Visual Abstraction and Stylisation of Maps

Tobias Isenberg

Appeared in The Cartographic Journal 50(1):8–18, February 2013. doi> 10.1179/1743277412Y.0000000007

Visual Abstraction and Stylisation of Maps

Tobias Isenberg

University of Groningen, the Netherlands, and DIGITEO/CNRS/INRIA, France
isenberg@cs.rug.nl

Abstract—*We explore visual map abstraction for the generation of stylized renderings of 2D map data. We employ techniques that are centered around the concept of shape simplification and graph layout and that allow iterative abstraction of 2D maps. We use data from publicly available sources and show how we can iteratively generate aesthetic renditions of these maps. These renditions do not have the goal to allow for navigation tasks but instead show the map data in a distorted manner. The techniques used to create these images apply simplification, abstraction/generalization, and displacement operations to the map elements in varying orders and add stylistic shading to produce aesthetic renditions for print or electronic displays. The degree of abstraction/generalization can be individually chosen and determines the characteristics of the distorted map: whether components retain their shape, degenerate, or are processed in a manner that the abstraction becomes the focus of the image rather than the underlying map data. The renditions can be further personalized by choosing shading and colors for this shading. Together, the presented techniques allow for playful and creative exploration of aesthetic renditions of 2D map data.*

Keywords: Map stylization; generalization, abstraction, and simplification; illustrative rendering; non-photorealistic rendering (NPR).

1 Introduction

Cartography plays an important part in our daily lives (Field, 2005, 2009)—we use maps for many navigation tasks, both in abstract and in concrete domains. In particular geographic maps are frequently used, for example, in car navigation with a GPS device. Map creation has historically involved much artistry (Field, 2009)—from early cartographers who have created magnificent atlases to beautiful examples in modern interactive infographics. Even though some modern maps (e. g., in GPS car navigation) tend to focus on their task such as supporting navigation and only as a secondary aspect on their aesthetic character (Kent, 2005), artists have also explored maps in their works (e. g., (Wood, 2006; Varanka, 2006; Krygier, 2006; Caquard et al., 2009; Harmon, 2009; Ljungberg, 2009; Watson, 2009)); some art historians argue, for example, that Piet Mondrian, during the later part of his life, was at least in part inspired by maps or city street layouts to some of his well-known abstract works (Schoenholz Bee & Heliczer, 2004).

This study of abstraction of map data and its depiction, inspired by artworks and other sources, is the subject of this article. We are thus not interested in precise and correct depictions of the world but instead in altering maps to satisfy a specific personal aesthetic.[1] Traditionally, people have been using different types of abstraction for depicting map data, for instance to adapt the rendering to a chosen scale level or for highlighting particular aspects of the data. However, in these cases the depiction of map elements is typically driven by their original location,[2] even though some elements may be depicted at sizes different from their actual ones. In this article we take a different approach: we explore the incremental generalization of geographic maps in terms of the location, orientation, and detail of map elements using several independent approaches (e. g., Figure 1). This exploration is based on real data and is made possible through freely available quality map data from on-line sources such as OpenStreetMap (Coast, 2004). In addition, we describe a way to depict the abstracted data as inspired by the Substrate simulation by Tarbell (2003), a type of animated growth simulation (Figure 2). One could argue that the combination of both is a form of non-photorealistic and stylized map rendering,[3] incorporating aspects of generalization/abstraction as well as artistic depiction.

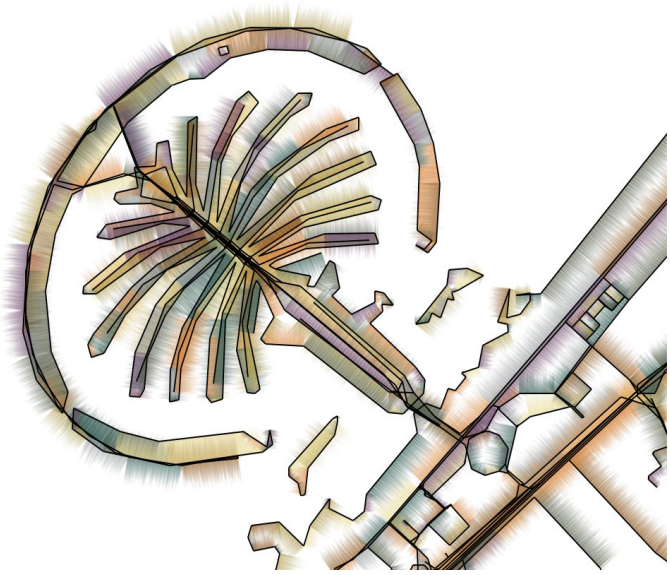


Figure 1. Visual map generalization/abstraction using the example of a map showing Dubai's Palm Jumeirah.

The application domain for this work is certainly not the precise and correct cartography (possibly with an aesthetic appeal) of a part of the world but instead the artistic and creative play with the abstraction of map data.[4] We see our contribution, instead, in the exploration of the creative interpretation of a map generalization process using visual map abstraction. We have implemented only some possibilities in this context and there are many variables to tune and adjust, but this freedom also brings with it the possibility to interactively explore these options. People may create, for example, aesthetic depictions of neighborhoods they are familiar with or places they would like to go. Potential other applications lie in generating imagery for electronic picture frames or personal print material such as increasingly popular photo books, for example about a family vacation.

The remainder of the article is structured as follows. First, we survey related work with respect to previous approaches to shape abstraction as well as some specific inspirations. Next, we introduce the em-

[1] The term "personal," here, does not refer to the article's authors but to the people working with the map generalization approach described in the article.

[2] An exception may be seen, e. g., in the heavily abstracting visualization of driving directions by Agrawala & Stolte (2001) or typical subway maps (see, e. g., the recent essay on subway map design by Roberts (2009)).

[3] It is not clear, however, what a photorealistic map rendering would look like (other than a satellite image) since maps inherently incorporate abstraction.

[4] We realize that the duality of art and science is being discussed for cartography (Krygier, 1995) but this article does not intend to answer this question.

ployed abstraction techniques and explain the rendering of the resulting data. Then, we give a number of details about the realization of the technique, show more results, and mention limitations. The article concludes with a summary and a discussion of potential future work.

2 Related Work

Research related to our own work can be found in several domains: in the computer graphics sub-domain of non-photorealistic rendering (NPR), generally in abstract styles of depiction (e. g., for visualization), in shape simplification for 3D shape representations, in gesture recognition, in information visualization, and in certain forms of computer art. We briefly discuss these connections in the following paragraphs.

