Usable and well scaled maps for consumers

Work Package 2

Theo Tijssen (Ed.)

Delft, March 2008

RGI 233 Report Nr 2

Usable and well scaled maps for consumers

Work Package 2

Theo Tijssen (Ed.)

Delft, March 2008

RGI 233 Report Nr 2

Summary

Today we see a huge increase of the use of geo-information in mobile devices. All current solutions are based on static copies that are stored on the mobile device. This makes dynamically adapting the map to new information and to the changing circumstances of the user impossible. With the availability of high bandwidth wireless connections (such as UMTS) better, more dynamic, solutions are possible: The server generates a proper, up-to-date map of the region of interest at the right level of detail for display and adjusted to the needs of the user. For a mass market (consumers of mobile maps) the human factors aspect is very important. The currently available mobile maps solutions still have insufficient user-interfaces. Extremely important is the issue of context as the user gets 'lost' very easily on the small mobile displays when zooming and panning. Based on a selection of use cases (navigation, tourist support, etc.), User-Centered Design techniques will be applied to develop small prototypes / simulations and the interaction and the quality of the maps in these prototypes / simulations will be evaluated.

The project runs from 2006 to 2008 and is organized in the following Work Packages:
- WP 1 'Preparation' (month 4-9, 2006);
- WP 2 'Prototype development' (month 10, 2006-6, 2007);
- WP 3 'Evaluation of first prototypes' (month 7-12, 2007);
- WP 4 'Improved prototypes' (month 1-6, 2008);
- WP 5 'Evaluation of improved prototypes' (month 7-12, 2008).

This report belongs to WP 2 ´Prototype development´ and consists of the following chapters:
1. Client application prepared by ESRI,
2. Decluttering and prevention of occlusion for map-based applications prepared by TNO.

Project leader	: Section GIS Technology OTB Research Institute for Housing, Urban and Mobility Studies Delft University of Technology
Address	: Jaffalaan 9 2628 BX Delft
Contact	: prof.dr.ir. P.J.M. van Oosterom
Tel.	: +31 (0)15 278 6950
Fax	: +31 (0)15 278 2745
E-mail	: p.j.m.vanoosterom@tudelft.nl
Website	: http://www.rgi-otb.nl/pages/uwms2

Contents

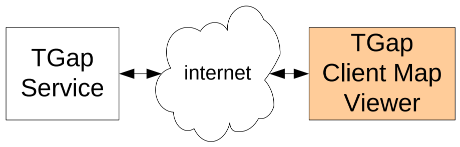
1 Client application

1.1 Introduction

This report describes the implementation of a simple map viewer which can visualize tGAP (topological Generalized Area Partitioning) data structures.

Only the client side implementation is described here. The tGAP data structures are created and maintained in a tGAP server environment. The server side technology for the tGAP is described in a separate report.

1.1.1 System context



- tGAP data structures are published on the server through a tGAP service (WFS/SOAP).
- The map viewer client application sends requests to the tGAP service and displays the results.

1.1.2 Original goals

- Service
  - The tGap data will be published through a WFS service.
- Dynamic map refresh
  - The objective was to implement a map viewer on a mobile platform that, dependent of the required map scale, would dynamically refresh itself while connected with the tGAP service.
- Animated rendering
  - The map view should render in an animated way. The mpa´s image should not be displayed at once but instead should be built up in a smooth way. Map elements should fade out/fadein gradually when the map image changes as a result of zooming or panning. Also map labels should smoothly fade in and fade out.
- Testdata
  - Several different datasets will be available for testing.
- Mobile platform
  - The map viewer must be deployed on a windows mobile device. For this reason the development environment chosen is the ESRI ArcGISServer Mobile Application Development Framework for .NET.

1.1.3 Accomplished goals

- Service
  - The ESRI client technology does not yet support WFS services. The support for WFS services is planned for the next major release of ArcGIS. TU Delft has therefore created a standard SOAP service which publishes the same information as the WFS service did. The geometries are packaged in GML format. At the client side logic has been implemented to interpret the soap response and unpack the GML geometries.
- Dynamic map refresh
  - The tGAP database was not fully prepared to receive map requests for an absolute map scale. Instead it was possible to request a map for a certain set of predefined scale levels. A scale level is

an integer value that does not correspond to a map scale. Valid scale levels range from 2 – 160. Since there was no logic to relate a certain map scale to a predefined scale level, the dynamic aspect of map viewing was not implemented.

- Animated rendering
  - All requirements regarding animated rendering can not be implemented by the ESRI technology at this moment. Therefore all these requirements have been left out and shifted forward to work package 4. It means that the current implementation of the map viewer renders its graphics in a standard way.

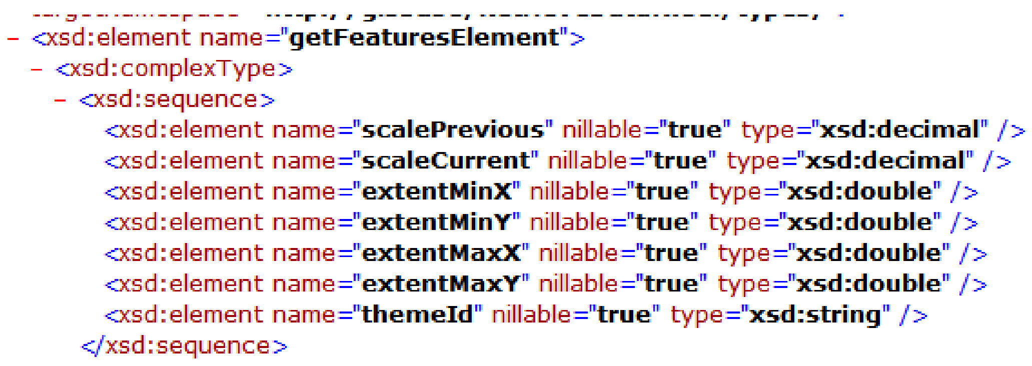
- Testdata
  - The available testdata in tGAP consists of a set of cadastral polygons. Not too much data but just good enough to demonstrate the concepts.

- Mobile device
  - Since there was no suitable mobile device available, the work package team decided that the map viewer could be deployed on a standard windows desktop platform. The program logic can easily be ported to a Windows Mobile 5 or 6 deployment platform. The user interface would have to be redesigned.

1.2 Service capabilities

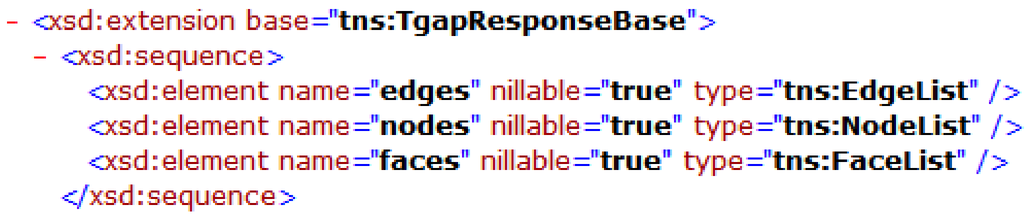
The following are pieces from the list of tGAP service capabilities and the data types that are used.

This web method is used to receive tGAP features from the service. The method is prepared to accept arguments scalePrevious, scaleCurrent, boundig box and themeID. The argument scaleCurrent is the only argument that is implemented in the service at this moment. The others are placeholders for future use.



