STUDIA INFORMATICA 2011
Volume 32 Number 2B (97)

Tomasz PŁUCIENNIK
Silesian University of Technology, Institute of Computer Science

GENERATING ROADS STRUCTURE OF A VIRTUAL CITY FOR GIS SYSTEMS TESTING

Summary. Geographic Information Systems are computer systems for storing, managing and presenting geographical data. Development of such systems requires extensive tests based on large datasets of actual data. Because of access restrictions it is apparent that artificial test data closely imitating the real data should be prepared. This article describes initial phase of creating a spatial dataset corresponding to a virtual city – roads structure generation.

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

GENEROWANIE SIECI DRÓG WIRTUALNEGO MIASTA DLA TESTOWANIA SYSTEMÓW TYPU GIS

Streszczenie. Komputerowe systemy informacji przestrzennej (Geographic Information Systems) wykorzystywane są do przechowywania, zarządzania i prezentacji danych geograficznych. Budowa takich systemów wymaga obszernych testów opartych na dużych zbiorach danych rzeczywistych. Ze względu na ograniczenia w dostępie do takich zbiorów, konieczne jest przygotowanie sztucznych danych, które jak najlepiej zastąpią prawdziwe. Niniejszy artykuł opisuje pierwszy etap tworzenia danych przestrzennych opisujących wirtualne miasto, którym jest generacja sieci dróg.

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

1. Introduction

Geographic Information Systems are computer systems for storing, managing and presenting spatial data as maps, globes, reports and charts [1]. *Spatial data*, also known as *geospatial* *data* or *geographic information*, is the information that identifies the geographic location of features and boundaries on earth [1]. The data is acquired using satellite or plane imagery, lidar scans etc. Preparing of this data is connected with a phase of postprocessing (filtering, repairing errors created by gathering method imperfections). This phase is sometimes done manually. Therefore the whole process costs a lot of time and money.

Development of GIS systems requires extensive tests based on data which should fulfil two conditions:

- size – must be large since in the real world the system will probably be used in an environment requiring big amount of data to be created, stored and exchanged,
- quality – data must be similar to real data to check correctness of systems functions.

One way to acquire this data is to have an example dataset of the real data. Because of legal restrictions concerning use of this data (not everything is public), the fact that the data might not be prepared before the system itself starts to run in the production environment and the fact that public spatial datasets are not that accurate, it is apparent that artificial test data should be prepared. This data should fulfil mentioned above two conditions. The data will then be used for testing Geographic Information Systems operation speed, proper data visualization and processing.

2. Spatial Information

Spatial data can have a form of pictures (rasters) or binary objects (vector geometry) representing elements on a map [2]. Different category is spatial *metadata* [2] which consists of special XML documents describing spatial data available in the web. Spatial data objects are called *features*. Spatial data in a form of pictures corresponds to satellite images or features (transformed into images to limit the amount of data downloaded and processed by a web client application – geoportal), like in the example taken from Google Maps in Fig. 1.

Features of the same type are grouped into *layers*. An image (e.g. ortophotomap which is a picture of an area) also constitutes a layer.

To maintain all this data the following special SOAP services are used:

- WFS – Web Feature Service for the features,
- WMS – Web Map Service for imagery,
- CSW – Catalog Service for the Web for the metadata,
- other services.

The services provide operations for adding, modifying and removing of supported objects, as well as additional operations like providing server capabilities (not all operations are mandatory in every service type). This covers the transmission of spatial data around the web. Spatial Data Infrastructure (SDI) [3] encompasses all standards for accessing and exchanging spatial data. In a GIS system, the geospatial information in most cases is stored in a database. Here support for spatial infrastructure is usually provided by the database extensions: Oracle Spatial or PostGIS for PostreSQL to name a few.