In NPR we can find several approaches that exchange detail on small scales with image or geometry objects on larger scales to achieve abstraction. Examples are techniques within stroke-based rendering (Hertzmann, 2003) including, e. g., watercolor rendering (Curtis et al., 1997) or image mosaics (Hausner, 2001) where this exchange happens by sampling the original image on a different scale. Other examples include approaches that detect (groups of) detail objects and replace them with abstracted objects using dedicated processes (e. g., (Bousseau et al., 2006; Coconu et al., 2006; DeCarlo & Santella, 2002; Luft & Deussen, 2006; Mehra et al., 2009)).

The techniques discussed in this article are inspired by previous work in computer graphics that specifically addresses the abstraction, generalization, and simplification of shape. In particular it relates to the work by Mi et al. (2009) who describe an approach to abstract 2D shapes in terms of their parts. They, in fact, distinguish between earlier approaches to shape simplification on the one side which iteratively remove detail on a small scale and techniques for shape abstraction on the other which try to recognize parts on larger scales such as their method. Mi et al.'s shape abstraction technique is well suited, in particular, to map shapes as the authors demonstrate. Specifically, Mi et al. show how outlines of islands and continents (along with other objects) can be abstracted because these can be thought of as areal shapes with outlines so that relations such as 'part of' are defined. However, it is not clear how to apply them to general 2D map representations as used in our work which largely consist of a graph of connected line segments where a 'part of' relation cannot as easily be defined. Instead, the mesh simplification techniques to which Mi et al. compare their approach (e. g., (Ramer, 1972; Douglas & Peucker, 1973; Hoppe, 1996; Garland & Heckbert, 1997)) and similar techniques (e. g., (Visvalingam & Whyatt, 1993; Zhou & Jones, 2005)) are better suited in our context as they do not require information beyond connected edges in a graph and an error metric. Such algorithms have long been used for processing map data and many are implemented in GIS tools (e. g., MapShaper[5] (Bloch & Harrower, 2006; Harrower & Bloch, 2006)).

A related form of shape abstraction/generalization focusing on connected line segments can be found in gesture recognition. Here, the task is to match samples from usually hand-drawn stroke shapes to previously recorded or defined archetypical (i. e., abstracted) ones. Thus, the processing of gestures for recognition also requires a generalization step in order to be able to compare a newly performed gesture stroke to a known stroke. Noteworthy in this context is, e. g., the $1 Gesture Recognizer by Wobbrock et al. (2007) which provides a relatively simple way to recognize gestures in a sampling-, rotation-, size-, and location-invariant manner. It is based on resampling, rotating, and path-matching the input strokes to derive quality measures of the match for the collection of pre-recorded templates.

Abstraction and selective detail for maps related to our visual map abstraction has previously been explored in some techniques, often aimed at navigation tasks. For example, Agrawala & Stolte (2001) describe how to render effective route maps by generalizing different sections of route descriptions to different degrees, guided by cognitive psychology and common use cases of these maps. Related is the approach by Böttger et al. (2008a,b) who warp real geographic maps such that they can be overlaid on abstracted maps that show public transportation systems (e. g., metro maps). Another example is presented by Grabler et al. (2008) who show how to generate abstracted tourist maps that emphasize the most important elements. Related to these navigation-focused methods are also information visualization techniques that distort maps such that additional data attributes can be shown (Brinton, 1939, Chapter 29). For example, Panse et al. (2006) display geo-spatial point set data on maps that are distorted using a global shape function while Dorling et al. (2006) represent information that records data by territory on distorted maps on which the territories were resized accordingly. Approaches such as these abstract from the original spatial shape of maps to varying degrees but with the intention of conveying information rather than (primarily) aesthetics. Beautiful historical maps, however, often contain (intentional or unintentional) geographic distortions as analyzed by Jenny & Hurni (2011), and this geographic distortion can be an aspect of the map's aesthetics.

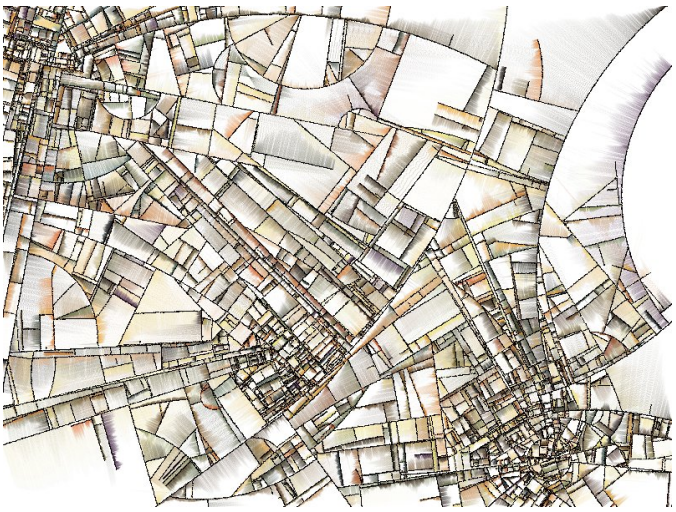


Figure 2. Screen capture of a Substrate simulation (Tarbell, 2003).[6]

Finally, the specific *visual* inspiration for this article largely comes from Tarbell's (2003) Substrate simulation (see an example in Figure 2).[6] This simulation spawns paths at various locations of the screen that are extended over the canvas as long as they do not hit an existing path. These paths are simple geometric primitives such as straight lines or parts of ellipsoids, and are complemented with watercolor-like shaded lines that radiate out perpendicular from the main paths. This results in depictions that have an appearance of abstracted cities and country-side between them, and led to the idea of replicating similar depictions but based on real map data.

3 Map Graph Abstraction/Generalization

Abstraction plays an important role in map making (Robinson et al., 1984). Cartographers employ many forms of generalization and simplification, both of the core spatial information as well as of the additional data shown on maps. For example, streets are typically shown wider than the map's scale requires and smaller features are omitted in larger-scale maps. For some mapping purposes (e. g., thematic maps) the cartographer may even introduce more simplification than technically necessary for the chosen scale. Most of these abstraction techniques are applied in cartography due to the technical limitations of scale and/or to promote understanding (Robinson et al., 1984).

In this work, however, we are interested in a visual abstraction or generalization of maps that goes beyond what is technically necessary for the creation of effective maps. Our approach is driven by (personal) aesthetics rather than a need for using maps for navigation tasks. For this purpose and based on map data available from OpenStreetMap (Coast, 2004), we describe a set of abstraction and simplification techniques that, in part, are inspired by the Substrate simulation as intro-

[5] See http://www.mapshaper.org/ .

[6] See http://www.complexification.net/gallery/machines/substrate/ , the simulation was also ported as a style to the Linux screensaver "XScreenSaver."

Figure 3. Map from OpenStreetMap (Coast, 2004) and filtered set of lines for the employed example map section (Schiermonnikoog island in the Netherlands).

duced above. These techniques can be applied independently of each other and in various sequences, each resulting in a different visual abstraction of the original map.