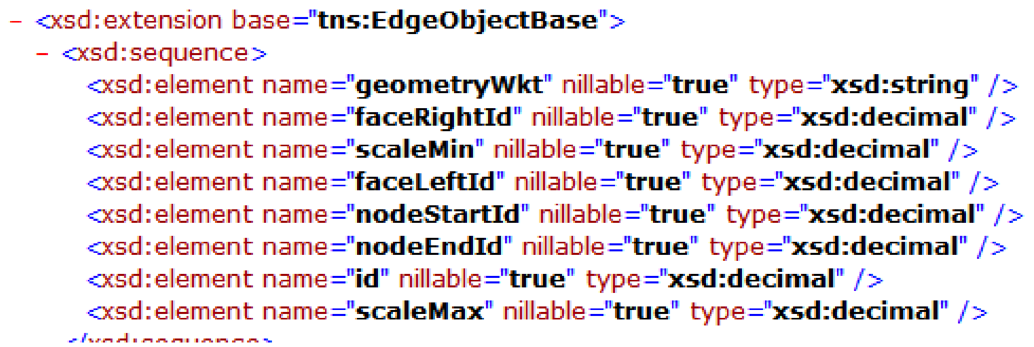
```
- <xsd:element name="getFeaturesElement">
  - <xsd:complexType>
    - <xsd:sequence>
        <xsd:element name="scalePrevious" nillable="true" type="xsd:decimal" />
        <xsd:element name="scaleCurrent" nillable="true" type="xsd:decimal" />
        <xsd:element name="extentMinX" nillable="true" type="xsd:double" />
        <xsd:element name="extentMinY" nillable="true" type="xsd:double" />
        <xsd:element name="extentMaxX" nillable="true" type="xsd:double" />
        <xsd:element name="extentMaxY" nillable="true" type="xsd:double" />
        <xsd:element name="themeId" nillable="true" type="xsd:string" />
      </xsd:sequence>
```

After a request is fired (in the form described above), the response from the service looks like this. It contains 3 lists. Edges, nodes and faces.



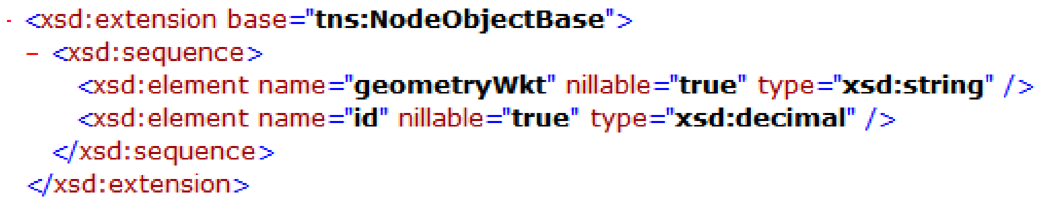
```
- <xsd:extension base="tns:TgapResponseBase">
  - <xsd:sequence>
      <xsd:element name="edges" nillable="true" type="tns:EdgeList" />
      <xsd:element name="nodes" nillable="true" type="tns:NodeList" />
      <xsd:element name="faces" nillable="true" type="tns:FaceList" />
    </xsd:sequence>
```

The EdgeList contains objects of type EdgeObjectBase. Each edge object has a geometry attribute. This geometry is used in the client application to construct the polygon geometry for the faces.



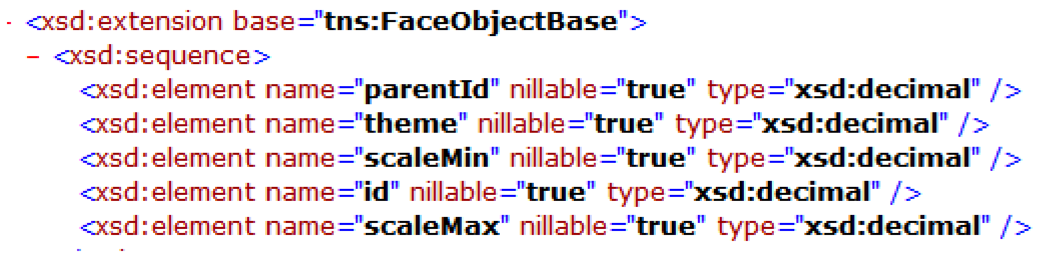
```
- <xsd:extension base="tns:EdgeObjectBase">
  - <xsd:sequence>
      <xsd:element name="geometryWkt" nillable="true" type="xsd:string" />
      <xsd:element name="faceRightId" nillable="true" type="xsd:decimal" />
      <xsd:element name="scaleMin" nillable="true" type="xsd:decimal" />
      <xsd:element name="faceLeftId" nillable="true" type="xsd:decimal" />
      <xsd:element name="nodeStartId" nillable="true" type="xsd:decimal" />
      <xsd:element name="nodeEndId" nillable="true" type="xsd:decimal" />
      <xsd:element name="id" nillable="true" type="xsd:decimal" />
      <xsd:element name="scaleMax" nillable="true" type="xsd:decimal" />
```

The NodeList contains objects of type NodeObjectBase. A node object also has a geometry attribute.



```
- <xsd:extension base="tns:NodeObjectBase">
  - <xsd:sequence>
      <xsd:element name="geometryWkt" nillable="true" type="xsd:string" />
      <xsd:element name="id" nillable="true" type="xsd:decimal" />
    </xsd:sequence>
  </xsd:extension>
```

The FaceList contains objects of type FaceObjectBase. These are only used for their id. The face geometry is constructed from the edges' geometries. For each edge we know to which face it belongs through the faceLeftID and faceRightID.



```
- <xsd:extension base="tns:FaceObjectBase">
  - <xsd:sequence>
      <xsd:element name="parentId" nillable="true" type="xsd:decimal" />
      <xsd:element name="theme" nillable="true" type="xsd:decimal" />
      <xsd:element name="scaleMin" nillable="true" type="xsd:decimal" />
      <xsd:element name="id" nillable="true" type="xsd:decimal" />
      <xsd:element name="scaleMax" nillable="true" type="xsd:decimal" />
```

The tGAP service can be accessed through this url:

http://casagrande.otb.tudelft.nl:8888/tGAP-Service-context-root/RetrieveDataSoapHttpPort

Conclusion:

- The tGAP response only contains information about nodes, edges and faces that is needed to reconstruct the faces, edges and nodes at the client side.
- The service response does not contain any other information or attributes for the nodes, edges and faces. So, there is no means to classify the resulting map or to label the map elements with attribute information apart from the id's of the objects.

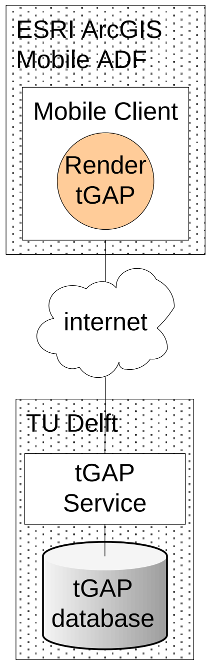
1.3 Client implementation

The main part of the implementation is the part in the client that fires requests to the tGAP service and receives responses from that service. That piece of logic is not standard available in the ESRI Mobile client environment. Therefore it has been designed and implemented from scratch.
Implementation in 2 steps.

The first step is to implement a simple mobile map viewer that is able to interpret the tGAP service response and render the results in a dedicated mobile viewer application. The goal of this step is to prove that the mobile technique from ESRI can use the tGAP service as a datasource.

The next step is to use the result from step 1 and implement it in a more generic solution such that not only a dedicated mobile map viewer can use the tGAP service as datasource but that the tGAP service can be used as a datasource for all sorts of clients.
The 2 steps are described in the following paragraphs Simple- and Generic-client solution.