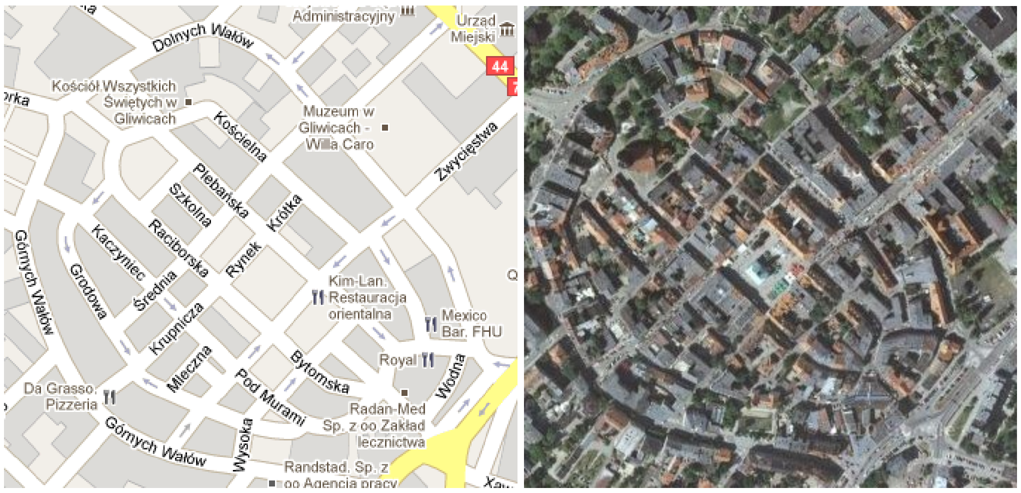


Fig. 1. WFS and WMS data inside of the Google Maps portal
Rys. 1. Dane WMS i WFS w portalu Google Maps

This paper deals with automatic generation of feature layers. As mentioned before WMS data can be created by drawing features on an image canvas. A *feature* contains *geometries* and additional *attributes*. *Attributes* hold user-readable descriptive values (text, date, number values etc.). They might contain description (e.g. street name) or any attribute required to show data properly (e.g. object identifier, material from which the object is built etc.). Geometries can be two- or three-dimensional. They can be divided into three groups:

- point geometries – used to represent e.g. trees,
- linear geometries – used for e.g. borders, roads axes; they have two subtypes: *LineString*, which is a list of connected vertices and *LinearRing*, which is a closed *LineString* [4],
- polygon geometries – polygons with any number of vertices higher than two.

Many geometries can be stored in one feature so the above types have their multi-geometry variations. A vertex has two (or three) coordinates placed in Cartesian *coordinate reference system* (CRS). European Petroleum Survey Group (EPSG) defined a group of standardized local coordinate systems, which use a meter as a unit and differs by the area (zone) which they cover (ranges of values of *x* and *y* in every CRS are explicitly defined, *z* value corresponds to altitude above the sea level). They all take into account the earth's curvature. Zones definitions changed over the years. CRS names are taken from the year they were defined (e.g. 1965, 2000). For example, all of the data concerning the Silesian area is covered by *Poland CS2000 zone 6* coordinate system. All reference systems have a EPSG code – this one has code *EPSG:2177*. There are also other reference systems, like *World Geodetic System '84* (WGS-84), which is a global reference system and has ellipsoid based coordinates describing longitude and latitude. It is used on paper maps to present coordinates in human-readable format.

As mentioned before spatial data is stored in databases and transferred using SOAP web services. Outside GIS systems, e.g. when they are being prepared for upload, it is stored in files. There are many special file formats [5, 6] ranging from text formats (e.g. WKT – Well-known Text) to binary formats (e.g. ESRI Shapefile, MapInfo). The data generator will use shapefiles for storing layers. This format is well-supported by GIS data editing applications (e.g. ArcGIS, uDig) and programming libraries (e.g. GeoTools). It is therefore simple to use and popular – GIS servers like GeoServer supports shapefile import. The data could be stored in a database. However using files will eliminate problems with database format or DBMS usage, making the solution more universal. Even if this kind of import is not possible, the editing applications can convert data to other formats.

