STUDIA INFORMATICA 2012
Volume 33 Number 2B (106)

Tomasz PŁUCIENNIK
Silesian University of Technology, Institute of Computer Science

VIRTUAL CITY GENERATION FOR GIS SYSTEMS TESTING

Summary. Geographic Information Systems are responsible for processing and presenting geographical data. Development of GIS systems requires extensive tests based on actual data. Access to detailed datasets is restricted which led to the necessity of replacing them with synthetic data. This article presents extended version of a generator of spatial data layers, which could be used in mentioned tests.

Keywords: GIS (Geographic Information Systems), OGC (Open Geospatial Consortium) Web Services, shapefile, spatial data, urban planning

GENEROWANIE WIRTUALNEGO MIASTA DLA TESTOWANIA SYSTEMÓW TYPU GIS

Streszczenie. Systemy informacji przestrzennej (ang. *Geographic Information Systems*) zajmują się przetwarzaniem i prezentacją danych geograficznych. Opracowywanie systemów GIS wymaga obszernych testów opartych na danych rzeczywistych. Dostęp do szczegółowych zbiorów jest ograniczony, co stworzyło potrzebę zastąpienia ich przez dane syntetyczne. Niniejszy artykuł prezentuje rozbudowaną wersję generatora warstw danych przestrzennych, które można wykorzystać we wspomnianych testach.

Słowa kluczowe: dane przestrzenne, plik shapefile, systemy informacji przestrzennej (GIS), urbanistyka, usługi sieciowe OGC (Open Geospatial Consortium)

1. Introduction

Geographic Information Systems (GIS) are systems for managing, storing and presenting spatial data describing objects placed on earth's globe [1]. Spatial information is, from the point of view of the user, presented as images corresponding to raster maps or objects. These objects are called features, grouped into layers. A *layer* contains objects of the same type.

A *feature* is constituted of descriptive attributes and the geometry used to display it on the map. Geometry can describe a point, a line or a polygon. One feature can contain multiple geometries or no geometries at all. All geometries consist of points describing its vertices and internal holes (if they are present) and can be either two- or three-dimensional.

Problem of testing arises during the development of GIS systems. It is required to test the system integrity and performance with representative, large datasets of spatial data. The perfect option is to own the target data for the system, but usually this is not the case. One can use dataset available on the internet e.g. in [2], however it is hard to find data with required amount of detail and/or containing the sufficient set of layers describing the same area. To the author's knowledge there are also no GIS data generators available. Furthermore, it might be required to test system's responses against broken data e.g. invalid geometry. Taking all this into consideration a need for generic dataset generator have arisen.

The generator is an application creating ESRI (Environmental Systems Research Institute) shapefiles [3] containing layers depicting objects in a virtual city area. Created data can be uploaded into a GIS system and presented as either a WFS (Web Feature Service) or WMS (Web Map Service) layers [4], which are part of OGC (Open Geospatial Consortium) standards.

The generator is a tool which will support the GIS systems development. The first version was created in 2011 and was presented in [5]. It was able to generate a road structure of a city but with some limitations. Additionally it was wrongly assumed that this kind of layers are polygon layers, while they are typically represented as lines. This made calculating bends on roads complicated. Therefore, in the first version of the generator all roads were straight. Based on this type of roads generation of parcels, buildings and trees began. Then new implementation of roads was created and other layers were accommodated to fit it. The result of this implementation is presented in this paper. The project is undergoing modifications and it is still being extended.

2. Current Status

As of today the layers that can be generated are in order:

- roads,
- parcels,
- buildings (and simple three-dimensional buildings),
- trees.

In the final version of the software rail tracks and water layers will be available. The user provides which layers are needed using command line (note that e.g. parcels cannot be gener-

ated without roads layer). The order of generation is important since the layers have to satisfy spatial joins operation [6].

Road layer is based on a fractal [5] – a road has its outgoing roads and so forth. The geometries are now lines. Using Bézier curves allow to calculate smooth bends (Fig. 1).