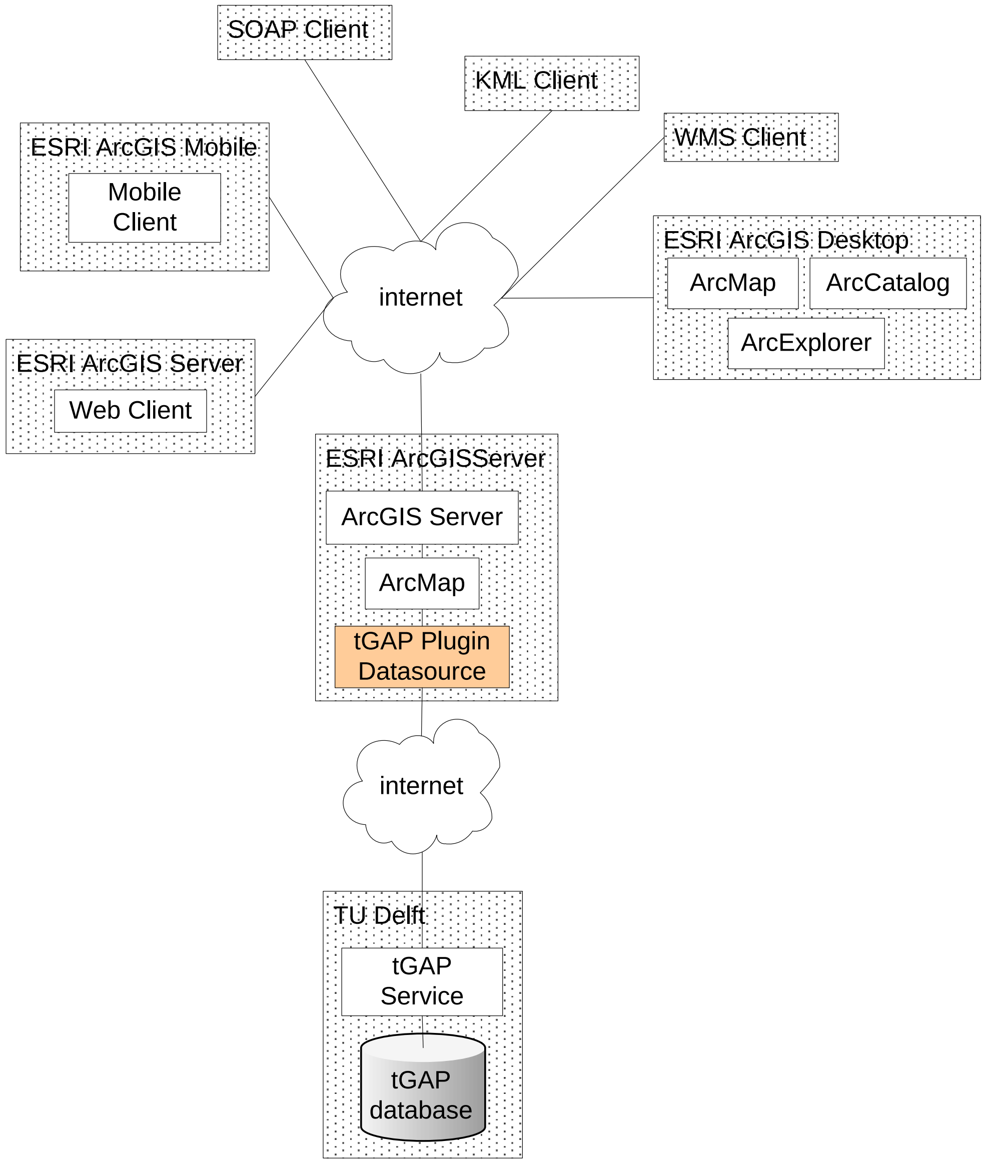
1.3.1 Simple client solution



Description of the simple client solution.

- The tGAP service at the server side is an Oracle Application Server service on top of a schema and packages in an Oracle database server.
- The architectural picture has been kept simple. In reality the Mobile Client needs a bit more context. It is part of a bigger framework within ArcGISServer.
- A dedicated function “Render tGAP” has been developed in the ArcGISServer Mobile Application Framework.
- The function does not (yet) work dynamically. That is, it does not get triggered by the zoom- or pan-events in the map viewer application. Instead it gets triggered by a menu button “Get data”.
- The menu button “Get Data” sends a request to the tGAP service. The only argument is scaleCurrent with one of the predefined values between 2 – 160. The response from the service is interpreted and the features are stored in the local map cache of the map viewer. The map control which does the actual rendering of the map data, takes its data from the local map cache and shows it in the map control.
- The features from the last service request are kept in the map cache. Each feature has attributes objectID (primary key) and scaleLevel. During rendering the features are filtered on scaleLevel such that only one scaleLevel gets drawn at the time. The result is an image that is representative for the scaleCurrent argument that was given to the tGAP service.
- The application also allows for a bulk request. The user can set a minimum- and a maximum-value for scaleCurrent and start the function “Get Data for ScaleRange” from the menu bar. The function will do multiple tGAP service requests in a row and store the results in the local map cache. The result is that the map cache will contain representations of tGAP features for all the scaleLevels that the user applied for. By sliding the trackBar at the left side of the user interface, the map viewer will “dynamically” display all the stored scaleLevels in the map.

1.3.2 Generic client solution



Description of the generic client solution.

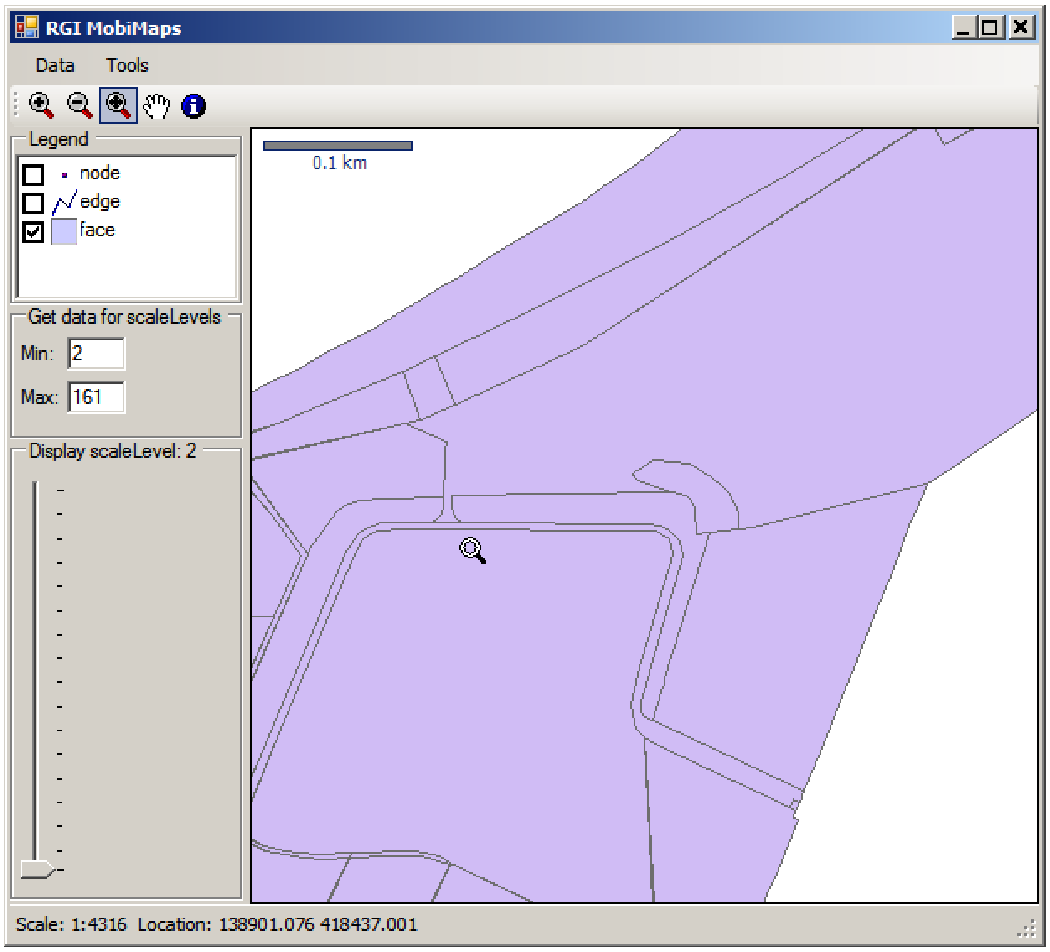
- In this solution the logic of the dedicated function “Render tGAP” from solution 1 is copied to a new component “tGAP Plugin Datasource”. Its functionality is almost the same. Only now it does not serve as a feature creation function in the mobile application but it acts as a standard ESRI dataprovider.
- The tGAP Plugin Datasource is built on top of a generic plugin datasource framework that is available in ArcGIS/ArcObjects. The plugin datasource recognizes when a tGAP service is selected as a source. Any ArcGIS product will now treat the tGAP datasource as a valid dataprovider.
- In the picture above, the plugin datasource acts as a dataprovider for ArcMap. Whenever a user zooms, pans or repaints ArcMap’s map control, the layers that are based on the tGAP plugin datasource will be refreshed from the tGAP service. In this case there is no buffering in a local map cache. That mechanism only works with the mobile client.
- The ArcMap document can be published as an ArcGISServer map service. Additionally it can also be published as a Mobile, KML and/or WMS service. As a result you can use many different client applications to display features from the tGAP service. One of them is a standard ArcGISServer Mobile