For these techniques we assume the map data to be available in form of an undirected graph of connected edges that, in our specific implementation, is extracted from the downloaded OpenStreetMap data, details of which are provided in Section 5. For the purpose of describing the technical details we use an example map of Schiermonnikoog, one of the Dutch islands in the North Sea shown in Figure 3. This particular map section was selected because it is relatively simple, has both low-detail and high-detail areas, and also contains a number of small and large loops of edges (e. g., the shore).

For achieving our goal of map abstraction we require techniques that are capable of generalizing graph data that consists of connected polylines, as opposed to polygons that have an inside and an outside such as used by Mi et al. (2009). As a second constraint we require the abstraction techniques to produce a sequence of simple local generalization steps that first remove small details before introducing larger changes. Thus, the sequence has to be ordered with respect to the degree of modification in order to be able to steadily increase the level of abstraction. Finally, we aim to achieve a general *visual abstraction* rather than an abstraction or generalization that satisfies qualitative or quantitative criteria with respect to understandability. That means that not only generalization techniques that reduce the number of map edges are suitable for our purpose but also those that instead appropriately change the location of these edges.

Below we describe a number of such techniques that we selected according to the mentioned criteria and which are mostly borrowed from other domains or which are relatively straightforward methods. As they are described in detail elsewhere we keep the description brief and focus on possibly necessary adaptations to the above requirements and on the application to the specific problem of map data abstraction.

Figure 4. Map from Figure 3 after simplification from 2086 edges to 452 edges using progressive meshes, with intersections and end nodes pinned to their original locations.

3.1 Progressive Mesh Technique

The first technique we examined—progressive meshes (Hoppe, 1996; Garland & Heckbert, 1997)—is borrowed from 3D surface mesh simplification. It works by computing an error metric that determines the potential error that would be introduced by collapsing a given edge in the mesh. In each step of the algorithm, it selects the edge whose deletion would introduce the smallest error and then collapses it. During an edge collapse, adjacent edges are affected, some other edges may also be removed, and the changed potential error for all changed edges is computed. By iterating this process one can compute an increasingly simplified version of the original mesh, until no further iterations are possible or the mesh starts to degenerate.

We apply the same technique to map graphs (similar to what Mi et al. (2009) did for the comparison of different shape generalization techniques), differing from the original progressive mesh technique (Hoppe, 1996; Garland & Heckbert, 1997) only in that our map 'meshes' may also contain vertices with a degree of < 3. The error metric we employ is the length of the edges so that shorter edges are collapsed first (other error metrics can easily be substituted). In addition, for computational reasons (one would have to re-sort the edge list in every step) we use the simplification that we only re-compute edge lengths and re-sort the edge list after N edge collapse operations. We start with $N = 40$ when the graph contains more than 1500 edges and reduce N in steps down to 1 for graphs with 300 edges or less.

The result of applying this technique to the example map is shown in Figure 4. We see that local features are removed while the resulting lines, in part, contain some roughness due to sharp corners. For creating Figure 4 we used additional constraints in form of pinning down vertices in the map graph such as ends of ways or vertices of degree > 2 (i. e., intersections). This prevents the algorithm from removing meaningful features such as detail within cities and towns.

3.2 Force-Directed Technique

Force-directed algorithms have long been used to achieve graph layouts that are aesthetically pleasing (Di Battista et al., 1999) by minimizing the 'energy' in a graph. Nodes are laid out by iteratively applying forces and, thus, moving the vertices in a graph based on their previous positions and possibly external constraints. In graph drawing, force-directed techniques are often employed to compute the layout for highly connected graphs, while our map graphs often have long chains of edges that represent boundaries or ways. In this context one could say that the application of a force-directed technique removes noise from the chains of edges, resulting in a more abstract representation that we are interested in.

Specifically, we start from a specific map graph layout and compute offsets for all nodes of the graph based on the location of their direct neighbors. To derive the offset for a node, we add up the the vectors to each of its direct neighbors, multiplied by a factor (we use 0.25). This results in a gentle smoothing of the map (see Figure 5) and a general

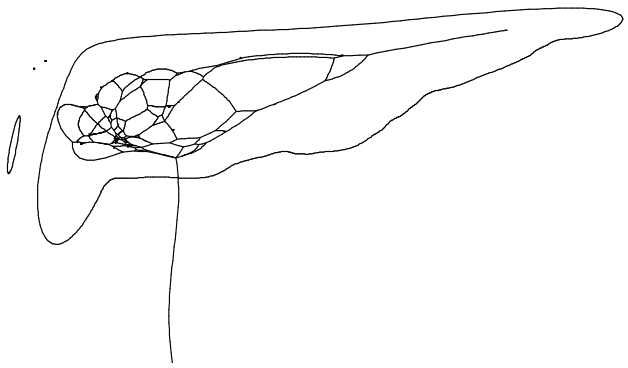


Figure 5. Map from Figure 3 after 134 iterations of the force-directed technique.

Figure 6. Map from Figure 3 (first simplified without restrictions using progressive meshes to 658 edges), after 555 iterations of the rectangularization technique.

contraction. To reduce the latter we can introduce similar constraints as done in Section 3.1 by pinning down certain vertices.

3.3 Orthogonalization Technique

Inspired by Piet Mondrian's abstract paintings with orthogonal lines, we also explored a special form of force-directed layout that aims to turn all edges toward that coordinate axis for which the angle between edge and coordinate axis is smallest (Figure 6). We achieve this by adding offsets to the vertices of each edge that turn it horizontally or vertically, rotating it around its center. While this would immediately 'orthogonalize' an independent edge, it is an iterative process in the case of a connected graph as connected edges influence each other. To simplify computation and due to this interaction of connected edges, we can employ a simpler computation for each edge: we simply use one half of the amount of the horizontal or vertical extent (whichever is smaller) of the edge as an offset (this can be computed as the dot product of the edge vector with a horizontal or vertical unit vector), directed to align the edge with the closer coordinate axis. While this approach turns non-connected edges horizontal or vertical in one step, it also introduces occasional oscillations for some edges with constrains from the connectivity in the graph. Therefore, we reduce the mentioned factor from 0.5 to 0.35 which leads to a slower convergence but also avoids oscillations.