One shapefile contains one layer of spatial data. Shapefile is actually built from the following files:

- *.shp – shape file with geometries,
- *.shx – index file with a index for the geometries,
- *.dbf – database file with attributes,
- other files (*.fix, *.gix, *.sbn, *.sbx etc.) which are not required.

3. Generator Background

In the final version, the spatial test data generator will generate layers corresponding to a virtual urban area. The requirement for the layers is to match each other (correctly overlapping geometries giving error-free result of spatial join operation [7]). The data itself does not have to look like actual data but like something that might be one when simplifications are assumed. The layers that will be created are: roads, parcels, watercourses, rail tracks, buildings (three-dimensional layer or at least layer containing attributes for floors), green areas. All these layers except buildings will be two-dimensional. Additionally three-dimensional terrain layer might be incorporated for use (along with buildings) in 3D visualization systems. Shapefiles containing created data will be manually uploaded to the destination GIS system. There are no special time restrictions for the phase of generating the layers.

The easiest way to complete the generator will be to find and adapt an existing solution to obtain required data format. Found solutions were not very realistic and not always open source preventing them to be modified:

- Random Map Generator v1.0 [8] – used a grid which was randomized to create a map with still visible pattern (Fig. 2222a),
- Roleplaying City Map Generator [9] – used in games development; source code is not accessible (Fig. 2b).

Spatial data generation attempt was made in [10] to help agricultural development by simulating fields expansion and in [11] to measure cities using fractals. This is of course totally different type of data. Therefore the generator has to be written from the ground up.

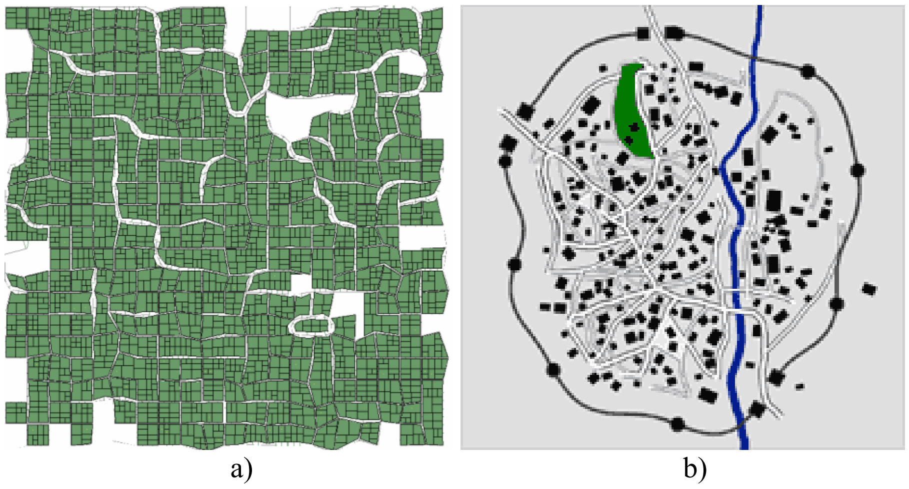


Fig. 2. Output of: a) Random Map Generator, b) Roleplaying City Map Generator
Rys. 2. Wynik działania aplikacji: a) Random Map Generator, b) Roleplaying City Map Generator

This paper deals with the first phase of implementation, which is the city streets structure. Other layers will be derived based on it to satisfy the correct geometry overlapping requirements. These requirements apply of course to the data inside the layer itself and between the layers. Now focus will be put on creation of roads features: their geometries and attributes.

3.1. Fractal Randomization

Fractal is an object created of itself having a property of *self-similarity* [11] as in Fig. 3. Fractals are often used in generating large structures with little programmer strain. For the roads structure a simple fractal from Fig. 4 is adopted. The “main road” has an outgoing road, this one also has an outgoing road and so forth. One can imagine that there are more outgoing roads on both sides of the “current road”, starting in different points of the road. Not all angles in the structure are right angles. Angle randomization should be done because it is better for synthetic data to be complicated rather than based on a regular mesh – tests based on them can be more reliable. Furthermore cities (especially in Europe) have rarely grid-based street network (like in the USA). Urban areas are built near rivers and transportation routes and are impacted by the local topography. Urban planning is described in [1 and 12]. To simplify the calculation, it is assumed that first road crosses the whole map and (at least for now) the roads are exactly straight.