Application which gets its data from the mobile map service which will give you the same result as in solution 1.

1.4 Results from solution 1

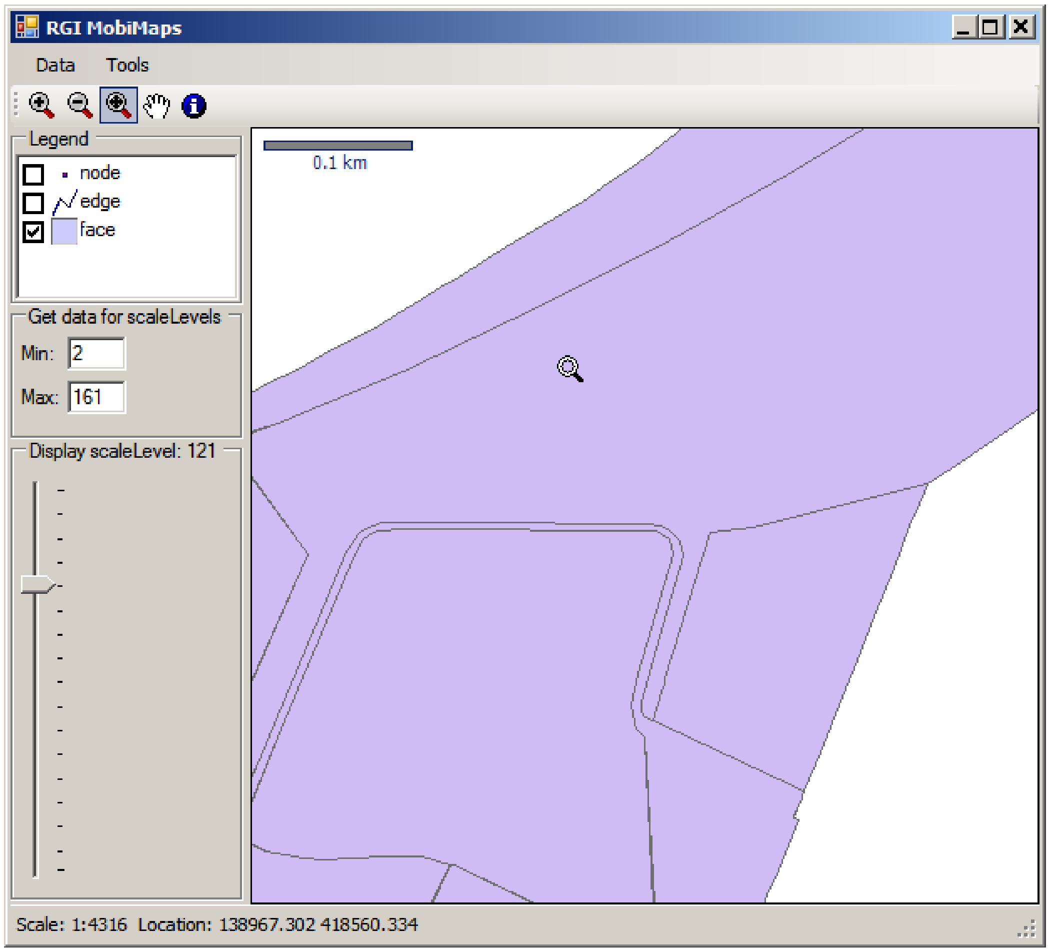
The test dataset in the tGAP database contains cadastral parcels. Therefore in the client application only the face layer is shown.

The map might show some wrong lines at certain scaleLevels. This is due to a known bug in the tGAP data structure.



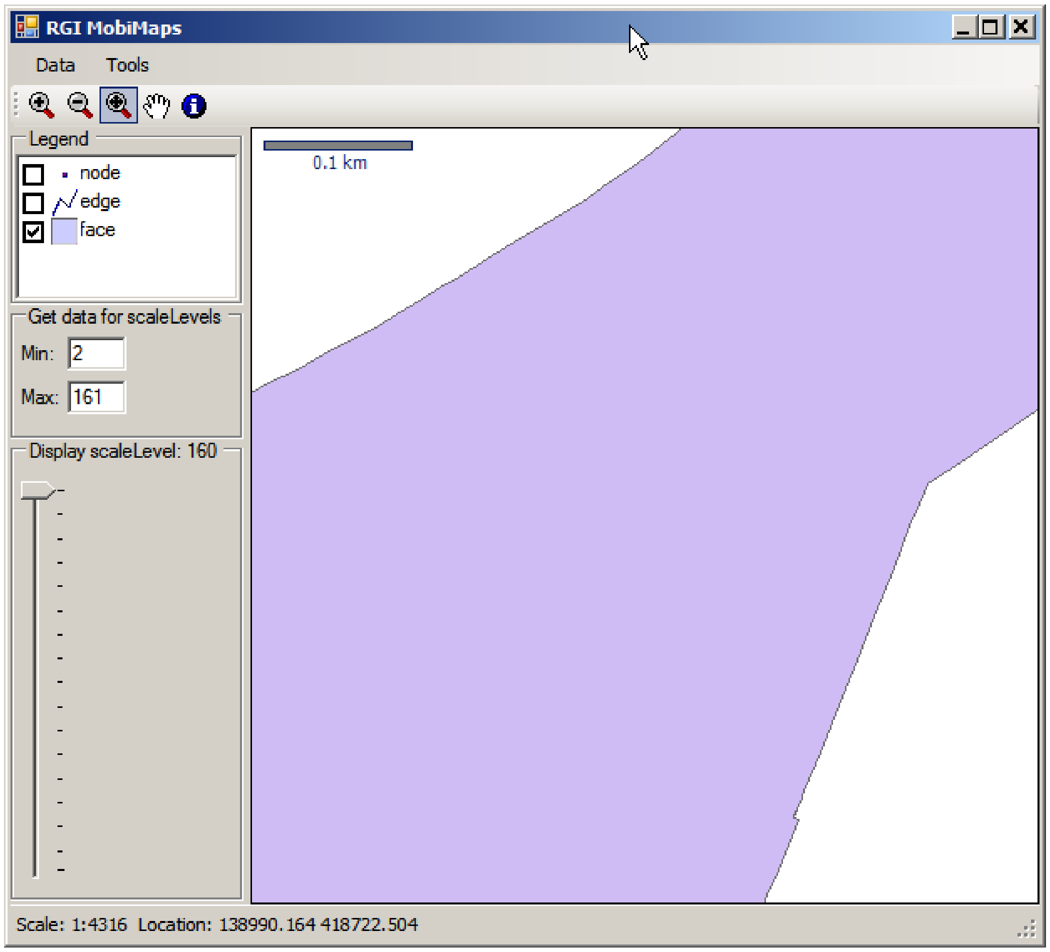
Get data for scaleLevels
Min: 2
Max: 161
Display scaleLevel: 2
Scale: 1:4316 Location: 138901.076 418437.001