The result of this technique is shown in Figure 6 (after an initial application of progressive meshes). An interesting effect is the behavior of long chains of small edges that are all close to horizontal (such as at the Northern coast of the island in the example) or vertical: these slowly propagate the horizontal or vertical property along the chain but do not converge to a fully horizontal or vertical line within a reasonable number of iterations (about 1 min of running).

```
ALGORITHM: Simplify using Ramer-Douglas-Peucker

INPUT:  array of nodes N, empty array of distances D
OUTPUT: array of nodes is updated with correct order

void simplifyRDP(nodes N, distances D) {
  // find index of node furthest from the baseline
  int  furthest  = findNodeFurthestFromBaseline(N);
  node furthestN = N[furthest];
  float distance =
    nodeDistanceFromBaseline(N, furthest);

  // recursively simplify sections before and after
  nodes     N1 = N.subArray(0, furthest-1);
  distances D1 = D.subArray(0, furthest-1);
  simplifyRDP(N1, D1);
  nodes     N2 = N.subArray(furthest+1, N.size()-1);
  distances D2 = D.subArray(furthest+1, N.size()-1);
  simplifyRDP(N2, D2);

  // merge simplification steps from the two parts
  // before adding the simplification step of this
  // level of the recursion
  int i1 = 0, i2 = 0, i = 0;
  while ( (i1 < N1.size()) || (i2 < N2.size()) ) {
    // in case we already copied all of D1 & N1
    // copy rest of D2 & N2
    if (i1 >= N1.size()) {
      D[i] = D2[i2];  N[i] = N2[i2];  i++;  i2++;
    }

    // or in case we already copied all of D2 & N2
    // copy rest of D1 & N1
    else if (i2 >= N2.size()) {
      D[i] = D1[i1];  N[i] = N1[i1];  i++;  i1++;
    }

    // otherwise merge normally
    else {
      if (D1[i1] < D2[i2]) {
        D[i] = D1[i1];  N[i] = N1[i1];  i++;  i1++;
      }
      else {
        D[i] = D2[i1];  N[i] = N2[i2];  i++;  i2++;
      }
    }
  }

  // finally add the simplification step of this
  // recursion level at the end of the sequence
  D[i] = distance;
  N[i] = furthestN;

  // N & D are now updated to the correct sequence
}
```

Listing 1. Pseudocode for reversing the output of the Ramer-Douglas-Peucker polyline simplification into a sequence from detailed to abstract.

3.4 Ramer-Douglas-Peucker Technique

Another goal for the abstraction of map graphs could be to attempt to replace long chains of edges (ways) with simpler representations of these. A way consists of a polyline in the graph for which all connector nodes have a degree of two; thus ways connect nodes of degree one (ends) or more than two (intersections). For this purpose we employ the Ramer-Douglas-Peucker algorithm (Ramer, 1972; Douglas & Peucker, 1973) which, in its original form, computes polyline simplification backwards: the technique starts with the most abstract form of a polyline—one edge that connects the first and the last point. It then proceeds to add detail by searching for the vertex farthest from the connecting edge and adding a new vertex at this location in the middle of the initial edge. The algorithm proceeds by recursively treating the two newly created smaller edges.

Figure 7. Map from Figure 3 after simplification with the Ramer-Douglas-Peucker technique to 452 edges.

Figure 8. Map from Figure 3 after simplification with the Visvalingam-Whyatt technique to 452 edges.

There are three issues with this original approach: (1) it computes the abstraction backwards (from abstract to detailed rather than from detailed to abstract as we need it), (2) it also does not provide a linear order of the abstraction steps due to its hierarchical design, and (3) it computes the abstraction by individual polyline/way and not for the entire maps as we need it. Thus, we adapt the original technique to address these issues as follows (see the pseudocode in Listing 1).

First, we compute the Ramer-Douglas-Peucker abstraction hierarchy as in the original algorithm. For each stepping out of a recursion, however, we adjust the technique by merging the abstraction steps (i. e., point collapse operations) from both recursions. This merging is done such that we always take the point collapse operation with a lower error (i. e., point distance from abstract edge) first but that the order within each series of abstraction steps is maintained. This ensures that, for any set of possible point collapse operations, the algorithm chooses the one that introduces the lowest error. After each merging, we add the point collapse operation between the two parts as the last operation to the list. This makes sure that the order of abstraction is reversed to 'from detailed to abstract.' Finally, for each newly computed list of point collapse operations (one for each way) we merge this with the already existing list from previously treated polylines the same way as the merging occurs in the recursion, again ensuring that for all possible next point collapse operations the one with the least error is chosen.

The result of applying this technique for the example map is shown in Figure 7. Similarly to the progressive mesh technique it abstracts the map graph resulting in a more polygonal shape, but does preserve some local features over more iterations (note the differences between Figures 4 and 7; both abstracted maps use the same number of edges).

3.5 Visvalingam-Whyatt Technique

As a final polyline generalization technique we examined Visvalingam-Whyatt algorithm (Visvalingam & Whyatt, 1993), being another method frequently used in map generalization (Bloch & Harrower, 2006). It examines the potential areal displacement that would be introduced by removing a node from one of the polylines and then picks that node to collapse that introduces the lowest areal displacement. Figure 8 shows a result of the application of this technique.

4 Rendering of the Abstraction/Generalization

For rendering the abstracted maps we take inspiration from Tarbell's (2003) previously mentioned Substrate simulation (Figure 2). It employs densely drawn colored lines, placed perpendicular to the main (black) lines on one of its sides. These lines have a length that varies randomly and they fade out toward the end. This creates an effect that is reminiscent of watercolor painting and we strive to render our abstracted maps in a similar way.

For this purpose we use each line in the map graph and turn it by 90° (see the schematic illustration in Figure 9). We compute a scaling factor that comprises a constant part (we use 40% of the original edge length) and a random part (up to 30% of the original edge length)

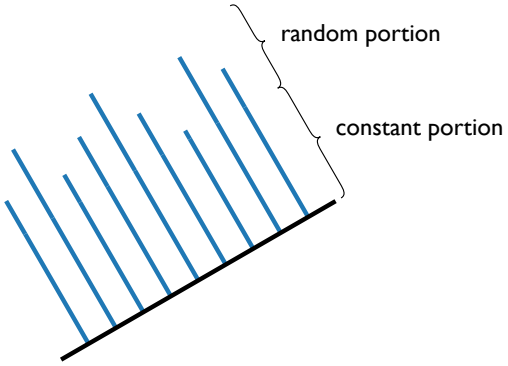


Figure 9. Schematic depiction of how the colored lines (blue) are created with respect to the base line from the map graph (black). In practice the colored lines are placed much denser and fade out toward their ends to create the appearance of colored patches.

and scale the turned line with it. We then draw colored lines with the computed length emerging perpendicularly from the graph edge where each starting point is separated from the next by one pixel. This general approach has the effect that for longer edges in the original map graph also wider colored patches are created. While the chosen percentages (colored lines being shorter than the original edges by 30–60%) typically limit the overlap of color patches and other lines, sometimes the overlap is too large. This happens, in particular, when the distribution of edge lengths in the map graph is uneven. Thus we introduce a user-specified maximum length for the colored edges.