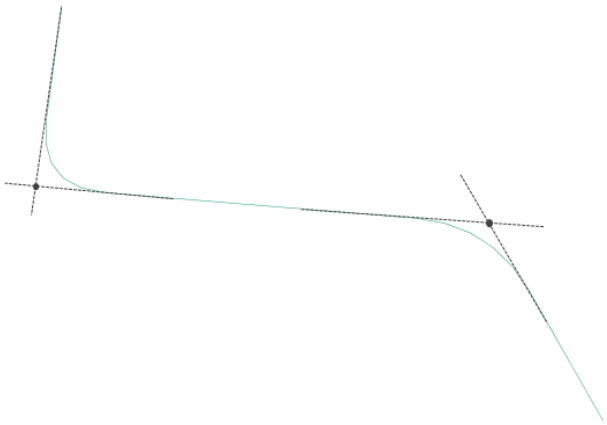

Fig. 1. Bézier curve based bends
Rys. 1. Zakręty oparte na krzywych Béziera

Parcels are of course polygons. Parcels are firstly placed over existing roads. The width of these parcels are based on road parameter describing its actual width plus sideway width. Then empty places between roads are transformed into temporary areas and divided into residential parcels by cutting then along chosen lines (Fig. 2). The prepared parcel types are *roads*, *residential*, *rail*, *water* and *empty*.

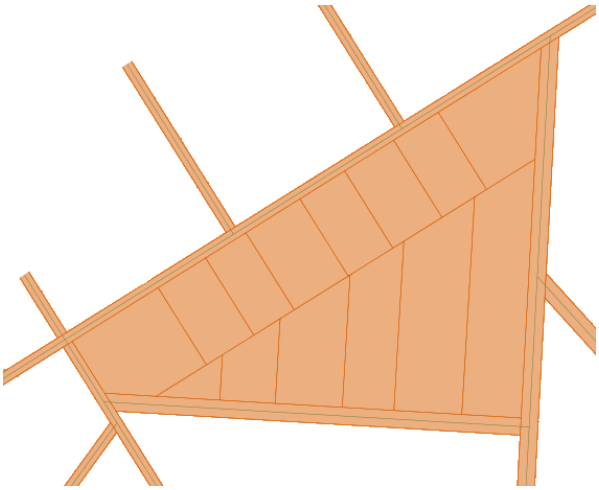

Fig. 2. Example of division of an area into parcels
Rys. 2. Przykład podziału obszaru na działki

Buildings are then placed on residential parcels. Each building have to fully fit inside its parcel. The shapes are based on the shape of the current parcel or based on rectangles (Fig. 3), therefore sides of the building are most probably parallel to parcel's borders.

Trees are placed randomly on residential places not overlapping buildings and other trees. Number of trees per parcel is randomized.

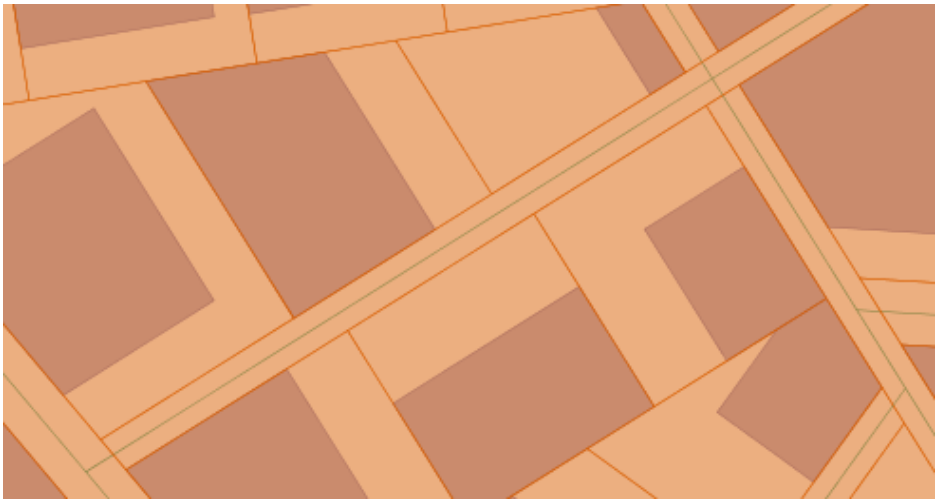

Fig. 3. Example buildings
Rys. 3. Przykładowe budynki

In every layer generation an element of randomization is added e.g. placing of bends when calculating roads, dividing parcel areas into not necessarily equal parts or randomly positioning buildings on residential parcels. Additionally, a Gauss density probability function is used to simulate city centre: closer to the middle of the map roads structure is denser, parcels are smaller and buildings have a tendency to be taller.