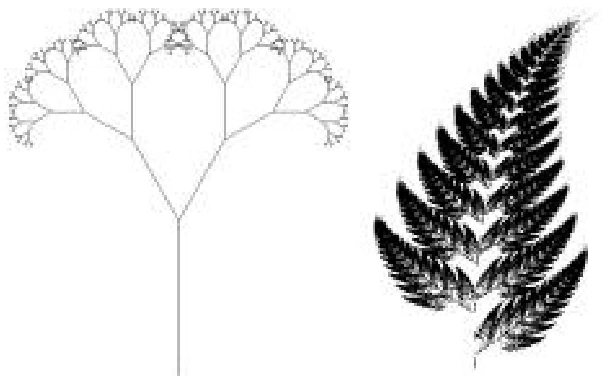


Fig. 3. Example fractals (taken from [13])
Rys. 3. Przykładowe fraktale (cytowane za [13])

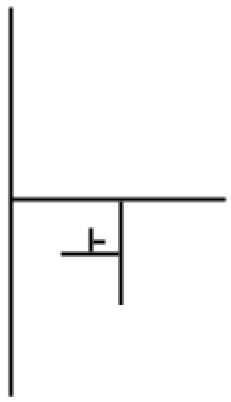


Fig. 4. Fractal used in roads structure generation
Rys. 4. Fraktal użyty do wygenerowania sieci dróg

The algorithm for creating streets geometries is a recursively expanding a data R-tree. This R-tree is a *spatial index* [14] used then for searching the near geometries for a newly added road. Roads geometries are polygons with two parallel sides with axes used to calculate the fractal. The roads layer should not have overlapping geometries so it is assumed that a road starting from its main road ends if one of the following conditions is fulfilled:

- the road reached the border of the generated area,
- the road reached a previously created road (or roads),
- the road achieved its maximal length and became a blind alley.

In case of the first two conditions special operations were implemented for polygons cutting:

- lines or vectors cross point calculations,
- cutting a polygon in two along a given line,

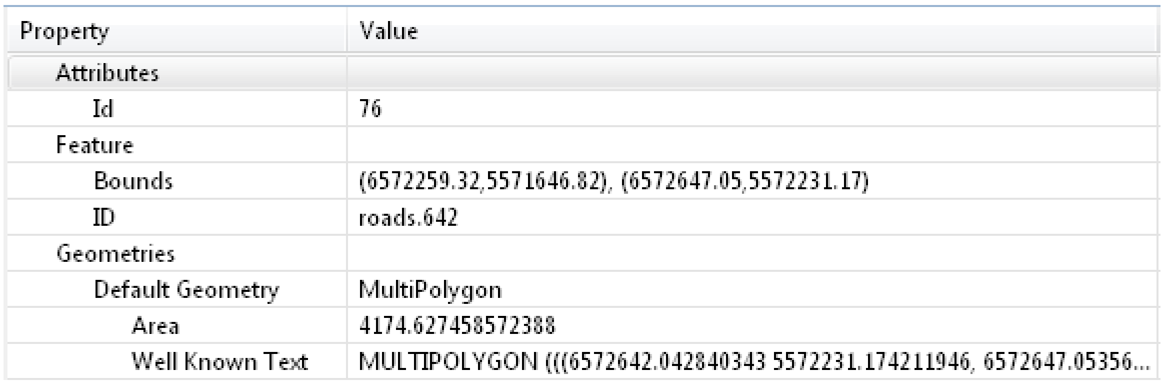
- triangulation of a polygon (finding whether a point lies inside of a polygon is much easier when this polygon is represented as a list of triangles),
- other additional operations.