The scaleLevel is 2. This is the most detailed level. All features and full geometry is shown.



Get data for scaleLevels
Min: 2
Max: 161
Display scaleLevel: 121
Scale: 1:4316 Location: 138967.302 418560.334

At scaleLevel 121 some of the tGAP features have disappeared. Some generalization has taken place.



At scaleLevel 160 all tGAP detail features have disappeared. Only the outer hull of the parcels is left over.The generalization was optimal.

The same scaleLevel displays, but now for the whole dataset.

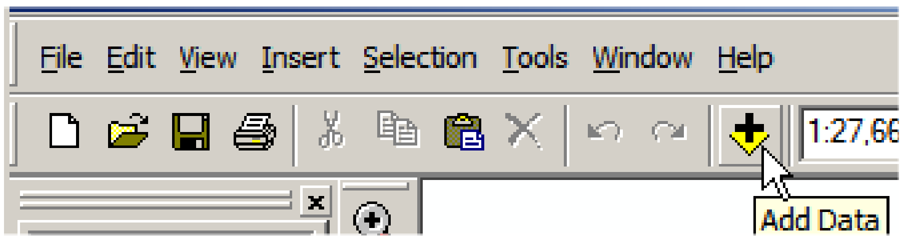
1.5 Results from solution 2

For the results of solution 2, the follwing clients will be shown here:

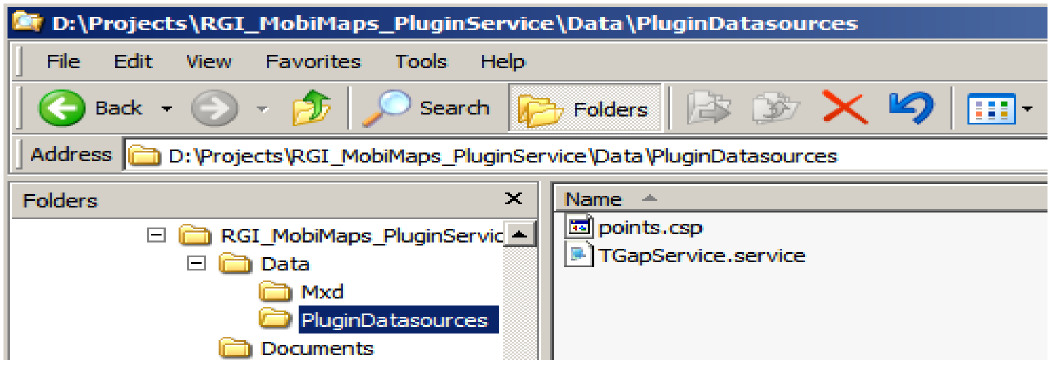
- ArcMap client;
- ArcGISServer Web Application.

1.5.1 ArcMap client

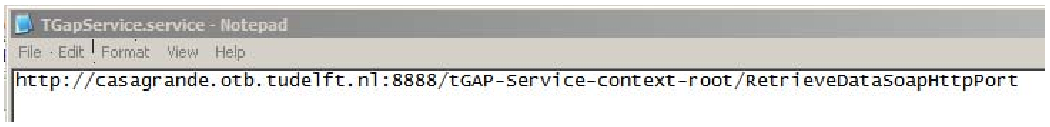
The following steps show how you can use the tGAP service as a plugin datasource in ArcMap.



In ArcMap, start the Add Data wizard.

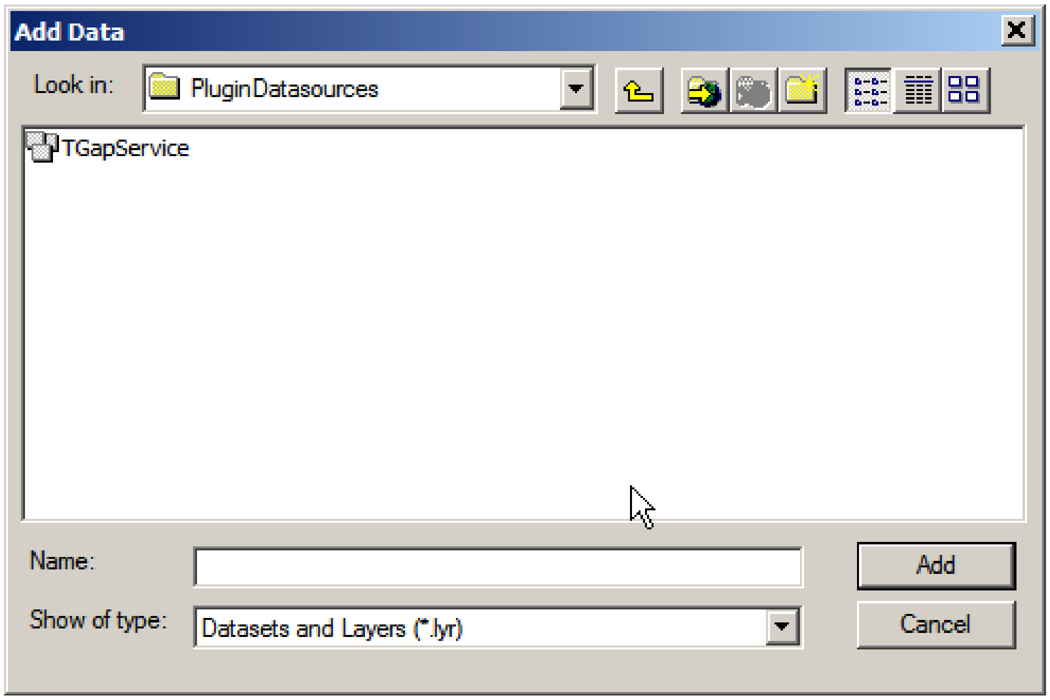

D:\Projects\RGI_MobiMaps_PluginService\Data\PluginDatasources
File Edit View Favorites Tools Help
Back
Search
Folders
Address D:\Projects\RGI_MobiMaps_PluginService\Data\PluginDatasources
Folders
RGI_MobiMaps_PluginServic
Data
Mxd
PluginDatasources
Documents
Name
points.csp
TGapService.service

In the Add Data wizard, browse to the directory where you keep your service connection file and select the file with a .service extension.