In the future release some of the parcels will be marked not as residential, but as rail or water. The problem will be to set them correctly i.e. to create continuous rail tracks and rivers (Fig. 4).

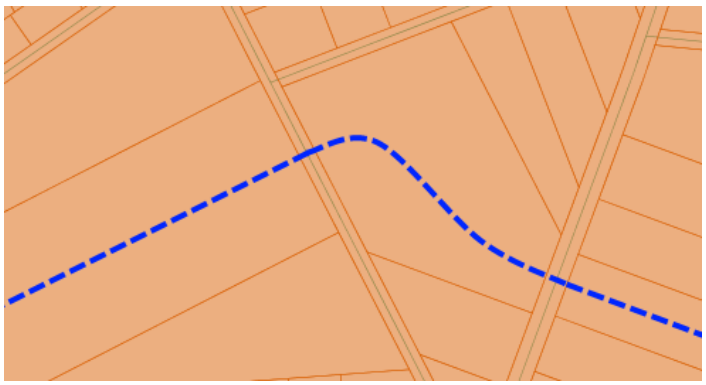

Fig. 4. Planned layers construction
Rys. 4. Konstrukcja planowanych warstw

A spatial feature is constituted of geometry and attributes. Attributes in generator are divided into two groups: automatic and user-defined. Automatic attributes are always outputted and they list is as follows:

- *id* – created for every layer,
- *type* – used for roads and parcels,
- *lane_count* – number of road's lanes,

- *name* – name of an object (now used only for roads),
- *tier_up* – number of building levels above the ground level,
- *tier_down* – number of building levels below the ground level,
- *tier_height* – building level height in metres.

User-defined attributes can represent integer or floating point number, text or a date. User can provide any name for the attribute and values will be automatically generated according to the attribute's type.

3. Data-related Optimization

The application is optimized in terms of its performance and memory consumption. In today's world memory optimization lost its previous priority. This of course does not mean that it should be ignored. In terms of optimizing speed, excessive memory usage might also decrease performance. Typical optimization techniques [7] (starting from. reorganizing loops or tree iteration etc.) can be extended if assumptions can be made upon the processed data. If the data is known, there are even more possibilities of better optimization. During the development two assumptions were affiliated:

- *Occam's Razor* [8] – during the generation of a complicated data layer always the simplest solution is the best (the quickest way to create a working program),
- if an object is somehow generated incorrectly it is removed, e.g. one missing road will not make any visible difference and recalculating that one geometry might take too much time.

In the previous version of the generator roads were represented as polygons. The amount of operation required to calculate the geometry exact position were high, especially when touching roads (crossings) were cut to fulfil the requirement of not overlapping each other [5]. In case of ridiculously large city maps (e.g. 100 over 100 km) the generation time was reaching 2 hours. To make matters worse no bends were yet generated. Fortunately typical size map (e.g. 10 over 10 km) were generated in minutes. Still no other layer was outputted.

In real datasets roads are represented as lines and the polygon-like view of them is generated on the basis of object's attributes. In the previous version of the generator roads' axes were created as auxiliary layer used for debugging and this layer was actually close to how the roads layer should look [5]. This design mistake is now corrected, which is not a optimization as such, but the new version of roads layer is generated in minutes even for the biggest areas. Furthermore it requires less memory and makes calculating bends' curves easier.

Parcel generation has two distinct phases:

- creating parcels for roads,
- filling empty spaces with parcels for e.g. buildings.

After changing the road layer format the first phase of parcels generation became a bit more complicated. Previously the only thing that has to be done is to copy the roads' geometries. Now it is required to take the road line, add the side lines (taking special care of the bends) and, as in previous version of roads layer, make sure that geometries are not overlapping each other as shown in Fig. 5.