The colors are randomly chosen from a palette with one color for all lines originating from a map graph edge. In addition, the colored lines have a given transparency at their start (we use approx. 60%) which is linearly increased to 100% at their ends, resulting in a fading out and visual shortening effect. Inspired by the colors chosen for the Substrate simulation (which in turn were inspired by colors used by Jackson Pollock (Tarbell, 2003)) we use a similar color palette (e. g., Figures 1, 10, 13 and 15). Alternatively, we explored color palettes from ColorBrewer (Harrower & Brewer, 2003; Brewer, 2009) (e. g., Figure 11), the use of one single color for all colored edges (e. g., Figure 12), or a palette with randomly chosen colors.

5 Realization

The implementation of the discussed techniques was done in Java and Processing. The Processing library[7] was chosen for its support of various rendering engines and export facilities while Java is used mostly for providing the UI elements to control the application of the algorithms. To internally represent the map graph data we employ the

[7] An extension of Java, see http://processing.org/ .

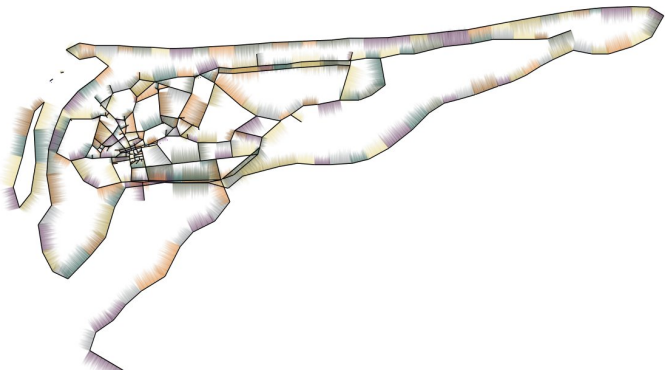


Figure 10. Abstracted map from Figure 4 with Substrate-inspired colors.

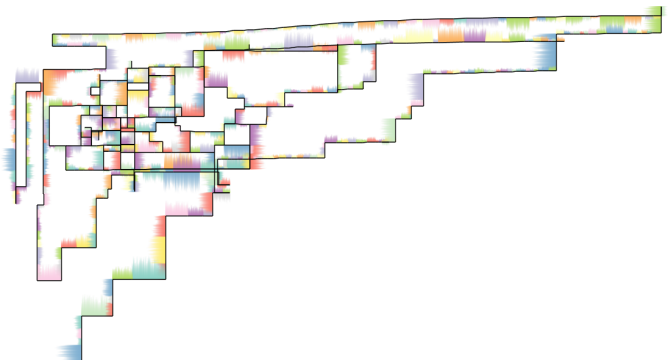


Figure 11. Abstracted map from Figure 6 with pastel color map from ColorBrewer (Harrower & Brewer, 2003; Brewer, 2009). Notice the different widths of colored patches due to the different lengths of the map graph edges.

Figure 12. Abstracted map from Figure 7 with a single color and limiting the length of color edges to approx. 30 pixels.

Jung library.[8] The resulting images are either exported to PDF (for lines only) or to PNG (for color images) through Processing's export facilities. The reason that color images are not exported to a vector format is that an interpolation of transparency along a line is not supported by Processing's PDF or SVG output.

The data is derived from OpenStreetMap (Coast, 2004) through their data export module which allows to extract a map section and save it as an XML file. These files are read in using Processing's XML import module or obtained directly through an on-line query using OpenStreetMap's Web API, and are stored in the Jung data structure. We filter this data so that we can concentrate on the most relevant map features. For example, single vertices (such as points of interest) and certain line features (e. g., administrative boundaries, city limits, etc.) can be removed from the data to be visualized.

[8]Jung stands for the Java Universal Network/Graph Framework; see http://jung.sourceforge.net/ .

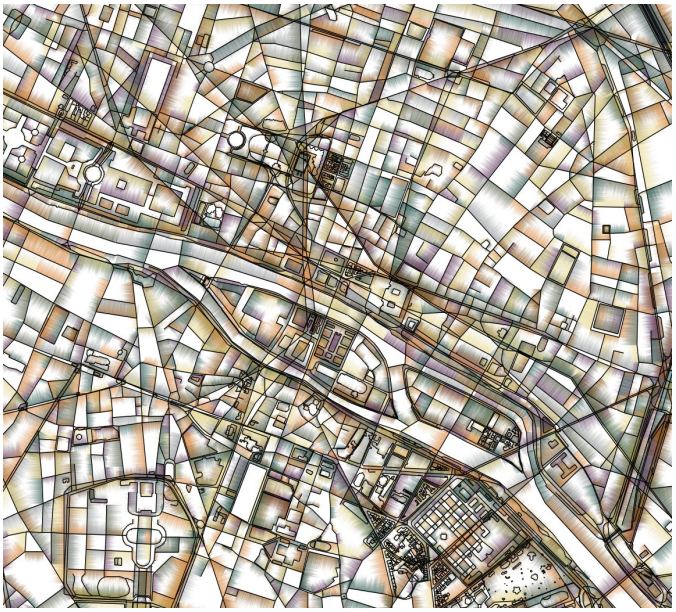


Figure 13. The center of Paris with Île de la Cité, without any simplification/abstraction applied to the map graph.

As some operations can be quite time-consuming, we employ several measures to speed up the processing. This includes the previously mentioned re-sorting only after a number of steps which was done for the progressive mesh technique. In addition, we also make sure that we only compute data when necessary. For example, we compute the simplification sequence for the Ramer-Douglas-Peucker technique only once for subsequent abstraction steps and only invalidate it when a different technique is applied. The result is that all techniques can be applied interactively and we even have to introduce wait cycles because otherwise the simplification of graphs with only few meshes happens too fast to be able to stop it when desired. For the example map (2086 initial edges) we compute 69 force-directed iterations and 97 orthogonalization iterations per second, while in the first second of computing we perform approx. 304 and 650 iterations for the progressive mesh and the Ramer-Douglas-Peucker technique, respectively (the last two values change over time as the graph is simplified). All measurements were made using full rendering on a 3 GHz Intel® Core™2 Extreme CPU running Windows Vista. Larger maps (i. e., more vertices) such as shown in Figures 14–15 may lead to fewer iterations per second.

6 Results

Both the map graph simplification/generalization and the choice of rendering technique add to the abstraction of map data. The rendering of lines within the map as simple black lines removes much of the information such as type or importance of its associated map element. In addition, the shading emphasizes the polygonal nature of the elements. While this works well together with the map graph simplification techniques it is not even always necessary to apply them as shown in Figure 13. Here, a part of Paris is shown with only the rendering applied and without any additional map graph simplification.

