on curved surfaces. For the prisms to be displayed properly, the vertex order had to be anti-clockwise.

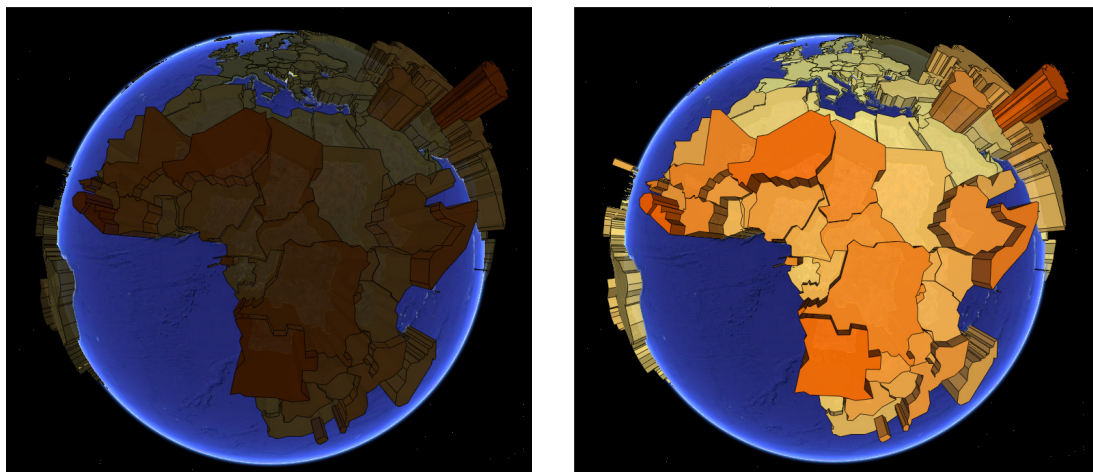


Figure 16: The left image shows a prism map rendered in Google Earth, where the polygon vertices are in a clockwise order. In the right-hand image, the vertex order is counter-clockwise, and the polygons are properly shaded.

³³ <http://bbs.keyhole.com/ubb/showflat.php/Cat/0/Number/166922>

The vertex order was corrected by adding this PHP function:

```
function kmlReverseCoordinates($coordinates) {
    $coordinates = explode(" ", $coordinates);
    $coordinates = array_reverse($coordinates);
    $coordinates = implode(" ", $coordinates);
    return $coordinates;
}
```

The above *wkt2kml* function returns this KML structure representing the Italian border:

```
<MultiGeometry>
  <Polygon>
    <outerBoundaryIs>
      <LinearRing>
        <coordinates>
          15.53,38.14,0 13.32,38.22,0 12.42,37.8,0 15.08,36.65,0
          15.53,38.14,0
        </coordinates>
      </LinearRing>
    </outerBoundaryIs>
  </Polygon>
  <Polygon>
    <outerBoundaryIs>
      <LinearRing>
        <coordinates>
          9.51,41.15,0 8.19,40.91,0 8.41,38.96,0 9.57,39.15,0
          9.51,41.15,0
        </coordinates>
      </LinearRing>
    </outerBoundaryIs>
  </Polygon>
  <Polygon>
    <outerBoundaryIs>
      <LinearRing>
        <coordinates>
          12.13,47,0 10.47,46.87,0 10.13,46.23,0 9.28,46.5,0 9.04,45.84,0
          8.44,46.46,0 7.86,45.92,0 7.04,45.93,0 7.13,45.26,0
          6.62,45.11,0 6.98,44.28,0 7.66,44.17,0 7.53,43.79,0
          8.75,44.43,0 10.11,44.01,0 11.1,42.39,0 15.67,40.03,0
          16.22,38.91,0 15.63,38.01,0 16.06,37.92,0 17.17,38.96,0
          16.49,39.77,0 16.91,40.45,0 18.35,39.79,0 18.51,40.14,0
          15.93,41.48,0 16.14,41.91,0 14.74,42.09,0 13.62,43.55,0
          12.37,44.25,0 12.28,45.47,0 13.72,45.6,0 13.38,46.3,0
          13.72,46.53,0 12.13,47,0
        </coordinates>
      </LinearRing>
    </outerBoundaryIs>
    <innerBoundaryIs>
      <LinearRing>
        <coordinates>
          12.46,43.9,0 12.42,43.96,0 12.51,43.99,0 12.46,43.9,0
        </coordinates>
      </LinearRing>
    </innerBoundaryIs>
    <innerBoundaryIs>
      <LinearRing>
        <coordinates>
          12.45,41.9,0 12.45,41.91,0 12.46,41.9,0 12.45,41.9,0
        </coordinates>
      </LinearRing>
    </innerBoundaryIs>
  </Polygon>
</MultiGeometry>
```

KML

This method of representing geometry is based on GML 2.1.2 (Lake, 2005).

5. Thematic mapping techniques for KML

This section describes how the KML standard was utilised for thematic mapping. The techniques introduced in the Research Paper are here described in detail.

5.1 The KML styling mechanism

In KML, features and geometries are specified along with their styles. KML does not support more flexible mechanisms, such as Cascading Style Sheets (CSS) or OpenGIS Symbology Encoding, which separates content from style. A KML *Style* element can contain the *substyle* elements listed in table 3 (Google, 2008a):

Table 3: KML style elements and the thematic mapping techniques where they are used.

Style element	Thematic mapping technique
IconStyle	Scaled image icons (5.3.1)
PolyStyle	Scaled regular polygons (5.3.2), choropleth (5.6) and prism (5.7) maps
LineStyle	Not in use (could be used for <i>flow maps</i>)
ListStyle	Temporal maps (5.8)
BalloonStyle	All maps
LabelStyle	All maps

Styles can be *shared* (i.e. defined once and used by several features) or *inline* (i.e. specified inside the feature that uses them).

5.2 Mathematical scaling of point symbols

Mathematical scaling sizes the point symbols in direct proportion to the data (Slocum et al., 2005). If a data value is 10 times that of another, the height, area or volume of the corresponding point symbol will be 10 times as large. Below are the formulae used for calculating the symbol sizes. These equations could not be embedded in a KML document, so the mathematical scaling had to be performed by a PHP script.

1-dimensional symbols (height)

Equation $\text{symbolSize} = (\text{value} / \text{maxValue}) * \text{maxSize}$

PHP $\text{\$symbolSize} = (\text{\$value} / \text{\$maxValue}) * \text{\$maxSize}$

This is how the height of the bars or prisms are calculated.

2-dimensional symbols (area)

Equation $\text{symbolSize} = \text{power}(\text{value} / \text{maxValue}; 1/2) * \text{maxSize}$

PHP $\text{\$symbolSize} = (\text{\$value} / \text{\$maxValue}) * \text{\$maxSize}$

This is how the proportional images and regular polygons (e.g. circle, square) are scaled. 2-D symbols use areas as the mean of expression and thereby a square root of the value.

3-dimensional symbols (volume)

Equation $\text{symbolSize} = \text{power}(\text{value} / \text{maxValue}; 1/3) * \text{maxSize}$

PHP $\text{symbolSize} = \text{pow}(\$value / \$maxValue, 1/3) * \$maxSize$

This is how the 3D Collada objects (e.g. cube, sphere) is scaled. 3-D objects use volume as the mean of expression to show the cube root of the value.

It is one degree harder for the viewer to assess the relative size of 3-dimensional symbols compared to 2-dimensional ones (see figure 17), which again are harder to compare that 1-dimensional symbols.

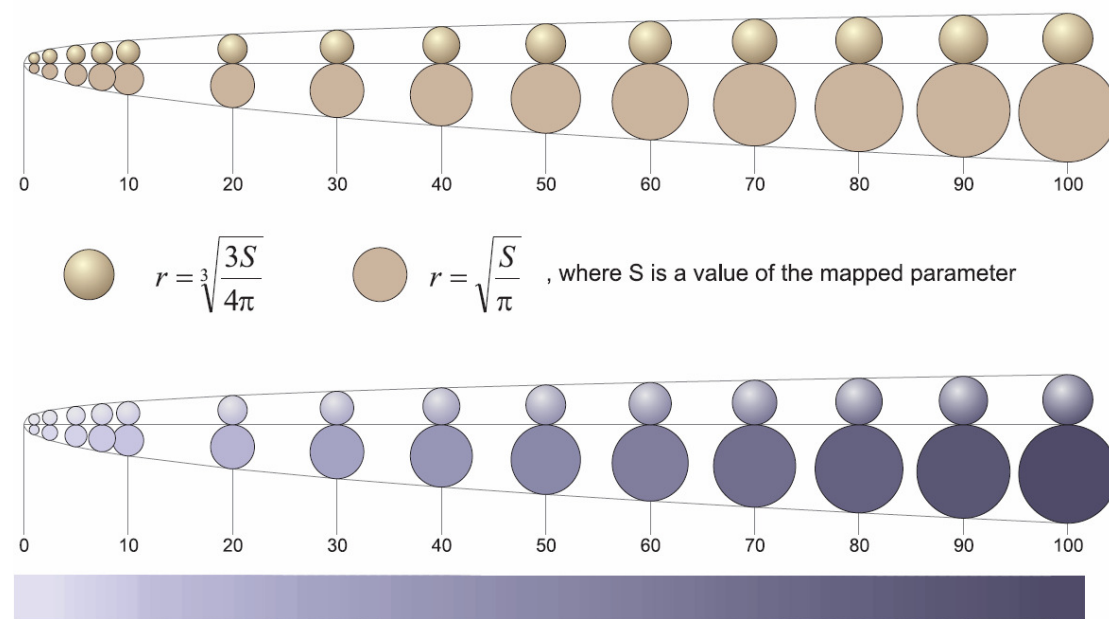


Figure 17: Comparison of the circle and sphere radius for the same mapped parameter. Credit: Dominik Mikiewicz