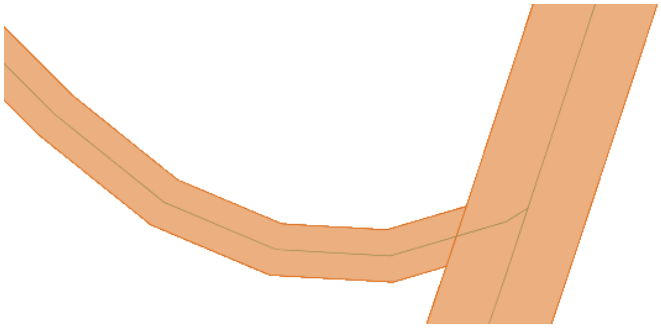

Fig. 5. Example of roads' parcels
Rys. 5. Przykładowe działki dla dróg

Filling empty spaces between roads was suppose to use a two-dimensional table storing flags describing whether a point on a map (with resolution to e.g. 1 metre) is occupied. An example is presented in Fig. 6. The table is filled as roads and parcels are added. The algorithm used a hashmap to store the table (coordinates in the table constituted the key) to limit memory usage. Search for the shape of empty area starts from a sample point (gray dot in Fig. 6) placed somewhere outside of any other geometry in the parcel layer (any road parcel or already found area). An algorithm similar to contour filling used in raster graphics [9] would help to locate geometries surrounding the supposed empty area. Area on the mentioned table was filled using optimized version of prairie fire algorithm using a queue instead of recursion. The result was a bounding area (marked rectangle in Fig. 6), used then to search in the spatial index for surrounding parcel geometries and to construct the shape of the final area (polygon marked in with dashed line in Fig. 6). The found area was then divided into parcels. This was implemented in very simple way along with very simple trees and buildings generation.

The above solution was working with straight roads, however when area shape was concave it either created empty spots on the map or returned a few polygons instead of one (which have to be then merged together), since an empty spot might be sampled later (geometry cutting was based on cutting along chosen lines [5]). The areas were then divided into parcels. When bends were added to road parcels the algorithm became insufficient since sampling points were not placed densely enough. Furthermore the performance were degraded because of the constant searching over map objects.

In the new version of the application two main changes were made to second phase of parcels generation. The first is using JTS (Java Topology Suite) [10] geometry cutting possibilities (requires converting geometry objects temporary into JTS format). This takes care of concave geometries. The second change is adding second index to roads layer. Aside of the spatial index (quadtree) a tree is built in which outgoing road is explicitly set as a child of the main road. Search for empty areas' geometries is done by assuming one big area equal to the map boundaries and iterating through successive levels of the tree. Every road divides the area(s) it is passing through. When all roads are used, the created list contains areas ready to be divided. No large table is required and geometries surrounding an area are found basically instantly.

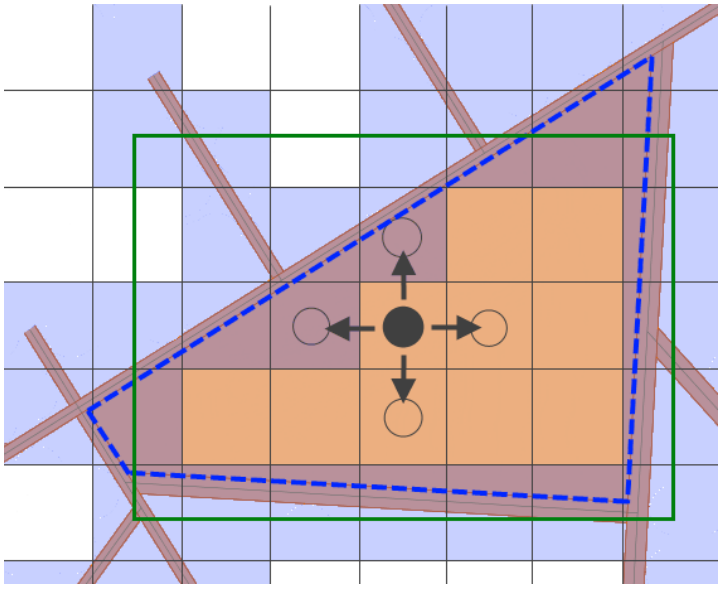

Fig. 6. Old version of search for areas to fill with parcels
Rys. 6. Poprzednia wersja wyszukiwania obszarów do wypełnienia działkami

Currently an empty area is divided only into residential parcels. It is cut in more or less half using line parallel to the longest vector in its exterior. These two parts are then cut again using lines perpendicular to area's exterior (refer to Fig. 2). Positions of cut lines are randomized to make it look more realistic. Additionally, every residential parcel touching the map's boundaries is removed to simply create the city's border.

4. Comparison with Real Data

To represent the current possibilities of spatial data generation a comparison will be presented against the real data. The real data is taken from city plan for city of Gliwice [11], where it is served as WMS layers. Please note, that WMS server is displaying some objects only in smaller scales. The view of generated data is done using uDig [12] software. The gen-

erated city is limited by 20 by 20 km boundaries. The actual area covered by the generated dataset is around 100 km^2, which makes it close to Gliwice city size.