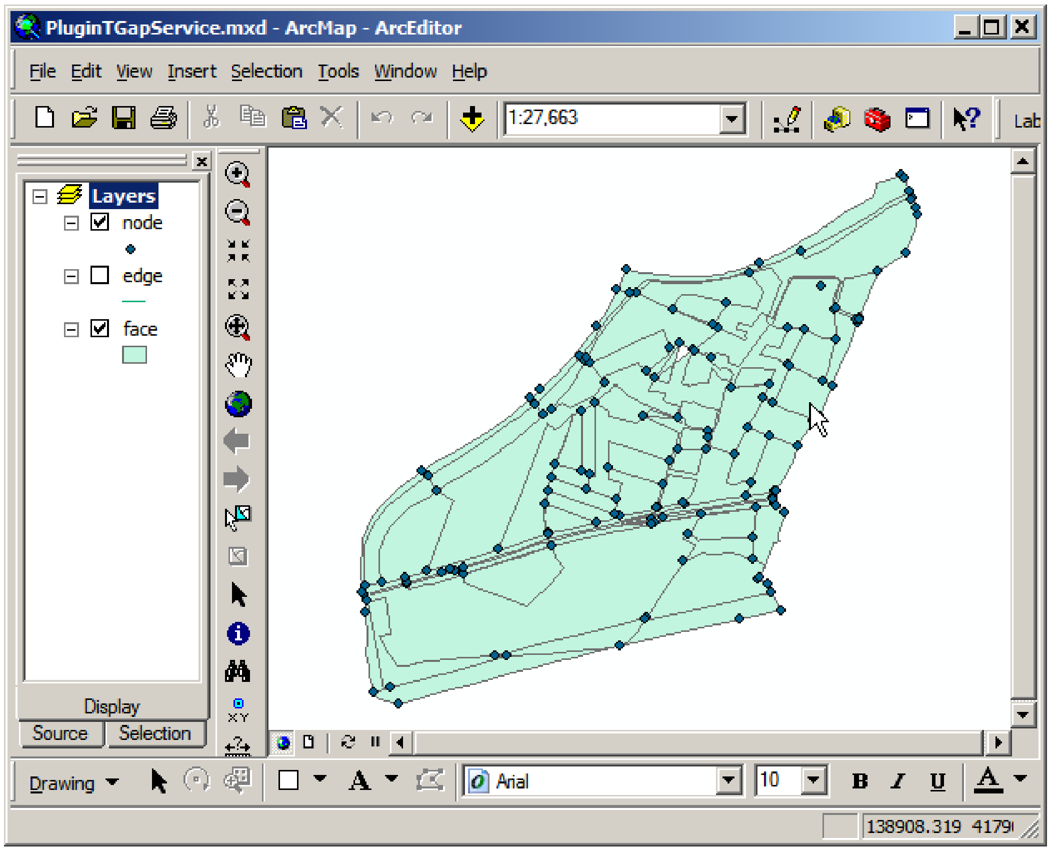


```
http://casagrande.otb.tudelft.nl:8888/tGAP-Service-context-root/RetrieveDataSoapHttpPort
```

The service connection file contains the url to the tGAP service.



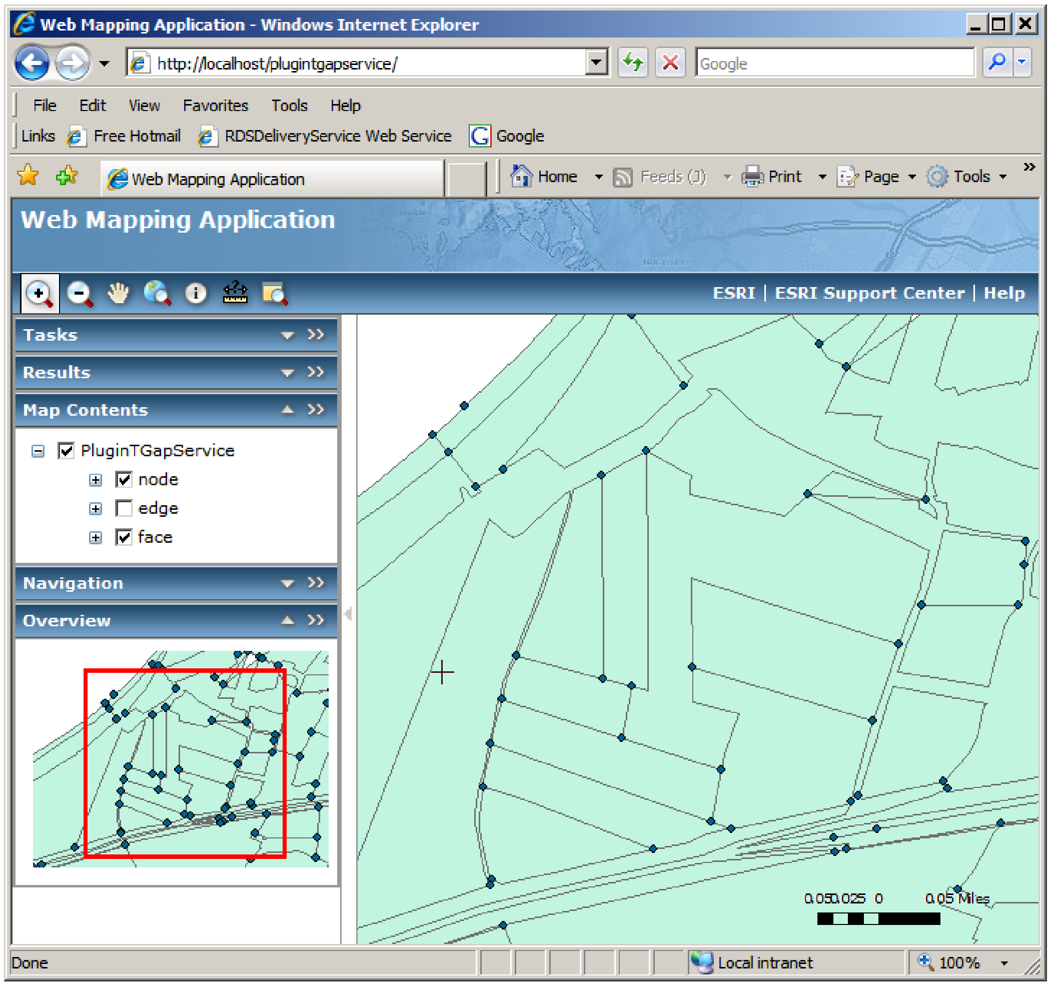
The Add Data wizard recognizes, through the tGAP plugin datasource, that the selected file contains a valid url to a tGAP service. Notice that the file is now displayed with the icon that represents a valid ESRI datasource. ArcMap opens the datasource and adds the 3 layers to the layer list.


PluginTGapService.mxd - ArcMap - ArcEditor
File Edit View Insert Selection Tools Window Help
1:27,663
Lab
Layers
node
edge
face
Display
Source
Selection
Drawing
Arial
10
B
I
U
138908.319 4179

The layers are rendered through the tGAP datasource. The result is the same as in solution 1. Only now you have all the rich functionality of ArcMap available.

1.5.2 ArcGISServer Web Mapping Application

After you have published the ArcMap document in ArcGISServer, you van simply create a web application through the web application wizard. The result is a web application that you can use from any internet browser.