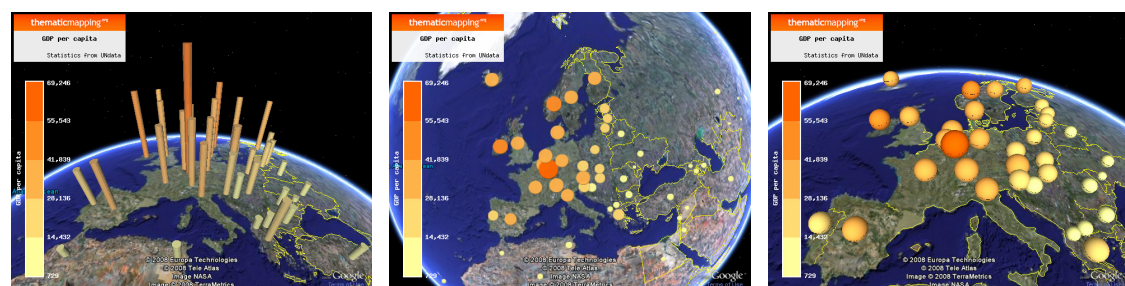


Figure 18: These three images show GDP per capita in 2006 (UNdata) using bars (1-D), circles (2-D) and spheres (3-D). Proportional symbols are not dependant on the size of the spatial unit associated with its attribute. A small country will have the same visual appearance as a big country if their respective attribute values are the same.

People tend to underestimate the size of larger symbols, and an alternative to mathematical scaling is *perceptual scaling* (Slocum et al. 2005). The underestimation is here taken into account when constructing the formula. The visual appearance could also be improved by using a *logarithmic scale*.

5.3 Proportional symbols

This section describes in detail three different ways of making proportional symbols in KML.

5.3.1 Proportional image icons

KML icons are used to visualise various point data, and custom icons can be added by referencing an image stored on the local file system or a remote web server.

Two symbols, a circle and a square, were created using Adobe Photoshop Elements. A shadow effect was added to give the icons a slightly 3-D appearance. The symbols are white on a transparent background, and saved as PNG files. An image size of 200 x 200 px seems appropriate when the image is scaled proportional to an attribute value.

KML icon images can be scaled and coloured by using the *IconStyle* element. Only one image is thereby needed to create symbols in different colours and sizes. This reduces the total file size and improves the performance. The symbol image is referenced in a shared style:

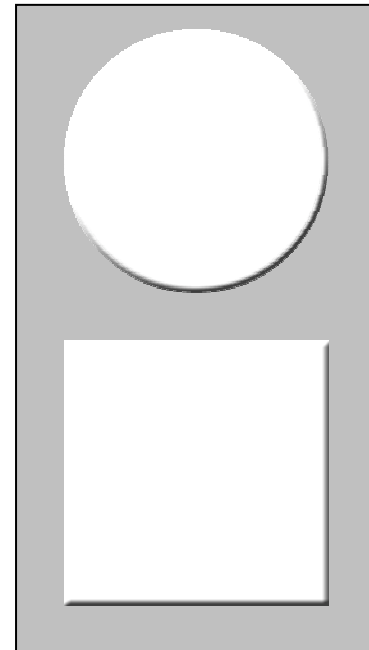


Figure 19: The KML icon images used to make proportional symbols.

```
<Style id='sharedStyle'>
  <IconStyle>
    <Icon>
      <href>files/symbol.png</href>
    </Icon>
  </IconStyle>
</Style>
```

KML

The colour can also be specified in the shared style if only one colour is used. If the colour varies, it needs to be defined as an inline style for each feature, together with the scaling factor:

```
<Placemark>
  <name>China</name>
  <Snippet>1,312,978,855 (2005)</Snippet>
  <styleUrl>#sharedStyle</styleUrl>
  <Style>
    <IconStyle>
      <color>e50066ff</color>
      <scale>7</scale>
    </IconStyle>
  </Style>
  <Point>
    <coordinates>106.514,33.42,0</coordinates>
  </Point>
</Placemark>
```

KML

When a KML *Point* element is contained by a *Placemark* element, the point itself determines the position of the *Placemark*'s image icon (Google, 2008a). This is achieved by defining the longitude, latitude and (optional) altitude within the *coordinates* element.

The drawback of this method is that the map is, so far, only viewable in Google Earth. Other geobrowsers are unable properly to scale and colourise the icon images (see figure 21).

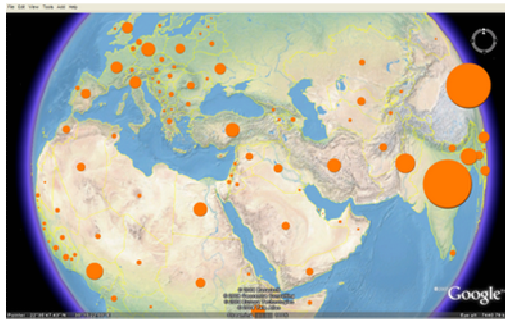


Figure 20: Proportional symbol map in Google Earth showing the relative population in each country of the world.



Figure 21: The same KML file shown in Microsoft Virtual Earth. The image icons are not scaled or colourised.

When comparing the results from the Google Earth plug-in with the Google Earth desktop program, there is a noticeable difference: the icons are much bigger in the plug-in. The reason is different viewport sizes, and it can be considered as a Google Earth bug. There are two different ways of adjusting the size of the planet in Google Earth. The left-hand visualisation in figure 22

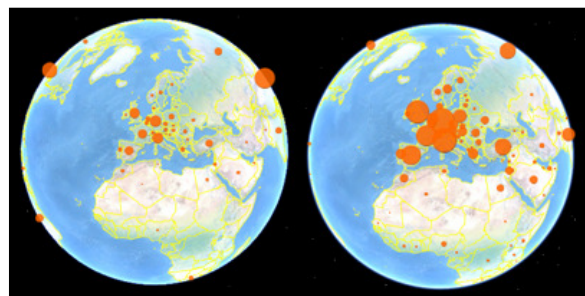


Figure 22: Proportional image icons in Google Earth. The circles on the right planet are not properly scaled.

shows the earth in a zoomed out view. The circle images are scaled properly. The other way of changing the size of the planet is by adjusting the Google Earth window. The problem is that the circle images maintain their size while the planet shrinks or expands. The KML *Icon* element is here used in a way that was probably not intended.

5.3.2 Drawing regular polygons

KML has no built-in support for regular polygons so these have to be generated by calculating the longitude/latitude location for each vertex of the polygon.

A PHP function from the Google Earth Community forum³⁴ was modified for this purpose. The modified function generates a KML polygon from a set of parameters: longitude, latitude, radius, number of vertices and (optional) altitude. This is, of

³⁴ <http://bbs.keyhole.com/ubb/showflat.php?Cat=&Board=SupportKML&Number=166379>

course, a complicated way of making regular polygons. Part of the function is shown below:

```
function kmlSymbolCalculator( $longitude, $latitude, $radius, $vertices,
                             $altitude ) {
    $EARTH_RADIUS_EQUATOR = 6378140.0;
    $RADIAN = 180 / pi();
    $long = $longitude / $RADIAN;
    $lat = $latitude / $RADIAN;
    $f = 1/298.257;
    $e = 0.08181922;

    for ( $bearing = 45; $bearing <= 405; $bearing += 360/$vertices ) {
        $b = $bearing / $RADIAN;
        $R = $EARTH_RADIUS_EQUATOR * (1 - $e * $e) / pow( (1 - $e*$e *
            pow(sin($lat),2)), 1.5);
        $psi = $distance/$R;
        $phi = pi()/2 - $lat;
        $arccos = cos($psi) * cos($phi) + sin($psi) * sin($phi) * cos($b);
        $latA = (pi()/2 - acos($arccos)) * $RADIAN;
        $arcsin = sin($b) * sin($psi) / sin($phi);
        $longA = ($long - asin($arcsin)) * $RADIAN;
        (...)
    }

    (...)
    return $kml;
}
```

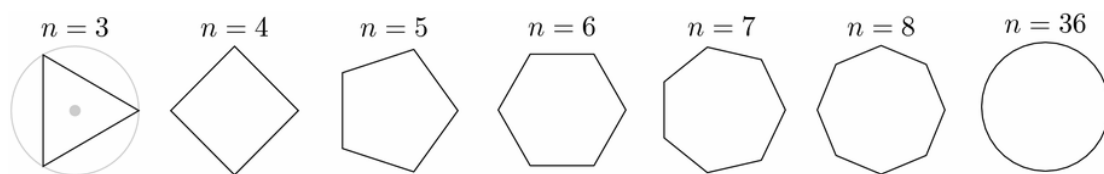


Figure 23: Various regular polygons. n refers to the number of vertices.