The simplification and abstraction techniques for the map graph seem to work best when there are fewer nodes with high degree. For example, in city centers (e. g., Manhattan in Figure 14) there are many street intersections connected by straight streets (i. e., only one edge between two intersections). In this case it is more difficult for both the simplification techniques (progressive meshes and Ramer-Douglas-Peucker) and the force-directed techniques to introduce changes that have an abstracting or generalizing effect without degenerating the

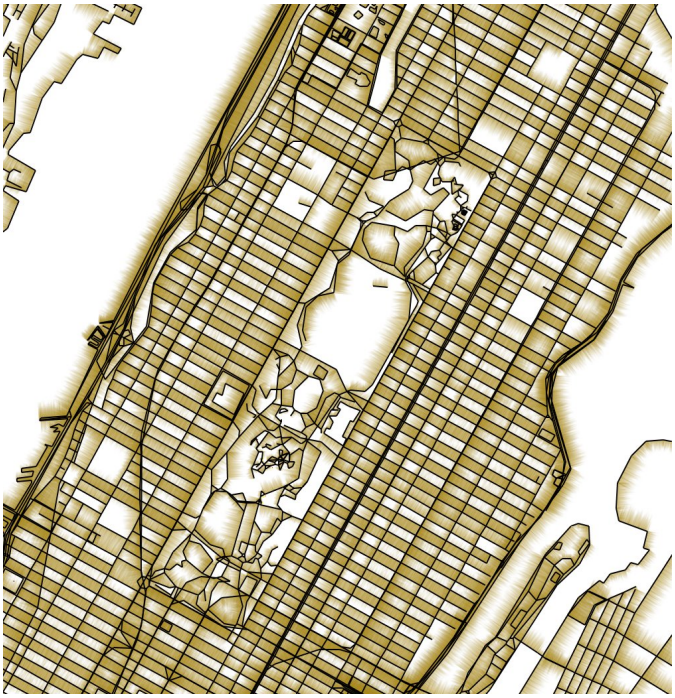


Figure 14. Manhattan with Central Park, simplified with Ramer-Douglas-Peucker to about ½ of the original edge count.

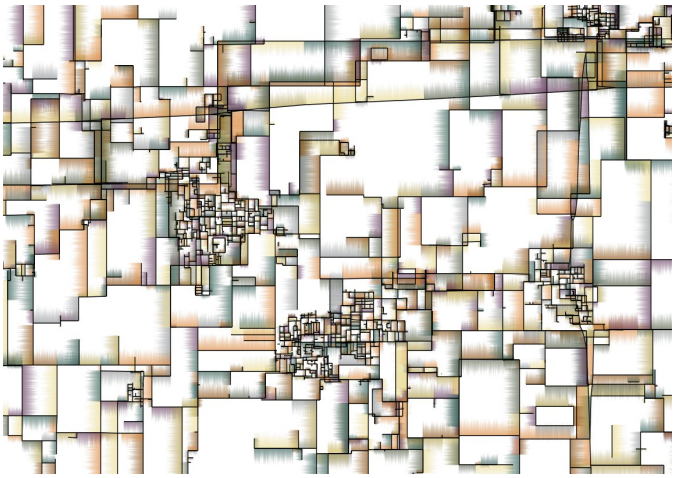
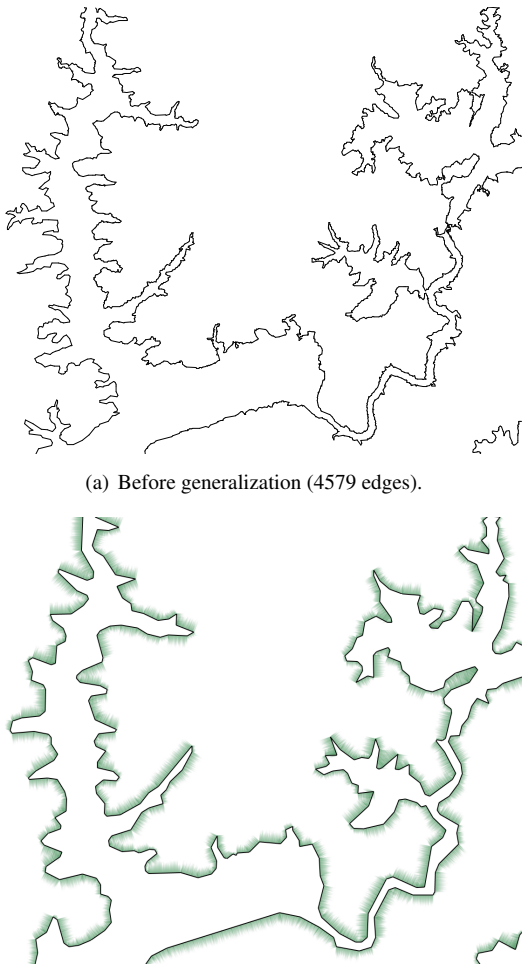


Figure 15. A rural area in Europe, after processing with the Ramer-Douglas-Peucker technique and orthogonalization, without pinning any vertices.

map graph. In contrast, in regions with longer edge chains and, thus, fewer intersections abstracting changes are easier to introduce (e. g., Central Park in Manhattan in Figure 14, rural areas such as the example from Europe in Figure 15, the river banks shown in Figure 16, or the man-made island "Palm Jumeirah" simplified using the Ramer-Douglas-Peucker technique in Figure 1).

The abstractions that result from the proposed techniques (when color patches are added as described) are reminiscent, to some degree, of artworks from the Cubism movement as well as NPR techniques inspired by Cubism such as the one presented by Collomosse & Hall (2003). This is due to the reduction to simple polygonal shapes which are emphasized by the color patches, reminiscent of the basic geometric shapes that are an important aspect in Cubist artworks.



(a) Before generalization (4579 edges).

(b) After generalization, with shading (395 edges).

Figure 16. A river section in Asia abstracted with the force-directed technique and Ramer-Douglas-Peucker.

7 Limitations and Possible Extensions

There are a number of limitations of the current implementation that should be noted. Some of these limitations arise from the treatment of the map as a simple graph of connected polylines and it being derived from OpenStreetMap. For example, it is currently not possible to consistently apply shading only to, for example, the land side of coastlines, lakes, or islands. While it would be possible to derive the information necessary for this behavior for closed polygons based on the type of the element, it is not guaranteed that map elements such as islands are represented as a single, closed polygon. Therefore, the shading is currently being placed always on the same side of the sequence of vertices in the data. An alternative, of course, would be to apply colored shading to both sides of a polyline. We did not choose to do so in order to retain the Substrate aesthetics and because shading on both sides would reduce the contrast of the depicted elements.