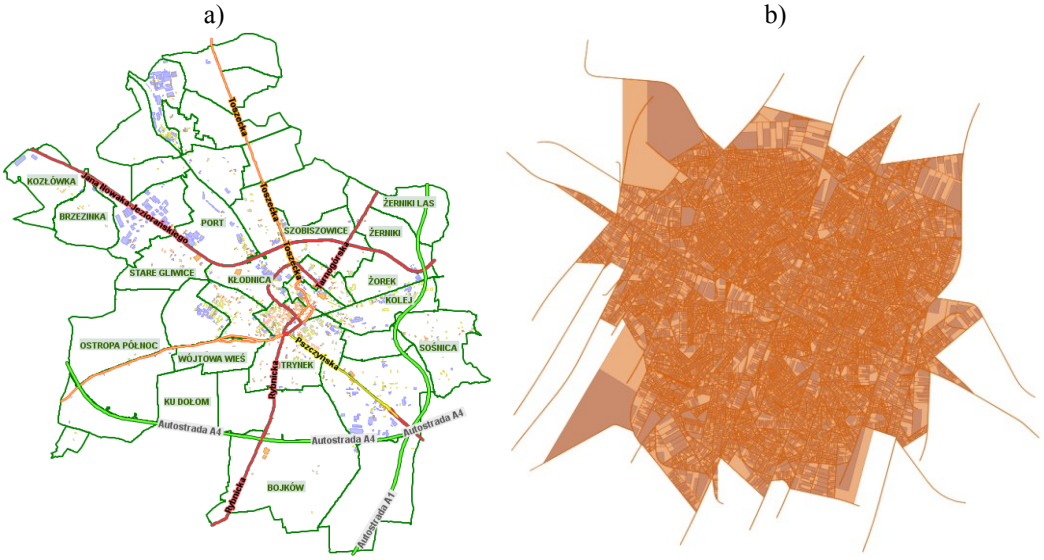

Fig. 7. Map view: a) real city data, b) generated city data
Rys. 7. Podgląd mapy: a) dane rzeczywistego miasta, b) dane wygenerowane

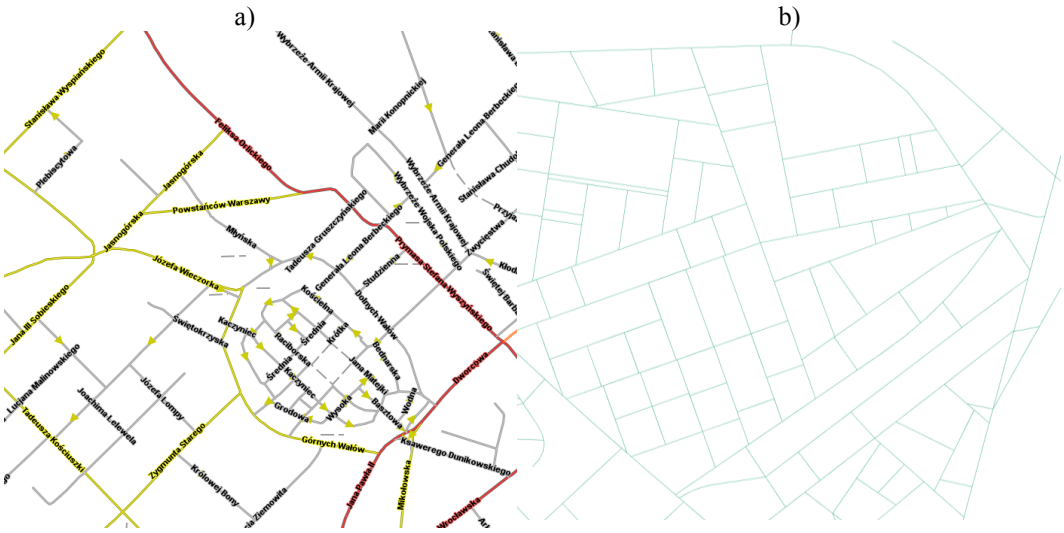

Fig. 8. Road layer view: a) real structure, b) generated structure
Rys. 8. Podgląd warstwy dróg: a) struktura rzeczywista, b) struktura wygenerowana

Fig. 7 presents the overview of real and generated city respectively. All layers except for trees are outputted on both map. The following figures will show more detailed views. Fig. 8 presents chunks of roads' structures. In the Fig. 9 a close-up view of parts of the cities are depicted with visible parcels and buildings. Note that generated datasets are not that different from what one can find in the real world. Fig. 10 shows trees placement on a virtual city. The features count for the example generated city is:

- roads: 1376,
- parcels: 15450 (including residential parcels: 14074),

- buildings: 13876,
- trees: 6330.