The polygon style (fill and outline) is specified in a shared style:

```
<Style id='sharedStyle'>
  <PolyStyle>
    <fill>1</fill>
    <outline>1</outline>
  </PolyStyle>
  <LineStyle>
    <color>cc000000</color>
  </LineStyle>
</Style>
```

KML

The fill colour is specified for each feature, since the regular polygons can be coloured according to a statistical value:

```

<Placemark>
  <name>China</name>
  <Snippet>1,312,978,855 (2005)</Snippet>
  <styleUrl>#sharedStyle</styleUrl>
  <Style>
    <PolyStyle>
      <color>e50066ff</color>
    </PolyStyle>
  </Style>
  <Polygon>
    <outerBoundaryIs>
      <LinearRing>
        <coordinates>
          102.77,36.48,0 102.77,30.24,0 110.26,30.24,0
          110.26,36.48,0 102.77,36.48,0
        </coordinates>
      </LinearRing>
    </outerBoundaryIs>
  </Polygon>
</Placemark>

```

KML

The *Polygon* element includes the coordinates of the vertices in the polygon, calculated by the *kmlSymbolCalculator* function above.

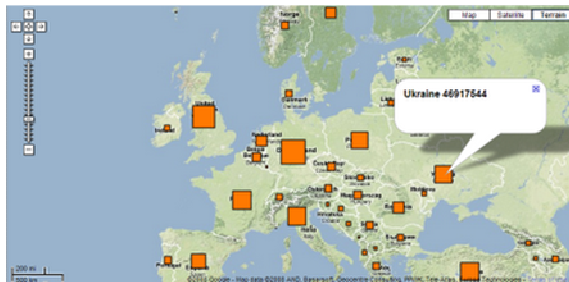


Figure 24: Showing regular polygons in Google Maps.



Figure 25: The polygons are skewed when visualised on a 3-D globe.

5.3.3 Proportional 3-D Collada objects

One unique feature to KML is the ability to embed 3-D models or objects into the KML file (Turner, 2006). The 3-D objects used in this project were downloaded from 3D Warehouse³⁵, an online repository for 3D models. The objects were edited in Google SketchUp³⁶ to make them more suitable for thematic mapping:

1. Each object was positioned so that the green and red lines were touching the bottom of the object. These lines mark the ground when the object is rendered. The blue line marks the latitude/longitude position where the object will be placed, and should bisect the object (figure 26).
2. The objects from 3D Warehouse had various scales. The Tape Measure Tools was used to apply a similar scale to all objects (figure 27).
3. A default colour was added (figure 28).

³⁵ <http://sketchup.google.com/3dwarehouse>

³⁶ <http://sketchup.google.com>

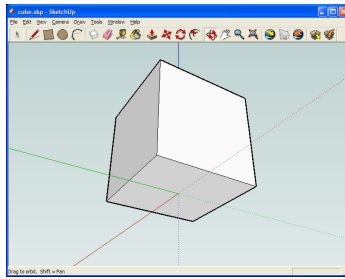


Figure 26: How an object should be positioned in Google SketchUp.

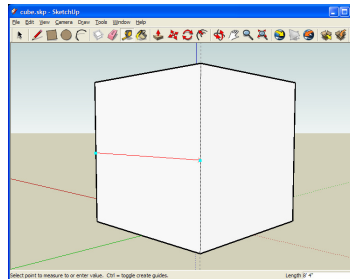


Figure 27: Use the *Tape Measure Tool* to change the scale of the object.

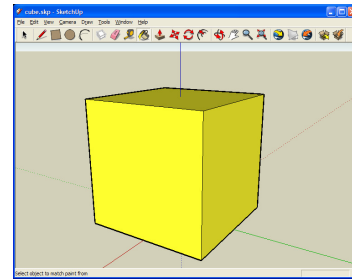


Figure 28: Use the *Paint Bucket* tool to add a default colour to the object. Click on the *Create Material* icon to select a colour.

The KML *Model* element specifies the location, orientation and scale of a Collada object. It is impossible to attach any styles to the *Model* element, which makes it harder to use the feature for thematic mapping.

```
<Placemark>
  <name>China</name>
  <Snippet>1,312,978,855 (2005)</Snippet>
  <Model>
    <Link>
      <href>files/object.dae</href>
    </Link>
    <altitudeMode>absolute</altitudeMode>
    <Location>
      <longitude>106.514</longitude>
      <latitude>33.42</latitude>
      <altitude>0</altitude>
    </Location>
    <Scale>
      <x>140000</x>
      <y>140000</y>
      <z>140000</z>
    </Scale>
  </Model>
</Placemark>
```

KML

The *Link* element specifies the Collada object to load. The *Location* element specifies the coordinates of the object's origin in latitude, longitude and altitude. The *Scale* element scales the object along the x, y and z axes in the object's coordinate space. As for bars, the z (height) dimension could represent a different statistical indicator than the x/y dimension. Collada objects can also be placed on top of each other by specifying an increasing altitude value (e.g. to create stacked bars).

Since it is impossible to colourise Collada objects in KML, this needs to be done in the Collada file itself. This PHP script shows how this can be achieved:


```

foreach($this->classColours as $class => $classColour){
    $colladaColour = self::rgb2collada($classColour);

    // Read collada model
    $filename = "files/$this->symbolShape.dae";
    $handle = fopen($filename, "r");
    $collada = fread($handle, filesize($filename));
    fclose($handle);

    // Search and replace colour
    $pos = strpos($collada, '<effect id="material0-effect" ...>');
    $pos = strpos($collada, "<diffuse>", $pos);
    $pos = strpos($collada, "<color>", $pos);
    $collada = substr_replace($collada, $colladaColour, $pos+7, 28);

    // Add collada object to kmz archive
    $zip->addFromString("files/object$class.dae", $collada);
}

```

After the Collada text file is read into the memory, the default colour is replaced. Each object is stored using a different filename which can be referenced by the *Model* element.

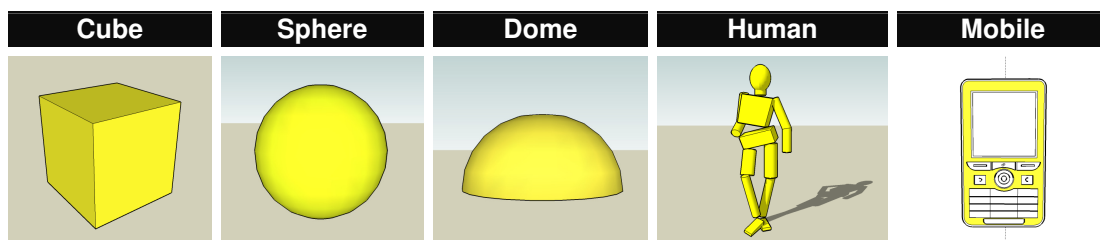


Figure 29: Collada objects available in the *Thematic Mapping Engine*. All objects were downloaded from 3D Warehouse. The human body is made by *Snah* and the mobile phone is made by *Mikeyjm*.

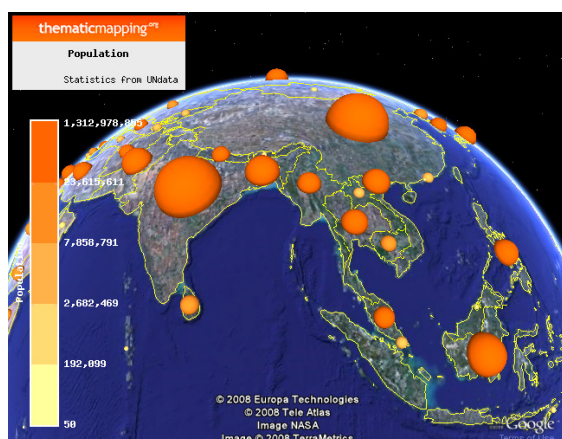


Figure 30: Population (2005) in Southeast Asia visualised with 3-D domes.



Figure 31: Same indicator, but different Collada object.

5.4 Chart maps

*Google Chart API*³⁷ allows chart generation by embedding data and formatting parameters in a URL. The API returns a PNG image of the chart, which can be included on a web page or overlaid onto a map.

The following URL will return the image in figure 32:

```
http://chart.apis.google.com/chart?cht=p&chd=t:20,45,5&chs=300x150&chl=0-14%20years%7C15-59%20years%7C60%20and%20over
```

The URL contains the information needed to generate the pie chart. URLs like this can be specified within the *IconStyle* element as long as it returns a valid image file:

```
<Placemark>
  <name>China</name>
  <Snippet>1,312,978,855 (2005)</Snippet>
  <Style>
    <IconStyle>
      <scale>7</scale>
      <Icon>
        <href>http://chart.apis.google.com/chart?cht=p(...) </href>
      </Icon>
    </IconStyle>
  </Style>
  <Point>
    <coordinates>106.514,33.42,0</coordinates>
  </Point>
</Placemark>
```

KML

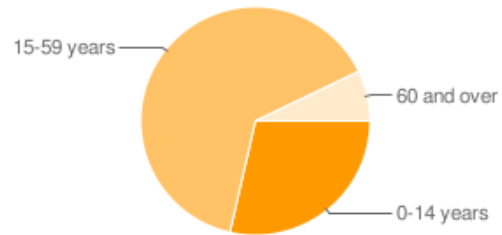


Figure 32: Pie chart showing age distribution.

The chart image itself can also be scaled to represent a statistical value (e.g. the total population).