Another problem arises from the labeling of elements with multiple attributes—not all map elements work well in our approach (for example, we are currently omitting elements such as political boundaries). However, some polylines in the data are used for multiple purposes, which sometimes results in complex filtering configurations being necessary—this has not yet been implemented to a satisfying degree. Moreover, the use of OpenStreetMap's Web API for loading on-line data imposes restrictions: the API only allows the loading of maps that cover a relatively small area. However, even if larger areas could be loaded, the sheer amount of data would likely prohibit its processing within a reasonable time. Therefore, for being able to

explore maps such as the boundaries of countries or the shape of the continents it would be necessary to add filtering when loading data from the database. If that were possible, it would also be interesting to explore the application of generalization techniques that do require an inside and an outside of elements (such as the one by Mi et al. (2009)), in addition to exploring additional generalization algorithms (e. g., a weighted Visvalingam-Whyatt algorithm (Zhou & Jones, 2005)).

In the future it will also be interesting to explore the use of other types of maps such as subway maps as well as the application to 2D shapes other than maps such as general vector data or graphs. In addition, for the currently employed maps we plan to further explore filtering as some elements may be suitable for one location while not as much for others. In this context it would also be interesting to add point data and explore their visual abstractions, e. g., in the form of point clouds. Also, we want to pursue other forms of abstraction such as the fitting of simple shapes such as circles or ellipsoids to polylines. Finally, an evaluation of the approach with people could inform a future development of the tool and/or the types of uses the presented techniques may have.

8 Conclusion

In this article we have explored techniques to *visually abstract* map graphs with the goal of producing aesthetic renditions of the data, distorting the context of map elements. This abstraction is visually inspired by the existing Substrate algorithm but realizes a similar aesthetic based on map data from OpenStreetMap. One may argue that the original Substrate simulation already produces map-like images, but the specific goal of this work is to produce images in a style inspired by Substrate but based on *real* maps that are chosen by the person interacting with the tool and with that person having control over the process—rather than completely controlled by the Substrate algorithm. Therefore, our contribution lies in enabling such interactive exploration of the abstraction process using a number of largely well-known techniques, and applying them to the specific case of map generalization. Specifically, we discussed how to generalize map graphs, how to affect the location of map graph edges, and how to render the resulting abstractions. The explorative interaction with the resulting tool involves controlling a large number of unknowns, which at the same time opens up plenty of possibilities to explore these unknowns. For example, when one applies map graph simplification, it is the specific order and number of steps that are applied for each of the different techniques as well as the chosen visualization that heavily influence the resulting abstraction. The aesthetic appeal of the produced images, consequently, is in the eye of the beholder and people other than the creator of the image may like it or not. We envision applications in the realm of ambient visualization with possibly additional data being displayed on abstracted or non-abstracted map displays. An on-line tool (e. g., similar to MapShaper (Bloch & Harrower, 2006; Harrower & Bloch, 2006)) would also offer the possibility for people to explore their own map data and to create renditions for personal use. Such an on-line tool could easily be created in the future based on the existing Java implementation and could also serve as an informed sandbox for exploring map abstraction designs.

Acknowledgments

We would like to thank Henk Bekker, Maarten H. Everts, Moritz Gerl, Petra Isenberg, and Ross Maciejewski for inspiring discussions on the topic and many suggestions for improvement as well as Jason Dykes for valuable comments on an earlier version of the article.

References

Agrawala, M., & Stolte, C. (2001). Rendering Effective Route Maps: Improving Usability Through Generalization. In *Proc. SIGGRAPH*, pp. 241–249. New York: ACM. doi> 10.1145/383259.383286

Bloch, M., & Harrower, M. (2006). MapShaper.org: A Map Generalization Web Service. In *Proc. AutoCarto*. Cartographic and Geographic Information Society.

Böttger, J., Brandes, U., Deussen, O., & Ziezold, H. (2008a). Map Warping for the Annotation of Metro Maps. *IEEE Computer Graphics and Applications 28*, no. 4 (September/October), pp. 56–65. doi> 10.1109/MCG.2008.99

Böttger, J., Brandes, U., Deussen, O., & Ziezold, H. (2008b). Map Warping for the Annotation of Metro Maps. In *Proc. VGTC Pacific Visualization Symposium*, pp. 199–206. Los Alamitos: IEEE Computer Society. doi> 10.1109/PACIFICVIS.2008.4475477

Bousseau, A., Kaplan, M., Thollot, J., & Sillion, F. X. (2006). Interactive Watercolor Rendering with Temporal Coherence and Abstraction. In *Proc. NPAR*, pp. 141–149. New York: ACM. doi> 10.1145/1124728.1124751

Brewer, C. A. (2009). ColorBrewer 2.0. Web site: http://www.ColorBrewer.org/. Accessed June 2011.

Brinton, W. C. (1939). *Graphic Presenation*. New York: Brinton Associates.

Caquard, S., Piatti, B., & Cartwright, W. (2009). Editorial: Special Issue on Art & Cartography. *The Cartographic Journal 46*, no. 4 (November), pp. 289–291. doi> 10.1179/174327709X12574225938589

Coast, S., et al. (2004). OpenStreetMap. Web site & map data: http://www.openstreetmap.org/. Accessed June 2011.

Coconu, L., Deussen, O., & Hege, H.-C. (2006). Real-Time Pen-and-Ink Illustration of Landscapes. In *Proc. NPAR*, pp. 27–35. New York: ACM. doi> 10.1145/1124728.1124734

Collomosse, J. P., & Hall, P. M. (2003). Cubist Style Rendering from Photographs. *IEEE Transactions on Visualization and Computer Graphics 9*, no. 4 (October–December), pp. 443–453. doi> 10.1109/TVCG.2003.1260739

Curtis, C. J., Anderson, S. E., Seims, J. E., Fleischer, K. W., & Salesin, D. H. (1997). Computer-Generated Watercolor. In *Proc. SIGGRAPH*, pp. 421–430. New York: ACM. doi> 10.1145/258734.258896

DeCarlo, D., & Santella, A. (2002). Stylization and Abstraction of Photographs. *ACM Transactions on Graphics 21*, no. 3 (July), pp. 769–776. doi> 10.1145/566654.566650

Di Battista, G., Eades, P., Tamassia, R., & Tollis, I. G. (1999). *Graph Drawing: Algorithms for the Visualization of Graphs*. Upper Saddle River, NJ, USA: Prentice Hall.