Web Mapping Application - Windows Internet Explorer
http://localhost/plugintgapservice/
File Edit View Favorites Tools Help
Links Free Hotmail RDSDeliveryService Web Service Google
Web Mapping Application
Home Feeds (J) Print Page Tools
Web Mapping Application
ESRI | ESRI Support Center | Help
Tasks
Results
Map Contents
PluginTGapService
node
edge
face
Navigation
Overview
0.05 0.025 0 0.05 Miles
Done
Local intranet
100%

1.6 Development tools

For the development of the client applications, the following tools have been used:

- Development environment: Microsoft Visual Studio 2005
- Development language: C#
- Web development: Microsoft ASP.NET 2.0
- ESRI ArcGISServer plus Mobile Application Development Framework for the Microsoft .NET Framework

1.7 Conclusion

1.7.1 Activities for work package 4

Some of the initial goals that have not been met must be implemented in work package 4.

- WFS service
  - If ESRI supports WFS services within the ArcGISServer framework at the next release, we could switch back to the originally planned use of WFS service and abandon the own made SOAP service.

- Dynamic generalization
  - The tGAP service should implement a map scale value argument. The client could than request for a generalized result for the given map scale. Now the service only works with a so called predefined scale level index number.
- Performance

- In this prototype no attention was given to the performance of the prototype. In a production environment a complete refresh of a busy map could take a long time. At the client side an intelligent caching system should be implemented. This should avoid redundant service requests for data that is already in the cache.

- Mobile platform
  - The current prototype of the map viewer application is deployed on a Windows Forms platform. For the next work package it should also be available on a Windows Mobile 6 platform. If a physical device is not available, then just on a device emulator.

- Animated rendering
  - Although very difficult to do in an ESRI client we should still try to keep this topic on the calendar for the next work package. If not possible with standard ESRI technique then maybe simulate the result through other techniques (animate series of map images).

2 Decluttering and prevention of occlusion for map-based applications

2.1 Introduction

In recent years, popular map-based applications such as Google Earth and Yahoo Maps have become increasingly popular. Using these applications, consumers can find heaps of information about hotels, restaurants, shops, monuments, events, and etcetera. This supply of information often comes at the price of cluttered displays, in which it can be hard to find the wanted information.

In the last 20 years, work has been done on measuring clutter and on supporting visual search in cluttered displays. This study focuses on the problem of many closely positioned similar objects, or objects overlapping each other, sometimes called 'icon collisions'. Proposed solutions will be examined. At the moment, it is unclear which of the several existing methods will present the user the most usable visualization. It is not unlikely that which method is best depends on the task the user is conducting. An experiment is planned to developed knowledge on this subject.

In the next section a review of recent literature is presented on clutter and overlapping icons.

2.2 Defining and measuring clutter

According to the definition of Rosenholtz et al. (2007) clutter is the state in which excess items, or their representation or organization, lead to a degradation of performance at some task. This definition already shows that whether a display is cluttered depends on goals and tasks of the user. Excess and/or disorganized display items can cause crowding, masking, decreased recognition performance due to occlusion, greater difficulty at both segmenting a scene and performing visual search, and so on. Given a reliable measure of the visual clutter in a display, designers could optimize display clutter. Rosenholtz et al present and compare several measures of visual clutter, which operate on arbitrary images as input. The first is a new version of the Feature Congestion measure of visual clutter (Rosenholtz et al., 2005). This Feature Congestion measure of visual clutter is based on the analogy that the more cluttered a display or scene is, the more difficult it would be to add a new item that would reliably draw attention. A second measure of visual clutter, Subband Entropy, is based on the notion that clutter is related to the visual information in the display. The third measure, Edge Density (Mack and Oliva, 2004) is used as a measure of subjective visual complexity. The measure of clutter in a display was shown to be a good predictor of response times for a visual search task. All three measures can be used to replace set size, a variable often found to have high correlation with response times for visual search tasks in artificial settings, but hard to define in many real-world (especially photographic) images.

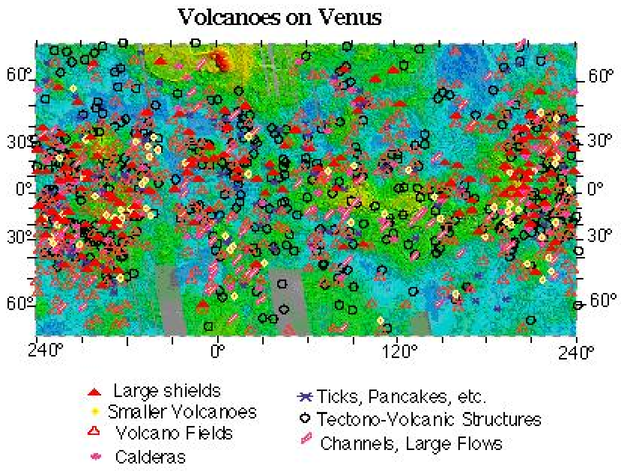


Figure 1: An example of a cluttered map.

The level of clutter is experienced very differently between people depending on their spatial abilities and expertise. A geologist will find contour lines on a map less cluttering then someone who is not used to them. In the same way some people prefer more information than others to build up a mental model of an area.

2.2.1 Dealing with clutter

Although clutter is very dependent on task requirements and user preferences, there are some general findings about how to reduce clutter for visual search tasks. One should focus on limiting the use of symbols that resemble each other in shape, size, or color (lines clutter lines, points clutter points). It is also important to avoid symbols that resemble letters in front of a word, for example an open round before a city name can increase search time significantly. A more advanced approach is generalization, defined as the process of reducing detail on a map as a consequence of reducing the map scale. Map generalization work focuses on changing representations of information at different map scales to reduce clutter and improve usability. Another approach to deal with clutter is using pop-out effects to distinguish different objects using features such as (Healey, 2005):

- color
- brightness
- animation (blinking, shaking, rotating, increasing/decreasing size)
- direction of illumination
- distinct curvature
- tilt
- outline emphasizing
- contrast enhancement
- opacity increasement
- focusing the target while blurring other objects
- enhancement of level of detail (LoD) against that of other objects

Lee, Forlizzi, and Hudson (2005) performed a study to order different pop-out effects. Pop-out is a bottom-up drawing of attention to an object, which occurs when an object within the visual field is distinctive along some visual dimension, for example, possessing a distinctive color or brightness when compared with other objects in the field. Prior studies have identified a range of visual features which can induce pop-out effects, including color, brightness, movement, direction of illumination, distinct curvature, and tilt. Notably however, size has not been shown to strongly induce this effect (Baldassi and Burr 2004). Other research has also shown that people are good in distinguishing colors. Phillips and Noyes (1982) coded 16 symbols either by color, texture, or a non-redundant combination of both. The colored symbols, with or without texture were much easier to find than symbols coded by texture alone. A problem with the use of colors is that changes in brightness of a display can have a disastrous effect on the distinguishability of the different colors. One has to keep in mind that pop-out effects have to be used sparingly, otherwise the effect is lost.

If it is known which information is not essential to task execution, this information may be removed from the display. This may be done automatically when the system uses a reliable task model, but more often it is the user's responsibility to turn-off unnecessary information layers. However, if for a task information is required from various domains, turning layers off may not be an option. Also, interaction with the system to select or deselect layers may come at a penalty in effort or time required (Yeh and Wickens, 2000).

Maps with high information density can suffer from overlapping icons, sometimes called 'icon stacking' or 'icon collisions'. Several solutions can be found in modern software applications, of which applying transparency is the most common. Alternatives are the grouping of several icons of the same type in a single aggregate icon, or replacement of the overlapping icons such that overlap is prevented. Several alternatives can be found in current applications. However, little research has been done on the usability of these alternatives. Fuchs and Schumann (2006) present an approach for intelligent icon placement and conflict resolution that can be used in real-time interactive systems, but not much work appears to be done in the scientific domain.