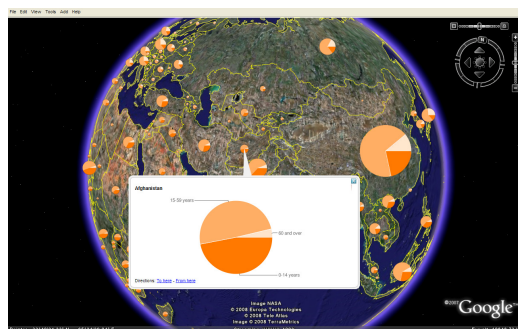


Figure 33: World population (2005) visualised with scaled pie charts. The size of the charts is proportional to the total population of a country, while the pie shows the age distribution.

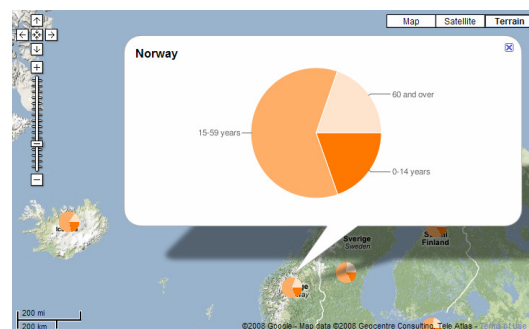


Figure 34: Same KMZ file visualised with Google Maps. The charts are loaded properly, but not scaled according to the population value.

³⁷ <http://chart.apis.google.com>

5.5 Bar maps

The regular polygons described above can be turned into bars by adding an altitude value for each coordinate tuple (vertex). Altitude values are in metres above sea level, and should be directly proportional to the statistical value.

```
<Placemark>
  <name>China</name>
  <Snippet>1,312,978,855 (2005)</Snippet>
  <styleUrl>#sharedStyle</styleUrl>
  <Style>
    <PolyStyle>
      <color>e50066ff</color>
    </PolyStyle>
  </Style>
  <Polygon>
    <extrude>1</extrude>
    <altitudeMode>absolute</altitudeMode>
    <outerBoundaryIs>
      <LinearRing>
        <coordinates>
          102.77,36.48,2000000 102.77,30.24,2000000
          110.26,30.24,2000000 110.26,36.48,2000000
          102.77,36.48,2000000
        </coordinates>
      </LinearRing>
    </outerBoundaryIs>
  </Polygon>
</Placemark>
```

KML

The *extrude* and *altitudeMode* elements are needed to create a 3-D bar. The *extrude* element specifies that the polygon is connected to the ground. When the *altitudeMode* is *absolute*, the altitude of each coordinate tuple is relative to sea level, regardless of the actual elevation of the terrain beneath the element. When bars are rendered on top of a 3-D landscape, the *relativeToGround* attribute should be used.

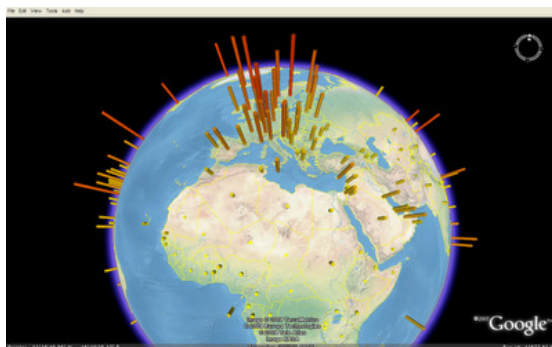


Figure 35: The height and colour of the bars represents Internet users per 100 people, and clearly visualises the digital divide in the world. Each country has the same visual appearance regardless of its population size.

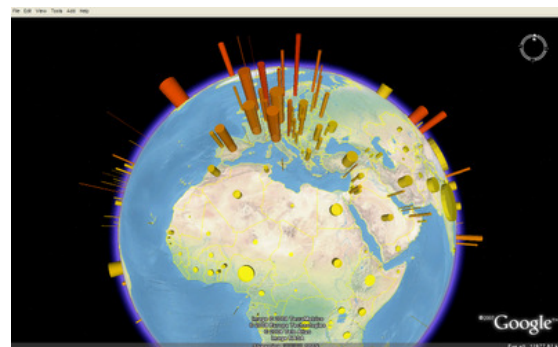


Figure 36: The bar diameter here represents the population of each country, while the bar height and colour still represent the proportion of internet users.

5.6 Choropleth maps

The thematic mapping techniques presented so far are all using the longitude/latitude position (centroid) for each feature. A choropleth map requires the geometry representing the border of the feature (e.g. country). Section 4.3 describes how KML polygons are created from geometry stored in a MySQL database.

The polygon style (fill and outline) is specified in a shared style:

```
<Style id='sharedStyle'>
  <PolyStyle>
    <fill>1</fill>
    <outline>1</outline>
  </PolyStyle>
  <LineStyle>
    <color>cc000000</color>
  </LineStyle>
</Style>
```

KML

The fill colour is specified for each feature, since the polygons are coloured according to a statistical value:

```
<Placemark>
  <name>China</name>
  <Snippet>1,312,978,855 (2005)</Snippet>
  <styleUrl>#sharedStyle</styleUrl>
  <Style>
    <PolyStyle>
      <color>DC0066FF</color>
    </PolyStyle>
  </Style>
  <MultiGeometry>
    <Polygon>
      <outerBoundaryIs>
        <LinearRing>
          <coordinates>
            110.72,20.06,0 109.26,19.9,0 108.63,19.28,0
            108.69,18.51,0 110.05,18.38,0 111.03,19.64,0 (...)
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
    </Polygon>
    <Polygon>
      (...)
    </Polygon>
  </MultiGeometry>
</Placemark>
```

KML

The KML polygons are contained by a KML *MultiGeometry* element. Figure 8 and 9 show how choropleth maps are displayed in Google Earth and Google Maps.

5.7 Prism maps

As regular polygons can be turned into 3-D bars by adding an altitude value for each coordinate tuple (section 5.5), irregular polygons can be turned into prisms.

```
<Placemark>
  <name>China</name>
  <Snippet>1,312,978,855 (2005)</Snippet>
  <styleUrl>#sharedStyle</styleUrl>
  <Style>
    <PolyStyle>
      <color>DC0066FF</color>
    </PolyStyle>
  </Style>
  <MultiGeometry>
    <Polygon>
      <extrude>1</extrude>
      <altitudeMode>absolute</altitudeMode>
      <outerBoundaryIs>
        <LinearRing>
          <coordinates>
            110.72,20.06,2000000 109.26,19.9,2000000
            108.63,19.28,2000000 108.69,18.51,2000000
            110.05,18.38,2000000 111.03,19.64,2000000 (... )
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
    </Polygon>
    <Polygon>
      (... )
    </Polygon>
  </MultiGeometry>
</Placemark>
```

KML

The *extrude* and *altitudeMode* elements have to be specified for each polygon contained by a *MultiGeometry* element. Figures 16 and 38 show examples of a prism map displayed in Google Earth.

When 3-D prism maps are rendered in Google Earth, holes appear in the polygons representing large countries with low values on a statistical indicator (i.e. those with a low altitude value). KML has three parameters for controlling the behaviour of polygons; *extrude*, *tessellate* and *altitudeMode*. By setting *altitudeMode* to *clampToGround*, the country polygons follow the great circle.

The problem arises when the polygons are extruded by adding an altitude representing a statistical value. Only the vertices of the polygon are extruded to the given altitude,

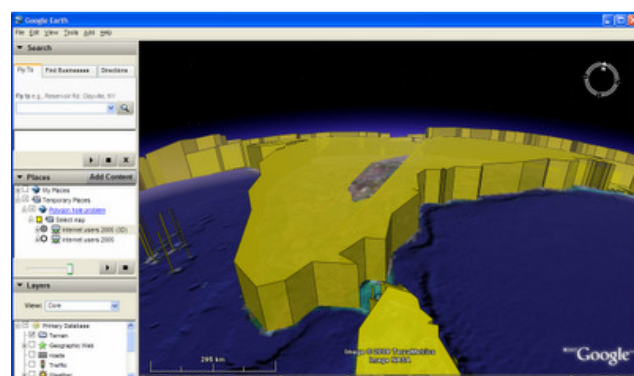


Figure 37: A hole in the polygon representing India.

and not the centre of the geometry. A new a *clampToAltitude* option in KML could solve this problem.

There are some ways of solving this problem:

- Give all polygons a minimum altitude to support a "flat roof". This has to be a high value for a country like Russia.
- Break up large polygons into smaller pieces.
- Add additional *clampToGround* polygons to "hide" the holes. This only works with solid fills (no transparency).

5.8 Temporal maps

The importance of the time dimension has been the subject of considerable recent debate within the fields of cartography and scientific visualisation (Harrower and Fabrikant, 2008; Andrienko et al., 2005; Kraak and van de Vlag, 2007). The same cardinal questions are often referred to (Andrienko et al., 2005): How to make computers understand temporality and handle time-related information, and how can computers support the visual exploration of spatio-temporal information? These questions are tightly connected, as techniques used for graphical representation of data have to correspond to characteristics of the data.

By using the time primitives in KML, one can create animated thematic maps showing how a variable changes over time. The KML standard contains two time elements (Wilson, 2008): *TimeSpan* represents an extent in time bounded by begin and end dates, while *TimeStamp* represents a single moment in time. The time elements are used to limit the display of features to given period of time or instant of time. This is a relatively easy way to provide sophisticated map animations in a geobrowser.

```
<Folder>
  <name>Years</name>
  <open>1</open>
  <Folder>
    <name>2000</name>
    <visibility>1</visibility>
    <TimeSpan>
      <begin>2000-01-01</begin>
      <end>2004-12-31</end>
    </TimeSpan>
    <Placemark>
      <name>Afghanistan</name>
      (...)
    </Placemark>
    (...)
  </Folder>
  <Folder>
    (...)
  </Folder>
  (...)
</Folder>
```

KML

All shaded polygons representing one year of data are wrapped in a KML *Folder* containing a *TimeSpan* element. Whenever a time element is specified in a KML file, the geobrowsers automatically displays a time slider that corresponds to the beginning and ending times in the file. The user can see the animation by using play button or moving the slider itself.