There is a limit for number of outgoing roads against the length of the main road and minimal distance between two outgoing roads. Maximal length and width of newly placed roads gradually decrease with recursion level. Generation stop condition is reaching a road length for which there is no place for outgoing roads.

The position of roads, although randomized, should create more dense areas in the network, which will correspond to e.g. a city centre. City outskirts should have more “free space” between roads and only few roads should leave the city – transit routes.

3.2. Features Attributes

At the time being the geometry is more important than the attributes. For every feature only one attribute is created: an unique integer identifier. The actual tests of GIS systems might reveal what attributes will be required. Then, probably randomly generated and maybe fulfilling additional requirements, text or numeric values will be added. Fig. 5 shows a view of an example feature data. Category *Feature* is created in a shapefile automatically.



Property	Value
Attributes	
Id	76
Feature	
Bounds	(6572259.32,5571646.82), (6572647.05,5572231.17)
ID	roads.642
Geometries	
Default Geometry	MultiPolygon
Area	4174.627458572388
Well Known Text	MULTIPOLYGON (((6572642.042840343 5572231.174211946, 6572647.05356...

Fig. 5. Example road feature data (view from uDig)
Rys. 5. Przykładowe dane obiektu mapowego drogi (obraz z programu uDig)

4. Results

The following chapter shows closely the already implemented generator part. It describes user input and the application output. The program itself is written in Java using GeoTools library for spatial data support.

The user provides only basic information about the generated city:

- a EPSG CRS code to place the data on the globe – the appropriate CRS projection is then set in the resulting shapefiles,

- the two coordinates of lower left corner of the generated area,
- the width and height of the area (*z* values are not limited),
- a list of layers names and types (roads etc.) to store in shapefiles.

An example of run sequence to generate roads layer called *roads* might look like this:

```
java -jar ShapeGenerator.jar EPSG:2177 6560000 5570000 20000 10000 roads=ROADS
```

Created layer are validated which concerns mainly geometries overlapping. The main problem is floating point operations accuracy. The application uses double precision values and has a calculations error threshold of 0.000001 meters. This seems to be good enough since adjacent geometries even in extreme close-up do not create holes or overlaps between each other (Fig. 6). In the further development validation of a layer will take into account also other layers having influence on this layer shape.

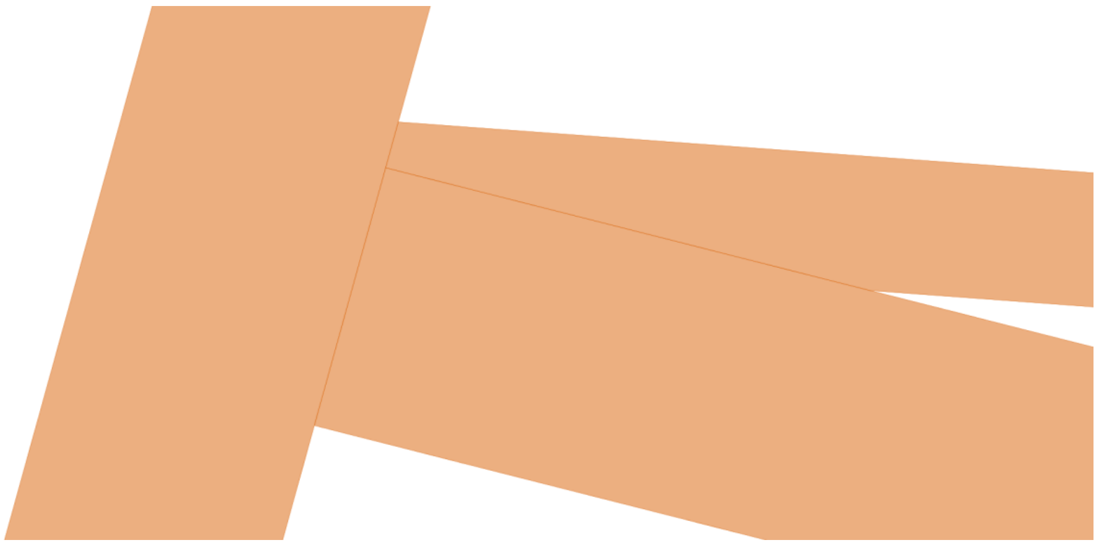


Fig. 6. Close-up of roads geometries in uDig (reported map scale is 1:102)
Rys. 6. Zbliżenie na geometrię dróg w aplikacji uDig (raportowana skala mapy to 1:102)