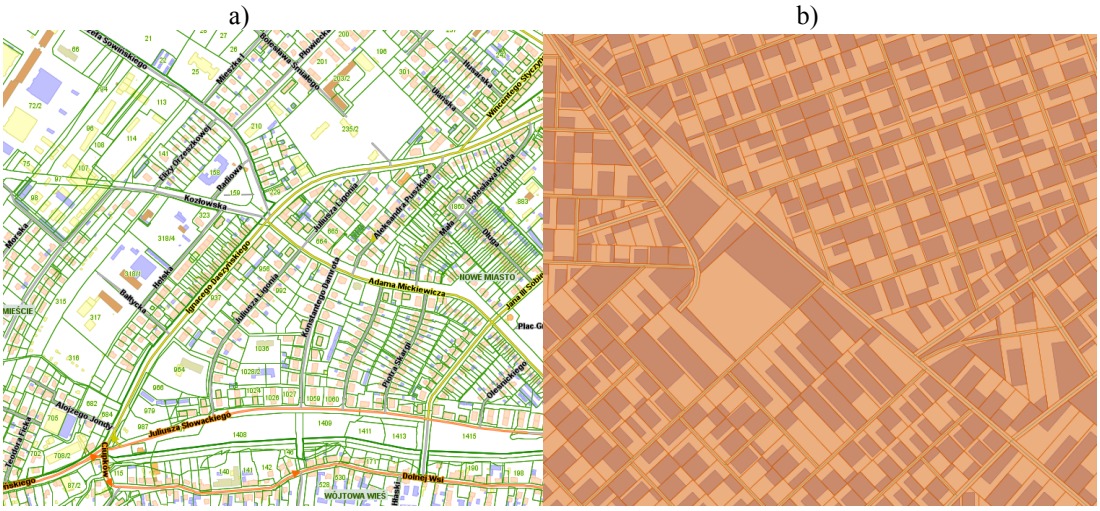

Fig. 9. Map close-up view: a) real data, b) generated data
Rys. 9. Podgląd fragmentu mapy: a) dane rzeczywiste, b) dane wygenerowane

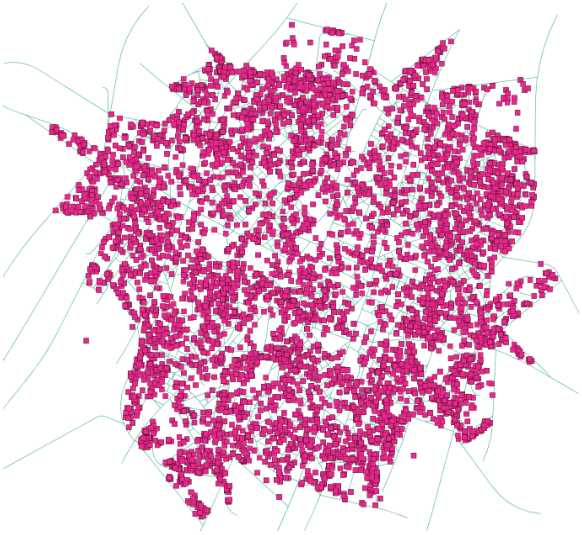

Fig. 10. Trees placement on a virtual city
Rys. 10. Rozmieszczeni drzew na terenie wirtualnego miasta

Finally, in Fig. 11 a three-dimensional visualization of buildings in Gliwice is presented against the generated ones. The screenshots were made using Aurora 3D [13] and ArcGIS ArcScene [14] respectively.