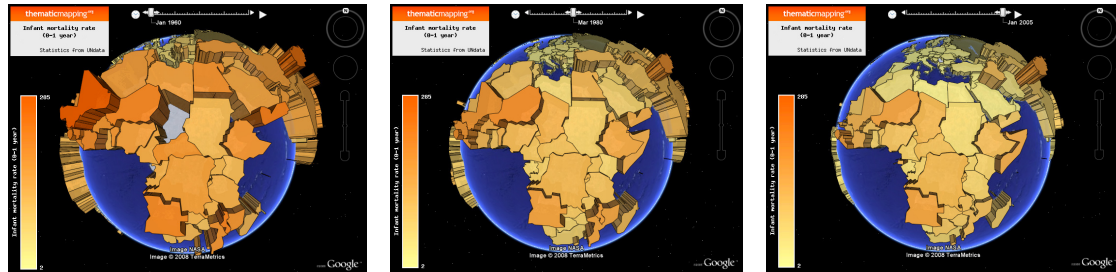


Figure 38: This visualisation shows the declining infant mortality rate in Africa (1960, 1980 and 2005). The Google Earth time slider is located above the 3-D globe.

5.9 Map tiles

A map tile approach is based on a multi-resolution image pyramid. The higher the zoom level, the greater the frequency of tiles required to cover any geographical area (see figure 39 and table 4). This image pyramid is a *quadtree data structure*, in which each region is sub-divided into four quadrants, each of which facilitates an increase in zoom level (Gibin et al. 2008).

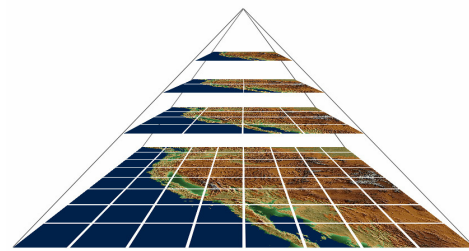


Figure 39: Multi-resolution image pyramid. Credit: OSGeo.org

5.9.1 GDAL2Tiles

GDAL2Tiles³⁸ is a command line tool that allows the easy publishing of raster maps on the Internet. The raster image is converted into a directory structure of small image tiles which can be copied to a web server. Each map tile image generated by GDAL2Tiles is 256 x 256 pixels. Table 4 shows the number of tiles that needs to be generated for each zoom level.

Table 4: Number of map tiles for increasing zoom levels.

Level	Image size	Tiles	Total tiles
1	512 x 256	2 x 1	2
2	1024 x 512	4 x 2	8
3	2048 x 1024	8 x 4	32
4	4096 x 2048	16 x 8	128
5	8192 x 4096	32 x 16	512
6	16384 x 8192	64 x 32	2048
7	32768 x 16384	128 x 64	8192

GDAL2Tiles is compatible with the *Tile Map Service (TMS) Specification*³⁹. TMS has been proposed as a standardised method whereby map tiles are requested by web clients and how servers describe their holdings. TMS is not an official standard and is currently managed by the Open Source Geospatial Foundation (OSGeo). The Open Geospatial Consortium (OGC) has released a discussion paper (Pomakis 2007) describing a tile map extension to the Web Map Service (WMS).

³⁸ GDAL2Tiles is included in the GDAL library from the 1.5.0 release (<http://www.gdal.org>)

³⁹ http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification

Tom Patterson's Natural Earth II⁴⁰ is here used to show how GDAL2Tiles can be used to generate map tiles. The same method can be used for raster-based thematic maps (Gibin, 2008).

If the supplied raster map uses the EPSG:4326 coordinate reference system (WGS84), GDAL2Tiles generates a KML super-overlay (see below) in addition to the map tiles. The map tiles were generated by using the following GDAL commands:

Gdal_translate was first used to georeference the raster map:

```
gdal_translate -a_srs EPSG:4326 -gcp 0 0 -180 90 -gcp 16200 0 180
90 -gcp 16200 8100 180 -90 NE2_modis3.jpg NE2_modis3.tif
```

The image should now have the correct coordinate reference system (EPSG:4326), but GDAL2Tiles was not generating the KML super-overlay before the *gdalwarp* command was used:

```
gdalwarp -t_srs EPSG:4326 NE2_modis3.tif NE2_modis3_4326.tif
```

The original raster map was 16,200 x 8,100 pixels. For this purpose, 5 zoom levels would be sufficient. The map image was therefore reduced to 8192 x 4096 pixels (512 map tiles):

```
gdal_translate -outsize 8192 4096 NE2_modis3_4326.tif
NE2_modis3_4326_5.tif
```

Finally, the GDAL2Tiles command was used to generate the map tiles and the KML super-overlay:

```
gdal2tiles -title "Natural Earth II" -publishurl
http://www.thematicmapping.org/maptiles/ -v NE2_modis3_4326_5.tif
```

5.9.2 KML super-overlay

A KML super-overlay is a hierarchy of KML files that can be used to serve a large set of images (Google, 2008b). Super-overlays take advantage of KML network links and their ability to determine whether a given region is within view and whether its projected size is appropriate for the view area. If both conditions are true, the KML file associated with the region is loaded. This KML files includes a reference to the map tile image. When the user is zooming or panning a tile map in a geobrowser, only the tiles not currently in the memory are requested from the server.



Figure 40: The Natural Earth II map tiles loaded as a KML super-overlay in Google Earth.

⁴⁰ <http://www.shadedrelief.com/natural2>

5.10 Map colours

The use of colour plays an important role in the visualisation and analysis of thematic map data, as it facilitates the observation of the patterns and interrelationships. The various colour models can broadly be divided into two main categories: perceptually-based and display based (Atlas of Canada, 2008). *Perceptually based models*, such as HSB (hue, saturation, brightness), are organised in a way similar to how humans perceive colour in everyday life. In *display-based models*, such as RGB (red, green, blue), the appearance of the colours produced depends on the settings of the display device.

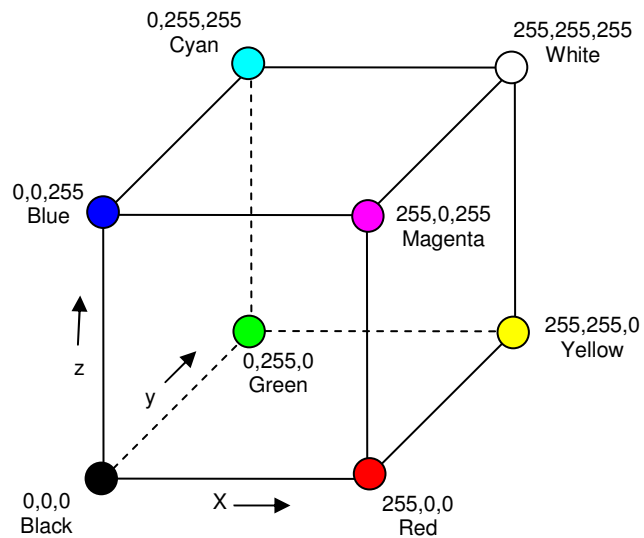


Figure 41: The RGB colour cube (after Robinson et al. 1995)

In the *RGB model*, colours are specified based on the intensity of red, green, and blue colour guns (Slocum et al. 2005). The range of intensities for the colour gun values may be visualised as a cube with positions specified by x, y, z integer coordinates (Robinson et al. 1995). These coordinates control the intensities of the red, green and blue colour guns. The maximum integer value is usually 255, ranging from 0 to 255. This gives 256^3 or 16,777,216 possible colour combinations.

In the Thematic Mapping Engine, the start and end colours are given as six-digit hexadecimal colour codes, which allows a colour value to be specified in a single parameter value. The lowest statistical value is assigned with a start colour, and the largest value is assigned with an end colour. The values between these have their colours calculated by linear interpolation between these two colours. The red, green and blue integer values are extracted from the hexadecimal colour codes. The colour difference or range is calculated by subtracting the start red/green/blue integer from the end red/green/blue integer (linear interpolation). The colour codes for all of the intermediate statistical values can then be calculated.

A disadvantage of the RGB model is that equal steps in the RGB colour space do not correspond to equal visual steps (Slocum et al. 2005). A colour value of 125,0,0 will not appear to fall midway between 0,0,0 and 255,0,0.

KML represents colours in an OBGR format. This format represents each colour as a 32-bit hexadecimal number, with 8 bits each for opacity (transparency), blue, green and red. The ordering of the colour values is different from the usual web colour specification, which is RGB.

5.11 Map legend

Map legends are important in thematic mapping, but KML/Google Earth has no build-in legend support. The best way to add a map legend is to use the KML *ScreenOverlay* element. This element only supports image overlays and not HTML layers. This makes it harder to create a dynamic legend that changes with the data.

As symbol size varies with scale (zoom level), useful symbol legends are difficult to create. An alternative method is to duplicate the symbology by supporting a colour legend for all thematic mapping techniques, also for proportional symbol maps. The colour legend informs the user about the range of values (min and max), and where the different symbols are positioned in this range.