Dorling, D., Barford, A., & Newman, M. (2006). Worldmapper: The World as You've Never Seen it Before. *IEEE Transactions on Visualization and Computer Graphics 12*, no. 5 (September/October), pp. 757–764. doi> 10.1109/TVCG.2006.202

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization 10*, no. 2 (December), pp. 112–122. doi> 10.3138/FM57-6770-U75U-7727

Field, K. (2005). Editorial: 'Maps still matter — don't they?'. *The Cartographic Journal 42*, no. 2 (September), pp. 81–82. doi> 10.1179/000870405X61405

Field, K. (2009). Editorial Preface: Art in C'art'ography. *The Cartographic Journal 46*, no. 4 (November), p. 287. doi> 10.1179/174327709X12574225348881

Garland, M., & Heckbert, P. S. (1997). Surface Simplification Using QuadricError Metrics. In *Proc. SIGGRAPH*, pp. 209–216. New York: ACM. doi> 10.1145/258734.258849

Grabler, F., Agrawala, M., Sumner, R. W., & Pauly, M. (2008). Automatic Generation of Tourist Maps. *ACM Transactions on Graphics 27*, no. 3 (August), pp. 1–11. doi> 10.1145/1360612.1360699

Harmon, K. (2009). *The Map as Art: Contemporary Artists Explore Cartography*. New York: Princeton Architectural Press.

Harrower, M. A., & Bloch, M. (2006). MapShaper.org: A Map Generalization Web Service. *IEEE Computer Graphics and Applications 26*, no. 4 (July/August), pp. 22–27. doi> 10.1109/MCG.2006.85

Harrower, M. A., & Brewer, C. A. (2003). ColorBrewer.org: An Online Tool for Selecting Color Schemes for Maps. *The Cartographic Journal 40*, no. 1 (June), pp. 27–37. doi> 10.1179/000870403235002042

Hausner, A. (2001). Simulating Decorative Mosaics. In *Proc. SIGGRAPH*, pp. 573–580. New York: ACM. doi> 10.1145/383259.383327

Hertzmann, A. (2003). A Survey of Stroke-Based Rendering. *IEEE Computer Graphics and Applications 23*, no. 4 (July/August), pp. 70–81. doi> 10.1109/MCG.2003.1210867

Hoppe, H. (1996). Progressive Meshes. In *Proc. SIGGRAPH*, pp. 99–108. New York: ACM. doi> 10.1145/237170.237216

Jenny, B., & Hurni, L. (2011). Studying Cartographic Heritage: Analysis and Visualization of Geometric Distortions. *Computers & Graphics 35*, no. 2, pp. 402–411. doi> 10.1016/j.cag.2011.01.005

Kent, A. J. (2005). Aesthetics: A Lost Cause in Cartographic Theory? *The Cartographic Journal 42*, no. 2 (September), pp. 182–188. doi> 10.1179/000870405X61487

Krygier, J. (2006). Jake Barton's Performance Maps: An Essay. *Cartographic Perspectives 53*, no. Winter, pp. 41–50.

Krygier, J. B. (1995). Cartography as an Art and a Science? *The Cartographic Journal 32*, no. 1 (June), pp. 3–10.

Ljungberg, C. (2009). Cartographies of the Future: Julie Mehretu's Dynamic Charting of Fluid Spaces. *The Cartographic Journal 46*, no. 4 (November), pp. 308–315. doi> 10.1179/000870409X12538748663496

Luft, T., & Deussen, O. (2006). Real-Time Watercolor Illustrations of Plants Using a Blurred Depth Test. In *Proc. NPAR*, pp. 11–20. New York: ACM. doi> 10.1145/1124728.1124732

Mehra, R., Zhou, Q., Long, J., Sheffer, A., Gooch, A., & Mitra, N. J. (2009). Abstraction of Man-Made Shapes. *ACM Transactions on Graphics 28*, no. 5 (December), pp. 137:1–137:10. doi> 10.1145/1618452.1618483

Mi, X., DeCarlo, D., & Stone, M. (2009). Abstraction of 2D Shapes in Terms of Parts. In *Proc. NPAR*, pp. 15–24. New York: ACM. doi> 10.1145/1572614.1572617

Panse, C., Sips, M., Keim, D., & North, S. (2006). Visualization of Geo-spatial Point Sets via Global Shape Transformation and Local Pixel Placement. *IEEE Transactions on Visualization and Computer Graphics 12*, no. 5 (September/October), pp. 749–756. doi> 10.1109/TVCG.2006.198

Ramer, U. (1972). An Iterative Procedure for the Polygonal Approximation of Plane Curves. *Computer Graphics and Image Processing 1*, no. 3 (November), pp. 244–256. doi> 10.1016/S0146-664X(72)80017-0

Roberts, M. J. (2009). Henry Beck Rules, not OK? Breaking the Rules of Diagrammatic Map Design. Online essay, available at http://privatewww.essex.ac.uk/~mjr/underground/Breaking_the_rules.pdf.

Robinson, A. H., Sale, R. D., Morrison, J. L., & Muehrcke, P. C. (1984). *Elements of Cartography*. New York: John Wiley & Sons, 5th ed.

Schoenholz Bee, H., & Heliczer, C. (Eds.) (2004). *MoMA Highlights: 350 Works from the Museum of Modern Art, New York*. New York: Museum of Modern Art, 2nd ed.

Tarbell, J. (2003). Substrate. Web site and simulation: http://www.complexification.net/gallery/machines/substrate/. Accessed June 2011.

Varanka, D. (2006). Interpreting Map Art with a Perspective Learned from J.M. Blaut. *Cartographic Perspectives 53*, no. Winter, pp. 15–23.

Visvalingam, M., & Whyatt, J. D. (1993). Line Generalisation by Repeated Elimination of Points. *The Cartographic Journal 30*, no. 1 (June), pp. 46–51.

Watson, R. (2009). Mapping and Contemporary Art. *The Cartographic Journal 46*, no. 4 (November), pp. 293–307. doi> 10.1179/000870409X12549997389709

Wobbrock, J. O., Wilson, A. D., & Li, Y. (2007). Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proc. UIST*, pp. 159–168. New York: ACM. doi> 10.1145/1294211.1294238

Wood, D. (2006). Map Art. *Cartographic Perspectives 53*, no. Winter, pp. 5–14.

Zhou, S., & Jones, C. B. (2005). Shape-Aware Line Generalisation With Weighted Effective Area. In *Developments in Spatial Data Handling*, pp. 369–380. Berlin, Heidelberg: Springer Verlag. doi> 10.1007/3-540-26772-7_28

A Demo Program

The Java program used to create the images shown in the article can be accessed via Java Webstart at http://osmabstraction.isenberg.cc/ and can be run on Windows, Linux, MacOS X, and other platforms that provide a Java Runtime Environment (JRE). The program does not only support the loading of map data files that were downloaded from the OpenStreetMap website but can also directly access the OpenStreetMap data through the OSM Web API.