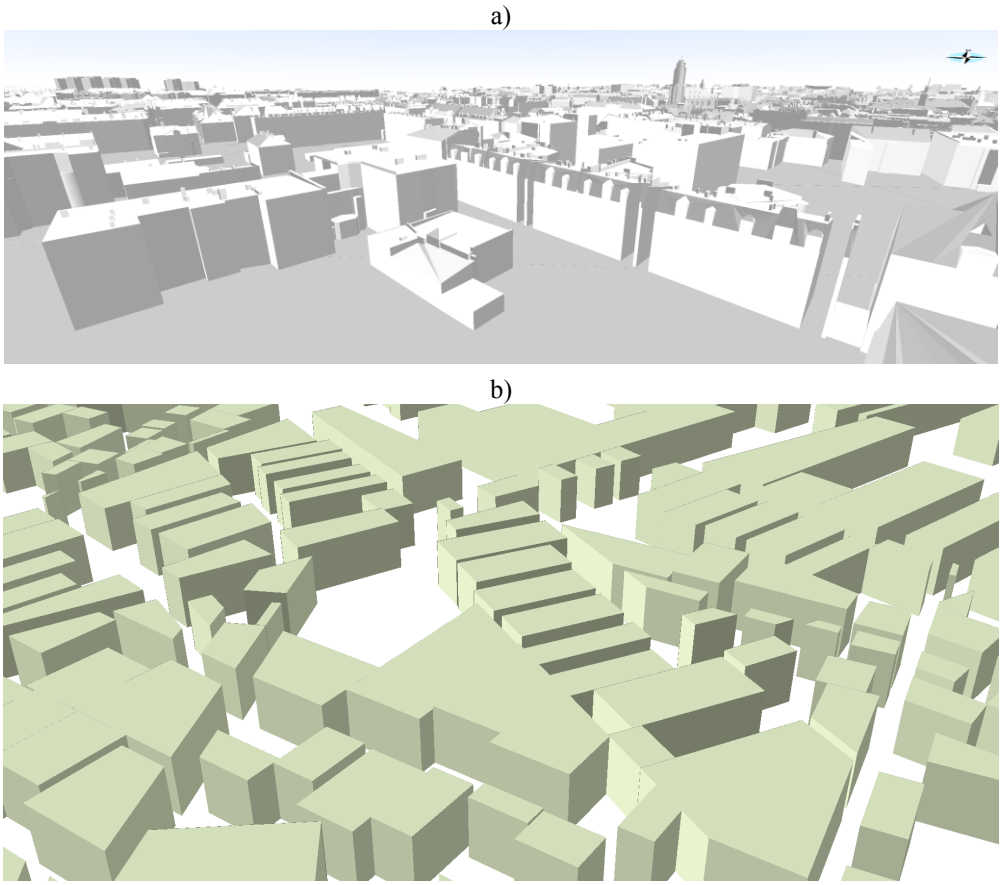

Fig. 11. View of the three-dimensional buildings layer: a) real dataset, b) generated dataset
Rys. 11. Podgląd warstwy budynków trójwymiarowych: a) zbiór rzeczywisty, b) zbiór wygenerowany

5. Conclusion

To sum up, the project underwent extensive rebuilding. Existing road layers were updated to have more realistic view and at the same time the application was given much needed optimizations. Although it is working fast enough at the current stage, the optimizations will be continued. The focus will be put on adding the missing water and rail layers. The basic idea here is to select some of the parcels to place rivers, water reservoirs and rail tracks on them.

In the further perspective a graphical user interface will be added along with possibility to change parameters of the generator, define additional attributes for layers depending on designed system requirements or force errors in geometries. Since all layers are created on demand there are endless possibilities of modifications. Changes might be also required during the generator's actual usage to create the test data. Aside of tweaking the style of the layers, a need for additional layers might be discovered during development of a GIS system. Since adding new layers themselves (like e.g. administration areas, power lines, pipes or simply markers of positions of selected public institutions) might not be especially complicated, only a plug-in mecha-

nism might be welcome. A generation plug-in could be written and tested separately and used without the need to rebuild the whole application. One have to remember that it is impossible to predict what types of layers will required in the future and so the generator must be as versatile as possible. There are also plans to implement a three-dimensional terrain layer.

Acknowledgements

This work was supported by the European Union from the European Social Fund (grant agreement number: UDA-POKL.04.01.01-00-106/09).