By using an unclassified scheme, a unique visual shade is assigned to each unique data value. The colour scale can also be classed by using equal intervals or quantiles:

Equal intervals

Each colour class occupies an equal interval along the value range. This scheme is easily interpreted by map readers, and is particularly useful for comparing a series of maps (Krygier and Wood 2005). The data distribution is not taken into account, and this classification may result in most data values falling into one of two classes, while other classes have no values. The classes are calculated by this PHP code:

```
$interval = ($this->maxValue - $this->minValue) / $numClasses;
for ($i = 0; $i < $numClasses; $i++) {
    $position = $this->minValue + ($interval * $i);
    $this->classBreaks[] = round($position, $this->precision);
}
$this->classBreaks[] = $this->maxValue;
```

Quantiles

Quantile schemes place the same number of data values in each class. A quantile classification is attractive because it always produces distinct map patterns: it will never have empty classes, or classes with only a few or too many values (Krygier and Wood 2005). The problem with this classification is that it often places similar values in different classes or very different values in the same class. The classes are calculated in this way:

```
$values = array_values($this->indicator['values'][$this->year]);
$numValues = count($values);
$classNum = $numValues / $numClasses; // Number in each class
for ($i = 0; $i < $numClasses; $i++) {
    $position = (int)($classNum * $i);
    $this->classBreaks[] = $values[$position];
}
$this->classBreaks[] = $values[$numValues-1];
```

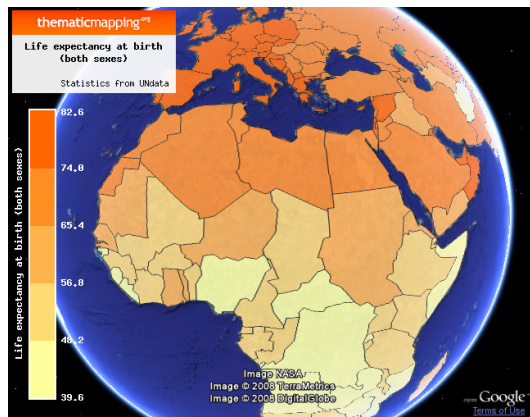


Figure 42: Choropleth map showing life expectancy by using equal intervals.

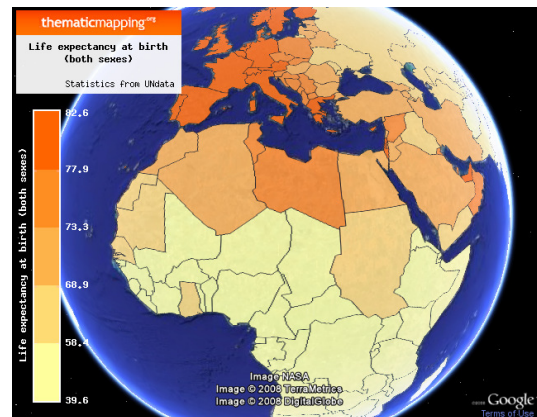


Figure 43: A quantile classification gives a different view.

5.11.1 Creating colour legends with GD

GD⁴¹ is an open source graphics library for the dynamic creation of images by programmers. The library is commonly used to generate graphics on-the-fly in web applications. TME uses the PHP-GD⁴² binding, which is part of the standard PHP installation. The legend generation process can be generalised into the following steps (based on Coggeshall, 2004):

1. Create an legend “canvas” in memory
2. Allocate legend colours
3. Draw the legend
4. Save the legend canvas as a PNG image on the web server

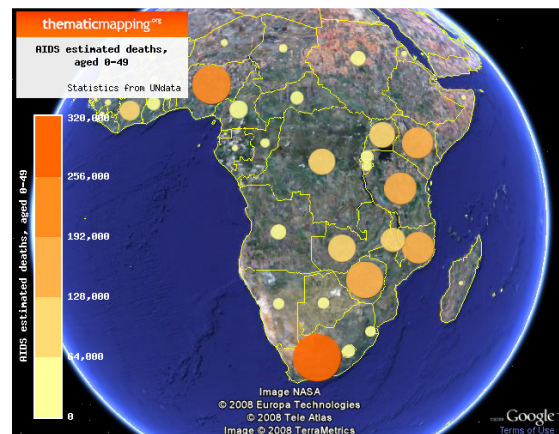


Figure 44: The colour legend is generated by the GD library and referenced by the KML *ScreenOverlay* element. The map title is added by using the same method.

This is performed by the *getColourLegend* method for the *ThematicMap* class. The code listing shows that creating colour scales with GD is a complicated process.

The PNG legend image is included in the KMZ file (see section 5.12) and referenced in the KML document as a *ScreenOverlay*:

⁴¹ <http://www.libgd.org>

⁴² <http://php.net/gd>

```

<ScreenOverlay>
  <name>Legend</name>
  <Icon>
    <href>files/legend.png</href>
  </Icon>
  <overlayXY x='0.01' y='0.14' xunits='fraction' yunits='fraction' />
  <screenXY x='0.01' y='0.14' xunits='fraction' yunits='fraction' />
  <size x='-1' y='-1' xunits='pixels' yunits='pixels' />
</ScreenOverlay>

```

KML

The KML *ScreenOverlay* element specifies an image overlay to be displayed fixed to the screen. Regardless of where the user navigates in the geobrowser, the screen overlay remains in the same position. The position is determined by a point relative to the image, specified by *overlayXY*, to a point relative to the screen, specified by *screenXY* (Wilson, 2008). The *size* element determines the size of the ScreenOverlay. A size value of -1 indicates the use of the native dimension of the image.

5.12 File compression

A series of files are often needed when a thematic map is created using KML: legend image, icon image, 3-D Collada object and the KML file itself. The KML file can also be very large when it contains a lot of features and complex geometries repeated for several time steps. Fortunately, KML files, linked images and 3D objects can be compressed into a single KMZ file. This makes file transfer easier, as only one file needs to be transferred, and more efficient, due to the reduced file size.

The Thematic Mapping Engine uses PHP ZIP functions⁴³ to create KMZ files. Basically, a KMZ file has the same properties as any other ZIP file, except for the file extension. By using the PHP ZIP functions, a new KMZ file can be created and the various files added.

```

$file = "tmp/tme". time(). ".kmz";
$zip = new ZipArchive();
if ($zip->open($file, ZIPARCHIVE::CREATE) !== TRUE) {
    exit("cannot open <$file>\n");
}

// Add colour legend to KMZ archive
if ($this->showLegend) {
    $zip->addFile(self::getColourLegend(), 'files/legend.png');
}

// Add kml to archive
$zip->addFromString("doc.kml", $kml);

$zip->close();

```

⁴³ <http://www.php.net/zip>

6. The thematic mapping website

A dedicated website⁴⁴ was established for this project. The website contains the Thematic Mapping Engine, a blog, the World Borders dataset and information about the project. The website was created in the beginning of March 2008 and has been continuously updated. The website turned out to be very popular and attracted a lot of visitors.

The website made it possible to share ideas with a wider community, and to receive important feedback. Several enquires came from developers who wanted to use the Thematic Mapping Engine as a plug-in to their mapping applications. The blog also enabled me to get in touch with individuals and organisations who are engaged in similar issues.

This turned out to be a crucial way of obtaining information and keeping up-to-date with the latest developments and trends. Blogs and wikis⁴⁵ are now widely used among “geeks” and professionals alike, to present their own work and perspectives and to comment on other people’s work. An increasing amount of the information we now consume digitally is user created (Hudson-Smith and Crooks, 2008).

It can, of course, be problematic to use these kinds of information sources in academic research. It is often hard to measure the credibility of those who write and comment on blogs, and it is not good practice to add references to wikis which might be edited by anyone at any time. This phenomenon has been described as the “cult of the amateur” (Keen, 2007):

The Web 2.0 revolution has peddled the promise of bringing more truth to more people – more depth of information, more global perspective, more unbiased opinion from dispassionate observers. But this is all a smokescreen. What the Web 2.0 revolution is really delivering is superficial observations of the world around us rather than deep analysis, shrill opinion rather than considered judgment. The information business is being transformed by the Internet into the sheer noise of a hundred million bloggers all simultaneously talking about themselves.”

Andrew Keen (2007)

in *The Cult of the Amateur: How Today’s Internet is Killing Our Culture*

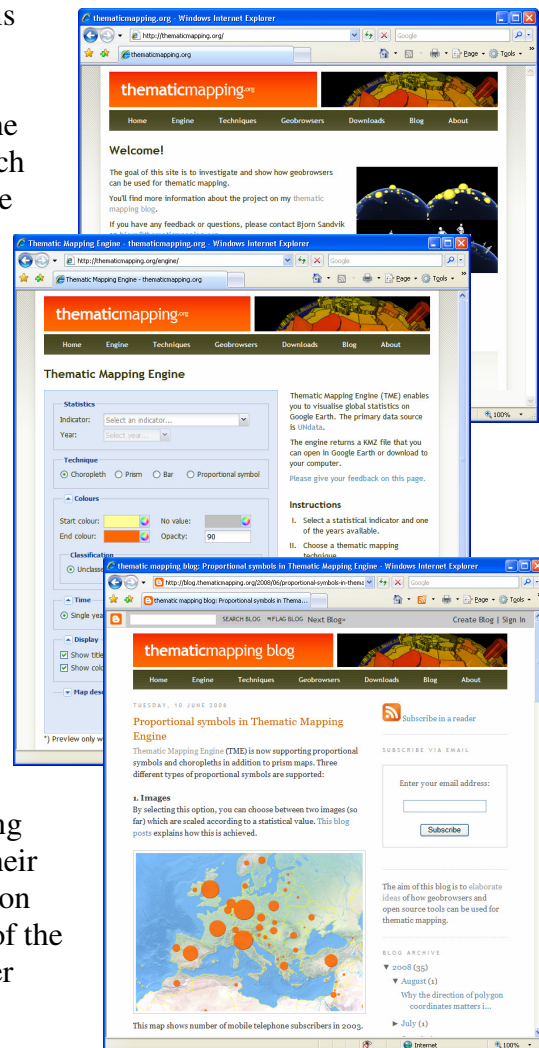


Figure 45: Screenshots from the thematicmapping.org website

⁴⁴ <http://thematicmapping.org>

⁴⁵ A wiki is a collection of web pages designed to enable anyone who accesses it to contribute to or modify its content.