At this time only roads layer and (for debugging purposes) roads axes layer can be generated. Therefore, up to two shapefiles are written on the disk. Both contain two-dimensional geometries. Axes are lines placed "in the middle" of the roads. Roads are based on randomly generated axes and every place where e.g. axis is outside of a road means a calculation error. Fig. 777 shows an example of a generated roads structure on a rectangle area with size of 20 over 10 kilometres.

It is assumed that density of roads network grows in the middle of the map. This density depends on a value of a probability taken from a Gaussian bell function defined for the whole map with amplitude of 1 and variance equal to 35% of the map width or height depending on the axis. The threshold preventing to start a new road in a given point of a map is not a constant value. It is chosen randomly every time from a range 0.5÷0.75. This makes possible for the city to have irregular shape and density rather than something close to a circle or ellipse. The density function will impact further layers e.g. near centre buildings will be higher.

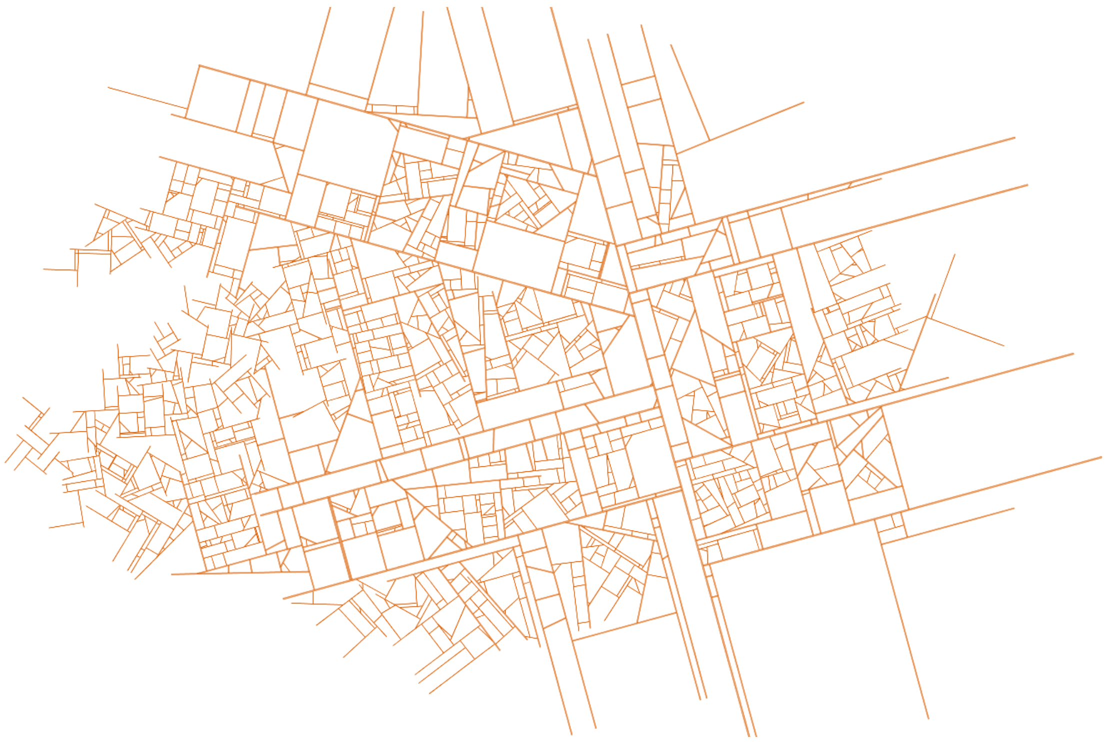


Fig. 7. Example roads structure with 1418 streets (view from uDig)
Rys. 7. Przykładowa sieć dróg z 1418 ulicami (obraz z programu uDig)