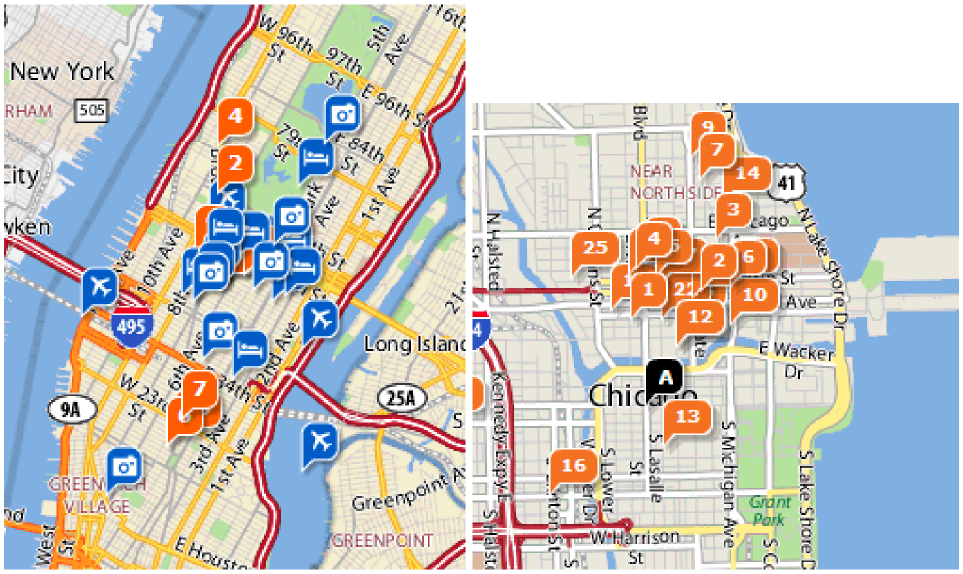


Figure 2: Yahoo Travel (left) and Google maps (right) produce cluttered maps with overlapping objects requiring much user interaction.

2.2.2 Experimental platform

A map-based software application has been developed in the.Net Compact framework using Visual Studio .Net 2005 to experiment with various declutter methods. In this framework applications for windows mobile 5 systems can be developed. The environment provides an emulator for debugging the program. The GUI is written in C#.

The map shows the center of a city, and three information layers can be selected: shops, parking, and restaurants. Furthermore there are three zoom levels that can be selected.

The user has to perform four different tasks using the application:
1. Search: Find restaurant X
2. Locate: In which street is shop Y
3. Identify: Find a restaurant of type Z
4. Navigate: You are in parking Q, and must plan the shortest circuit of three shops and a restaurant

The user is provided with a PDA as is shown in Figure 3, and can turn the three information layers on or off as desired.

The display can get very cluttered, especially when multiple information layers are required. Therefore, three different declutter methods were implemented.
1. Aggregate, combining several overlapping objects of the same type in a single large object.
2. Spread, shifting (partly) occluded objects until no longer overlap is present.
3. Spread-aggregate, combines both previous approaches. A large icon is placed nearby the overlapping objects, which are now represented with very small icons showing the exact location.

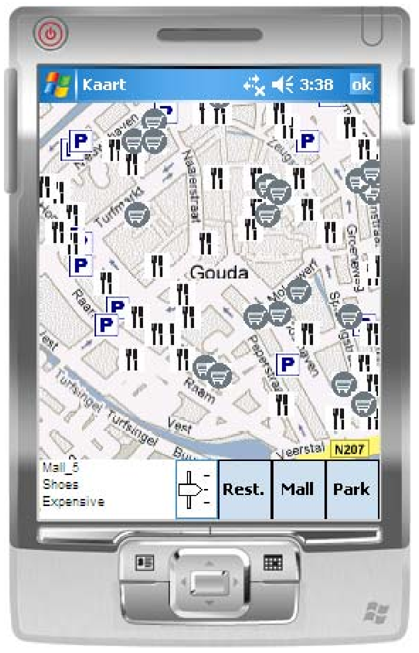


Figure 3: PDA provided to the users showing a cluttered display with overlapping objects.

These four conditions are shown in Figure 4, in which the top left part of display of Figure 3 is shown with the declutter alternatives.

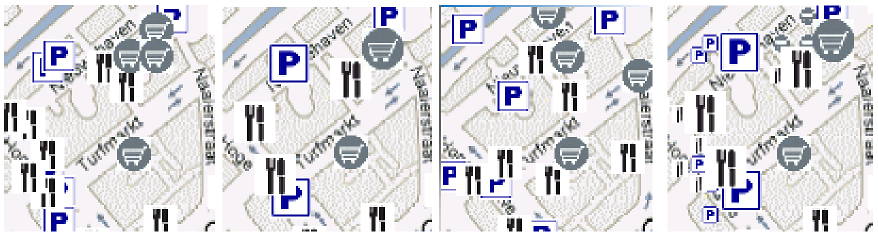


Figure 4: The four different conditions: 1) clutter, 2) aggregate, 3) spread, 4) aggregate-spread.

An experiment will be conducted in which users have to conduct these four tasks in four different conditions:

1. A cluttered display
2. A display decluttered using the aggregate algorithm
3. A display decluttered using the spread algorithm
4. A display decluttered using the spread-aggregate algorithm

2.2.3 References

Baldassi, S. and Burr, D. C. (2004). "Pop-out" of targets modulated in luminance or colour: the effect of intrinsic and extrinsic uncertainty." Vision Research 44: 1227-1233.

Fuchs, G. and Schumann, H. (2004). Intelligent Icon Positioning for Interactive Map-based Visualization Systems. 15th IRMA International Conference, New Orleans, USA

Healey, C.G. (2005). Perception in Visualization. Available online at: http://www.csc.ncsu.edu/faculty/healey/PP/ (Accessed November 2007).

Lee, J., Forlizzi, J. & Hudson, S.E. (2005). Studying the effectiveness of MOVE: A contextually optimized in-vehicle navigation system. Proceedings of the SIGCHI conference on Human factors in computing systems (CHI), 571-580. New York, NY: ACM Press.

Mack, M. L., and Oliva, A. (2004). Computational estimation of visual complexity. Paper presented at the 12th Annual Object, Perception, Attention, and Memory Conference. Minneapolis, Minnesota.

Phillips, R. J., and Noyes, L. (1982). An investigation of visual clutter in the topographic base of a geological map. Cartographic Journal, 19, 122–132.

Rosenholtz, R., Li, Y., Mansfield, J., & Jin, Z. (2005). Feature congestion, a measure of display clutter. SIGCHI (pp. 761–770). Portland, Oregon.

Yeh, M., and Wickens, C. D. (2000). Attention filtering in the design of electronic map displays: A comparison of color-coding, intensity coding, and decluttering techniques. (Technical Report ARL-00-4/FED-LAB-00-2). Savoy, IL: University of Illinois, Aviation Research Lab.

Corresponding address

Delft University of Technology
OTB Research Institute for Housing,
Urban and Mobility Studies
Jaffalaan 9, 2628 BX Delft
PO Box 5030, 2600 GA Delft
The Netherlands
tel. +31 (0)15 278 30 05
fax +31 (0)15 278 44 22
e-mail: mailbox@otb.tudelft.nl
www.otb.tudelft.nl