Even though Keen makes some valid points, more specialist and technical oriented blogs, wikis and user forums are arguably a good source of information – in the absence of books and journal articles.

Most of the thematic mapping techniques shown in this document were also presented on the blog. Much feedback was received, which enabled me continually to evaluate and improve my methods. Short reviews of the Thematic Mapping Engine were published on popular GIS blogs:

“Periodically during the last three years since Google Earth was released, some talented developer gets the Google Earth bug and develops a wave of cool stuff. Right now, the guy with the most momentum is Bjørn Sandvik at the United Nations Association (UNA) of Norway. He's written a flood of blog posts in the last few weeks at his Thematic Mapping Blog with really innovative visualizations and applications of GIS technology for Google Earth.”

Frank Taylor – Google Earth Blog 6 June 2008⁴⁶

“I always considered Google Earth to be a bit of a dead fish as far as cartographic elements go, but Bjørn's work shows that some truly amazing representations are possible with a bit of work. If you haven't checked this blog out, give it a read.”

Jason Birch – Random Notes 31 May 2008⁴⁷

“On his excellent Thematic Mapping blog (now part of my blog roll), Bjorn Sandvik has been running a series on thematic mapping techniques in Google Earth. Examples have included choropleth maps, bar charts, prism charts, Collada objects, map legends, including examples of animated time series. If you're into the techy part of this stuff, the series is well worth a look for the ideas you'll get from it, and the downloadable example files. But for non-techy types, the prospect of learning how to do this might have been a bit intimidating. Heck, I find it intimidating, and I know at least a little about this stuff. But Bjorn has taken his examples to the next level, by creating an online tool that can convert datasets to thematic Google Earth maps on the fly, the Thematic Mapping Engine”

Leszek Pawlowicz, Free Geography Tools 2 June 2008⁴⁸

Most of the critical feedback was questioning the effectiveness of 3-D globe visualisations. The criticisms were not directly related to KML, but aimed more towards the use of 3-D visualisation in general. These issues are important, and a response to these critics was published in a series of blog posts⁴⁹. This feedback was used to improve the visualisations created with the Thematic Mapping Engine.

⁴⁶ http://www.gearthblog.com/blog/archives/2008/06/thematic_mapping_engine_plugin.html

⁴⁷ <http://www.jasonbirch.com/nodes/2008/05/31/186/thematic-mapping-blog-rules/>

⁴⁸ <http://freegeographytools.com/2008/thematic-mapping-in-google-earth>

⁴⁹ http://blog.thematicmapping.org/2008_06_01_archive.html

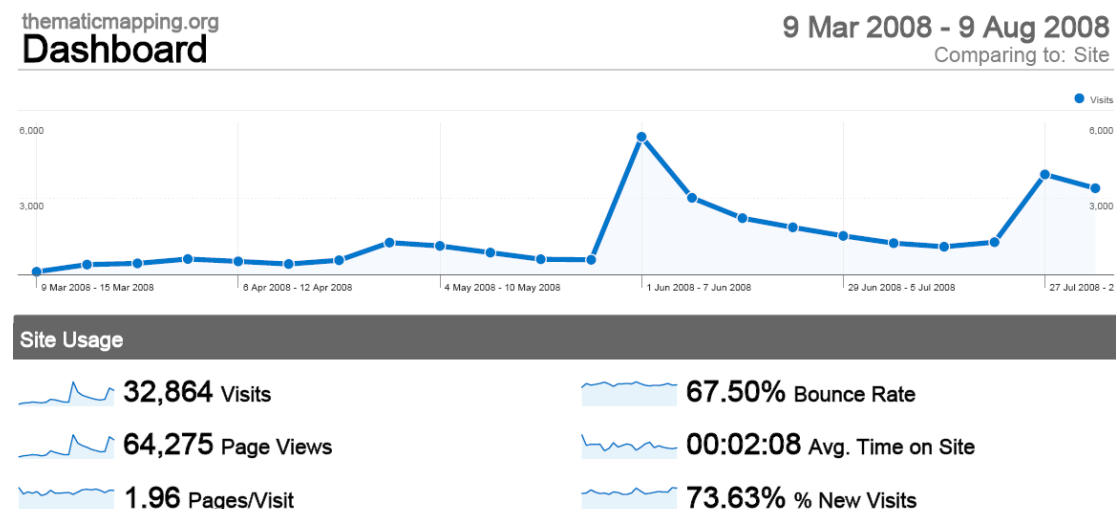


Figure 46: Number of visitors to thematicmapping.org website 9 March – 9 August 2008.

The thematicmapping.org website had more than 30,000 visits between 9 March and 9 August 2008⁵⁰. There was a sharp increase when the Thematic Mapping Engine was launched on 1 June (see figure 46). The second highest number occurred when TME was featured on the various Google sites⁵¹. TME was also featured on the UNdata start page for a month. Most of the visitors came from the United States and United Kingdom (see table 5).

2008 was a perfect time for this project, since three incidents occurred during its evolution: the UNdata portal was launched on 11 February 2008, providing instant access to a wealth of statistical data (UNdata, 2008); on 14 April, KML became an international standard of the Open Geospatial Consortium (OGC) and the “first broadly accepted standard for the visualisation of geographic information” (OGC, 2008); and, on 28 May, the Google Earth Browser Plug-in was released (Rademacher, 2008), bringing the power of Google Earth to the Thematic Mapping Engine.

Table 5: Number of visits from the top ten countries (9/3 – 9/8 2008).

Country	Visits
United States	10,523
United Kingdom	3,645
France	1,820
Spain	1,585
Canada	1,573
Germany	1,286
Italy	878
Brazil	849
Australia	848
Netherlands	680

⁵⁰ Measured with Google Analytics (www.google.com/analytics)

⁵¹ Google Code (code.google.com), Google Earth API (code.google.com/apis/earth) and the KML documentation page (code.google.com/apis/kml)

7. References

- Andrienko, N, Andrienko, G., Gatalsky, P., 2005, "Impact of Data and Task Characteristics on Design of Spatio-Temporal Data Visualization Tools", in Exploring Geovisualization, J. Dykes, A. M. MacEachren, M.-J. Kraak (editors), Elsevier Ltd.
- Atlas of Canada, 2008, "Colour Design and Tools", National Resources Canada (NRCan). Available online:
http://atlas.nrcan.gc.ca/site/english/learningresources/carto_corner/map_content_design_tools.html [Last accessed: 22 July 2008]
- Butler, H., 2006, "MySQL Spatial support for OGR", Hobu Inc., Available online:
<http://hobu.biz/2006/02/12/mysql-spatial-support-for-ogr-2> [Last accessed: 12 August 2008]
- Crampton, J. W., 2001, "Maps as social constructions: power, communication and visualization", *Progress in Human Geography*, 25(2), pp. 235-253
- Davis, S., 2007, "GIS for Web Developers. Adding *Where* to Your Web Applications", The Pragmatic Programmers.
- Gibin, M., Singleton, A., Milton, R., Mateos, P., Longley, P., 2008, "An Exploratory Cartographic Visualisation of London thorough the Google Maps API", *Applied Spatial Analysis and Policy* (2008) 1:85-97, Springer Netherlands
- Google, 2008a, "KML Reference", Google Code. Available online:
<http://code.google.com/apis/kml/documentation/kmlreference.html> [Last accessed: 13 June 2008]
- Google, 2008b, "KML 2.1 Tutorial", Google Code. Available online:
http://code.google.com/apis/kml/documentation/kml_21tutorial.html [Last accessed: 10 August 2008]
- Harrower, M., Fabrikant, S., 2008, "The Role of Map Animation for Geographic Visualization". Book chapter in Dodge, M., McDerby, M. and Turner, M., 2008, "Geographic Visualization: Concepts, Tools and Applications", Wiley
- Hudson-Smith, A., Crooks, A., 2008, "The Renaissance of Geographic information: Neogeography, Gaming and Second Life", Centre for Advanced Spatial Analysis (CASA), University College London. Available online: http://www.casa.ucl.ac.uk/working_papers/paper142.pdf [Last accessed: 6. August 2008]
- Kay, R., 2006, "Quick Study: LAMP", Computerworld. Available online:
<http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=263611> [Last accessed: 10 August 2008]
- Keen, A., 2007, "The Cult of the Amateur: How Today's Internet is Killing Our Culture", Doubleday Business.
- Kraak, M.-J., van de Vlag, D.E, 2007, "Understanding Spatiotemporal Patterns: Visual Ordering of Space and Time", *Cartographica*, Volume 42, Number 2, University of Toronto Press
- Krygier, J., Wood, D., 2005, "Making Maps: A Visual Guide to Map Design for GIS", The Guilford Press
- Lake, R., 2005, "GML and KML Syntax", Galdos Systems Inc. Available online:
<http://www.galdosinc.com/archives/183> [Last accessed: 8 August 2008]