Smaller roads will be shorter and outgoing roads will be packed more densely. Street lane has 3 meters and the roads width corresponds to 6, 4 or 2 lanes. Outgoing roads must start at least 40 meters from each other (measured from generated roads start points as shown in Fig. 8) – this does not concern roads ends. A road cannot start in a place already occupied by another road. There is 80% probability that outgoing road angle will have 90°.

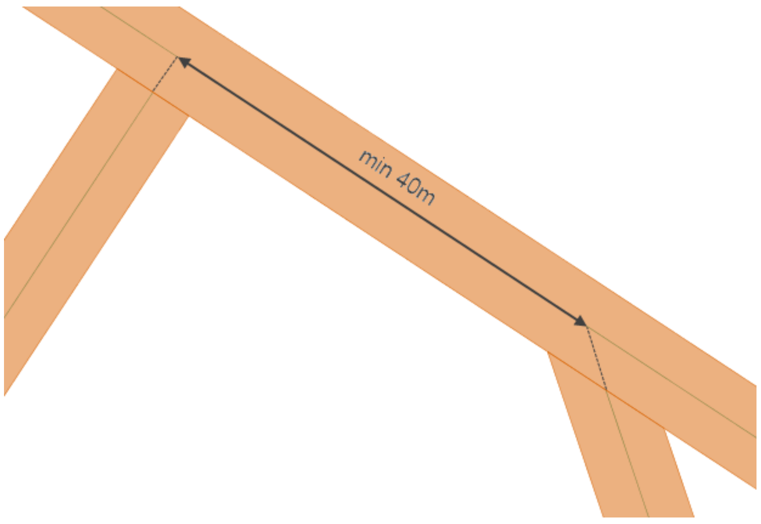


Fig. 8. Minimal outgoing roads gap
Rys. 8. Minimalny odstęp dróg wychodzących

5. Conclusions and Further Work

All implementations assumed for the first phase of generator development were completed. The roads structure has simple geometries, however the amount of features can be significant – even tens of thousands. Using this data in a GIS system (still developed or already working) might also find errors in the data format. This might be mutually beneficial.

The introduced prototype will be now undergoing extension to support other layers and eventually be able to create a full virtual city. Data coherence will become more important when more layer types will be available. Relation between layers will emerge and they will influence the way a layer is generated. A full check of spatial joins between layers and validation of obeying rules like "one building on one parcel" will be required.

During the future development the generator should also be used in an actual GIS system. Not only to check it but also to tweak the generator's behaviour. One of the tweaks might be to make roads with bends which might complicate axes implementation and cutting operation. Another will be to divide very long roads since one route can have different names in different parts. Assumptions concerning the implementation might become user provided parameters to personalize generated data. Also typical street crossing could be implemented like e.g. roundabouts.

The next phase for implementation is parcels layer. Here roads immediately impact the shape of the layer. A street is placed on a parcel or parcels so their shape will be determined by the street. After that pavements, residential parcels and space for natural topographical features (like rivers, ponds etc.) should be added. There are a lot of possibilities here which might also alter the way roads structure is represented.

Acknowledgements

This work was supported by the European Community from the European Social Fund.

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. Kubik T.: GIS. Rozwiązania sieciowe. PWN, Warszawa 2009.