BIBLIOGRAPHY

1. Longley P. A., Goodchild M. F., Maguire D. J., Rhind D. W.: Geographic Information Systems and Science. John Wiley & Sons Ltd, 2005.
2. http://www.esri.com/data/free-data/index.html (access 2012-02-15).
3. ESRI Shapefile Technical Description, An ESRI White Paper, 1998.
4. Kubik T.: GIS. Rozwiązania sieciowe. PWN, Warszawa 2009.
5. Płuciennik T.: Roads Structure of a Virtual City for GIS Systems Testing. Studia Informatica, Vol. 32, No. 2B (97), Wydawnictwo Politechniki Śląskiej, Gliwice 2011.
6. Papadias D., Mamoulis N., Theodoridis Y.: Processing and Optimization of Multiway Spatial Joins Using R-trees. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), 1999.
7. Korte B., Vygen J.: Combinatorial Optimization: Theory and Algorithms (Algorithms and Combinatorics) Fifth Edition. Springer, 2012.
8. Ford N.: The Productive Programmer. O'Reilly Media, 2008.
9. Desai A. A.: Computer Graphics. PHI Learning Private Limited, 2008.
10. JTS Topology Suite Technical Specifications Version 1.4. Vivid Solutions, 2003.
11. http://msip-mapa.um.gliwice.pl/geoportaltoolkit/map.php?skin=gliwice_new_mieszkaniec (access 2012-02-15).
12. http://udig.refractions.net/ (access 2012-02-15).
13. http://msip-mapa.um.gliwice.pl:8080/aurora-gliwice/app.jnlp (access 2012-02-15).
14. http://www.esri.com/software/arcgis/extensions/3danalyst/index.html (access 2012-02-15).

Wpłynęło do Redakcji 31 stycznia 2012 r.

Omówienie

Systemy GIS służą do przechowywania, przetwarzania i prezentacji obiektów położonych na globie ziemskim. Dane geograficzne są prezentowane użytkownikowi w postaci graficznej, a zapisywane są w postaci tychże grafik lub w postaci obiektów (*features*) zgrupowanych w warstwy (*layers*). Obiekty posiadają atrybuty opisowe oraz geometrię, która może być pojedynczym punktem, linią, wielokątem bądź grupą geometrii. Geometria może też być nieobecna. Budowa systemu typu GIS wymaga rozbudowanych testów, najlepiej wykorzystujących dane rzeczywiste. Ograniczenia w możliwości uzyskania kompletnego zbioru warstw dla danego obszaru stworzyły potrzebę wygenerowania syntetycznych warstw, których struktury przestrzenne będą do siebie pasować.

Powstały generator testowych danych przestrzennych jest w stanie wygenerować dane dotyczące ulic, działek, budynków i drzew dla zurbanizowanego obszaru. Warstwa dróg jest drzewem opartym na fraktalu, a zakręty są tworzone przy użyciu krzywych Béziera (rys. 1). Warstwa działek jest częściowo oparta na drogach, a częściowo wygenerowana przez dzielenie niewykorzystanych obszarów na działki budowlane (rys. 2). Budynki (rys. 3) i drzewa są na koniec losowo rozmieszczane na działkach. W stosunku do poprzedniej wersji generatora nastąpiła optymalizacja, zaczynając od zmian w warstwie dróg, a kończąc na wykorzystaniu dodatkowych indeksów do generowania działek.

W porównaniu do danych rzeczywistych, dane z generatora wypadają na tyle dobrze, by moc zastąpić je podczas testów systemów GIS. Przykładowe wyniki zostały przedstawione na rys. 7, 8, 9 i 10 w porównaniu z danymi pobranymi z mapy miasta Gliwice, dostępnej w internecie [11]. Dodatkowo na rys. 11 pokazany został przykład trójwymiarowej warstwy budynków wraz z modelem rzeczywistym.

Planowana jest rozbudowa generatora, obejmująca nowe warstwy danych (linie kolejowe i cieki wodne) oraz graficzny interfejs użytkownika, który pozwoli lepiej sparametryzować aplikację. Nie wykluczone jest, że już podczas opracowywania systemów GIS z użyciem generatora wymagane będą dodatkowe parametryzacja i wymuszenia określonych zachowań generatora, jak np. generowanie niepoprawnych geometrii. W przyszłości mogą okazać się również potrzebne nowe, nie przewidziane jak dotąd, warstwy danych. Aby przygotować się na taką ewentualność, planuje się umożliwienie dodawania do aplikacji wtyczek budujących nowe warstwy przestrzenne.

Address

Tomasz PŁUCIENNIK: Silesian University of Technology, Institute of Computer Science, Akademicka 16, 44-100 Gliwice, Poland, Tomasz.Pluciennik@polsl.pl.