- Loosley, C., 2006, "Rich Internet Applications: Design, Measurement, and Management Challenges", Keynote Systems. Available online: http://www.keynote.com/docs/whitepapers/RichInternet_5.pdf [Last accessed 4: August 2008]
- MySQL, 2008, "MySQL 5.0 Reference Manual", MySQL. Available online: <http://dev.mysql.com/doc/refman/5.0/en/spatial-extensions.html> [Last accessed: 12 August 2008]
- OGC, 2008, "OGC Approves KML as Open Standard", Geospatial Press Releases. Available online: <http://geospatialpr.com/2008/04/14/ogc-approves-kml-as-open-standard/> [Last accessed: 5 August 2008]
- OGC, 1999, "OpenGIS Simple Feature Specification For SQL", Revision 1.1, Open GIS Consortium Inc. Available online: http://portal.opengeospatial.org/files/?artifact_id=829 [Last accessed: 30 July 2008]
- Pomakis, K., 2007, "OpenGIS Tiled WMS Discussion Paper", OGC 07-057r2, version: 0.3.0, Open Geospatial Consortium Inc. Available online: http://portal.opengeospatial.org/files/?artifact_id=23206 [Last accessed: 4 August 2008]
- Rademacher, P., 2008, "Google Earth, meet the browser", Google Lat Long Blog. Available online: <http://google-latlong.blogspot.com/2008/05/google-earth-meet-browser.html> [Last accessed: 3 August 2008]
- Sandvik, B., 2008, "Research Paper: Using KML for Thematic Mapping", MSc GIS Dissertation, University of Edinburgh
- Slocum, T. A., McMaster, R. B., Kessler, F. C., Howard, H. H., 2005, "Thematic Cartography and Geographic Visualization", Second Edition, Person Education Inc.
- Turner, A. J., 2006, "Introduction to Neogeography", O'Reilly Short Cuts, O'Reilly Media Inc.
- Ullman, L., 2007, "PHP 5 Advanced. Extend your PHP skills the fast, efficient way", Peachpit Press
- UNdata, 2008, "UNdata - a data access system to UN databases", United Nations. Available online: <http://data.un.org/wiki/aboutus.ashx> [Last accessed: 8 August 2008]
- Walsh, J., 2008, "The Beginning an End of Neogeography", GEOconnexion International Magazine, April 2008. Available online: http://www.geoconnexion.com/uploads/neogeography_intv7i4.pdf [Last accessed: 8 August 2008]
- Wilson, T., 2008, "OGC KML 2.2.0", Document #07-147r2, Open Geospatial Consortium. Available online: <http://www.opengeospatial.org/standards/kml/> Last accessed: 5 May 2008

Appendix

Appendix A – World Borders Dataset	51
Appendix B – MySQL database tables	52
Appendix C - Thematic Mapping Engine API.....	53
C1: TME API Parameters	53
C2: Proportional symbol map	54

Appendix A – World Borders Dataset

Provided by Bjorn Sandvik, thematicmapping.org

Use this dataset with care, as several of the borders are disputed.

The original shapefile (world_borders.zip, 3.2 MB) was downloaded from the Mapping Hacks website: <http://www.mappinghacks.com/data/>

The dataset was derived by Schuyler Erle from public domain sources. Sean Gilles did some clean up and made some enhancements.

COLUMN	TYPE	DESCRIPTION
Shape	Polygon	Country/area border as polygon(s)
FIPS	String(2)	FIPS 10-4 Country Code
ISO2	String(2)	ISO 3166-1 Alpha-2 Country Code
ISO3	String(3)	ISO 3166-1 Alpha-3 Country Code
UN	Short Integer(3)	ISO 3166-1 Numeric-3 Country Code
NAME	String(50)	Name of country/area
AREA	Long Integer(7)	Land area, FAO Statistics (2002)
POP2005	Double(10,0)	Population, World Population Prospects 2005
REGION	Short Integer(3)	Macro geographical (continental region), UN Statistics
SUBREGION	Short Integer(3)	Geographical sub-region, UN Statistics
LON	FLOAT (7,3)	Longitude
LAT	FLOAT (6,3)	Latitude

CHANGELOG VERSION 0.1 - 13 March 2008

- Polygons representing each country were merged into one feature
- Åland Islands was extracted from Finland
- Hong Kong was extracted from China
- Holy See (Vatican City) was added
- Gaza Strip and West Bank was merged into "Occupied Palestinian Territory"
- Saint-Barthelemy was extracted from Netherlands Antilles
- Saint-Martin (French part) was extracted from Guadeloupe
- Svalbard and Jan Mayen was merged into "Svalbard and Jan Mayen Islands"
- Timor-Leste was extracted from Indonesia
- Juan De Nova Island was merged with "French Southern & Antarctic Land"
- Baker Island, Howland Island, Jarvis Island, Johnston Atoll, Midway Islands and Wake Island was merged into "United States Minor Outlying Islands"
- Glorioso Islands, Parcel Islands, Sparty Islands was removed (almost uninhabited and missing ISO-3166-1 code)
- Added ISO-3166-1 codes (alpha-2, alpha-3, numeric-3). Source: <https://www.cia.gov/library/publications/the-world-factbook/appendix/appendix-d.html>
- Added ISO-3166-1 codes (alpha-2, alpha-3, numeric-3). Source: <http://unstats.un.org/unsd/methods/m49/m49alpha.htm>
- Added ISO-3166-1 codes (alpha-2, alpha-3, numeric-3). Source: <http://www.fysh.org/~katie/development/geography.txt>
- AREA column has been replaced with data from UNdata: Land area, 1000 hectares, 2002, FAO Statistics
- POPULATION column (POP2005) has been replaced with data from UNdata: Population, 2005, Medium variant, World Population Prospects: The 2006 Revision
- Added region and sub-region codes from UN Statistics Division. Source: <http://unstats.un.org/unsd/methods/m49/m49regin.htm>
- Added LAT, LONG values for each country

Appendix B – MySQL database tables

Table: indicator

Field	Type	Index
id	smallint(6)	PRIMARY
name	text	
description	text	
source	text	
precision	tinyint(4)	

Table: indicator_values

Field	Type	Index
indicator	smallint(5)	
country	smallint(3)	PRIMARY
year	varchar(4)	
value	double	INDEX

Table: country

Field	Type	Index
OGR_FID	smallint(3)	PRIMARY
unicode	smallint(3)	UNIQUE
iso2	varchar(2)	
iso3	varchar(3)	
fips	varchar(2)	
name	varchar(50)	
border	geometry	INDEX
lon	double(8,3)	
lat	double(8,3)	
region	smallint(3)	
subregion	smallint(3)	

Table: geometry_columns

Field	Type	Index
F_TABLE_CATALOG	varchar(256)	
F_TABLE_SCHEMA	varchar(256)	
F_TABLE_NAME	varchar(256)	
F_GEOMETRY_COLUMN	varchar(256)	
COORD_DIMENSION	int(11)	
SRID	int(11)	
TYPE	varchar(256)	

Table: spatial_ref_system

Field	Type	Index
SRID	int(11)	
AUTH_NAME	varchar(256)	
AUTH_SRID	int(11)	
SRTEXT	varchar(256)	

Appendix C - Thematic Mapping Engine API

C1: TME API Parameters

This table includes all valid parameters for the TME API.

Parameter	Allowed values	Default value
barSize		50000
classification	unclassified equal quantile	unclassified
colour		FF6600
colourType	scale single	scale
endColour		FF6600
indicator		
mapDescription		From dataStore
mapSource		From dataStore
mapTitle		From dataStore
mapType	choropleth prism bar symbol	choropleth
maxHeight		2000000
noDataColour		CCCCCC
numClasses		5
opacity		90
showLegend	true / false	true
showNames	true / false	false
showTitle	true / false	true
showValues	true / false	false
startColour		FFFF99
symbolMaxSize		5
symbolShape	Varies	
symbolType	image polygon collada	
timeType	year series slider	year
year		

M = Mandatory parameter / O = Optional parameter

C2: Proportional symbol map

Required and optional parameters for proportional symbol maps.

Required parameters:

- indicator
- mapType (symbol)
- symbolType (image / polygon / collada)
- symbolShape (circle / square)
- symbolMaxSize
- year

Optional parameters:

- classification (unclassified / equal / quantile)
- colour
- colourType (scale / single)
- endColour
- mapDescription
- mapSource
- mapTitle
- numClasses (2-9)
- showLegend (true / false)
- showNames (true / false)
- showTitle (true / false)
- showValues (true / false)
- startColour
- opacity (0-100)
- timeType (year / series / slider)

Quantile classification is only valid when timeType = 'year'.

Example:

```
$dataStore = $dataConnector->getDataStore(12, 2005, 0);

$parameters = array( 'mapType'           => 'symbol',
                    'symbolType'        => 'image',
                    'symbolShape'        => 'circle',
                    'symbolMaxSize'      => 10,
                    'indicator'          => 12,
                    'year'               => 2005
                    );

$map = new ThematicMap($dataStore, $parameters);
$file = $map->getKML();
```