3. Ferraggine V.E., Doorn J.H., Rivero L.C.: Handbook of Research on Innovations in Database Technologies and Applications: Current and Future Trends. Information Science Reference, Vol. II, 2009, p. 325÷333.
4. JTS Topology Suite Technical Specifications. Vivid Solutions, Version 1.3, 2006.
5. Carlson R.: Understanding Geospatial Data Files used in Geographic Information System Software, A Basic Introduction. Environmental and Remote Technologies Lab, Brown University.
6. ESRI Shapefile Technical Description. An ESRI White Paper, 1998.
7. 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.
8. http://rescuecore.sourceforge.net/ (access 2011-01-15).
9. http://sites.google.com/site/stargazersrpgstuff/news/rpgcitymapgenerator (access 2011-01-15).
10. Pokrajac D., Fiez T., Obradovic Z.: A Data Generator for Evaluating Spatial Issues in Precision Agriculture. Precision Agriculture, Vol. 3, No. 3, 2002, p. 259÷281.
11. Longley P.A., Batty M.: Using Fractal Geometry to Measure Maps and Simulate Cities. Computer Education, Vol. 56, 1987, p. 15÷19.
12. Gao H., Zhang H., Hu D., Tian R., Guo D.: Multi-Scale Features of Urban Planning Spatial Data. 18th International Conference on Geoinformatics, 2010.
13. Ibrahim M.M., Krawczyk R.J.: Generating Fractals Based on Spatial Organizations. Illinois Institute of Technology College of Architecture, 2001.
14. Guttman A.: R-trees. A Dynamic Index Structure for Spatial Searching. ACM SIGMOD Record, Vol. 14, Issue 2, 1984.

Recenzenci: Dr inż. Zbigniew Buchalski
Dr inż. Jacek Frączek

Wpłynęło do Redakcji 16 stycznia 2011 r.

Omówienie

Systemy informacji przestrzennej (Geographic Information Systems) projektowane są w celu udostępniania i zarządzania danymi geograficznymi. Ze względu na wymagania dotyczące wydajności i niezawodności takich systemów w środowisku produkcyjnym, konieczne

są rozbudowane testy. By były one wiarygodne, powinno się przeprowadzać je na prawdziwych danych, a najlepiej danych, na których system będzie docelowo pracował. Jest to trudne do osiągnięcia, zwłaszcza wtedy, gdy aplikacja jest zamawiana przez instytucję, dla której dane nie są jeszcze opracowane. Tworzenie danych przestrzennych, np. ze zdjęć lotniczych, jest czasochłonne i drogie. Dostęp do danych przygotowanych dla systemu bywa też ograniczany prawnie, z kolei te dostępne publicznie nie są dostatecznie dokładne. Rozwiązaniem może być generator testowych danych przestrzennych. Generator tworzy warstwy (*layers*) danych, składające się na mapę wirtualnego miasta.

Niniejszy artykuł przedstawia pierwszą fazę generacji zbiorów testowych, jaką jest generacja sieci dróg. Generowane warstwy zawierają obiekty mapowe (*features*), które składają się z geometrii i atrybutów opisowych. Warstwy te powinny być odpowiednio duże, by odpowiadać ilościom danych, jakie mogą pojawić się w docelowym środowisku. Powinny też jak najlepiej naśladować dane rzeczywiste (przykład na rys. 1). Parametrami wejściowymi, określanymi przez użytkownika, są jedynie podstawowe informacje, takie jak położenie miasta na kuli ziemskiej oraz lista warstw do wygenerowania.

Generacja warstwy ulic oparta jest na fraktalu (rys. 4), w którym z linii odpowiadającej drodze głównej wychodzą drogi podrzędne. Położenia dróg są do pewnego stopnia losowe, by zaburzyć wzór fraktala (rys. 7). Do indeksowania i wyszukiwania danych wykorzystane jest R-drzewo, będące indeksem przestrzennym. Wynik generacji zapisywany jest w pliku shapefile.

Rozbudowa aplikacji obejmie dodanie generacji warstwy działek, cieków wodnych, torów kolejowych, budynków, zieleni i prawdopodobnie trójwymiarowej warstwy terenu. Dodatkowo rozszerzony zostanie moduł walidacji warstw o badanie poprawności nałożenia warstw na siebie.

Address

Tomasz PŁUCIENNIK: Silesian University of Technology, Institute of Computer Science, Akademicka 16, 44-100 Gliwice, Poland, tomek.pluciennik@gmail.com.