GEOMATICS AND ENVIRONMENTAL ENGINEERING • Volume 9 • Number 3 • 2015

http://dx.doi.org/10.7494/geom.2015.9.3.35

Mateusz Ilba*

An Analysis and 3D Visualization of Shading of Urban Spatial Objects with the Use of the Phython Programming Language in the Blender Application**

1. Introduction to the Ray Tracing Algorithm

This algorithm is one of the first methods for the creation of photo-realistic 3D visualizations. It is based on specifying the color and brightness of a pixel seen by an observer / a camera on the screen. In order to achieve it, light vectors are used, which collect information on color and light output. The most computing power is used by the algorithm for calculating intersections of elements of 3D solids with light (vectors) [5]. It is for optimization that an initial location of the intersection is determined first and then its exact position. In Figure 1, the formation process of shadows and their presentation can be seen. It can be noticed that the visible pixels of the observer's image are subjected to the test of the light source visibility. The intersection of the plains of the 3D object with the vector, which is defined by the point of the light source and the visible point, which is tested for shading, is searched [3].

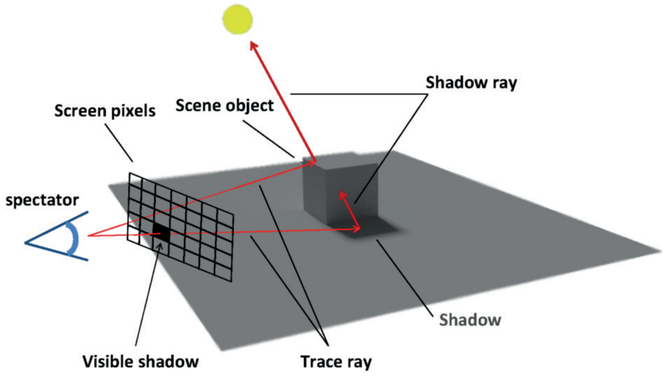

Fig. 1. Generating shadows in the ray tracing algorithm

* AGH University of Science and Technology, Faculty of Minining Surveying and Enviromental Engineering, Krakow, Poland

** The study was the carried out with the financial support from grant no. 15.11.150.324, AGH University of Science and Technology. Author is a beneficiary of Anna Pasek Foundation Scholarship VII edition

2. Saving Data on Lighting for Textures

The ray tracing algorithm is used to determine the shadows of the currently visible areas of the 3D image. In order to perform the shading analysis, it is not only the knowledge of the shading of the visible scene at the given time that is needed since we will not get information for all 3D elements in this way. We have to obtain data that show shading for all planes in the given scene.

It is in this case that the function of the shadow location recording for textures, which is put on the object using specific rules, is extremely important. The Blender software has such a possibility and the function responsible for this process is called "baking textures". Textures are bitmap images, which simulate details of surfaces of the 3D objects; among others their colors and texture. The rules determining putting the two-dimensional texture on a 3D object are called texture mapping and define how the image will be put on the object.

The UV mapping creates a link between the texture and the object on the basis of the distribution of all planes in the image. Specified planes may have a connection with the same pixels. While saving the shading, it is very practical to put each plane of the 3D model on a clean fragment of the image without overlapping them. It will provide uniqueness of the stored information on the shading of the object [12]. In Figure 2, we can see an example of the UV texture mapping.

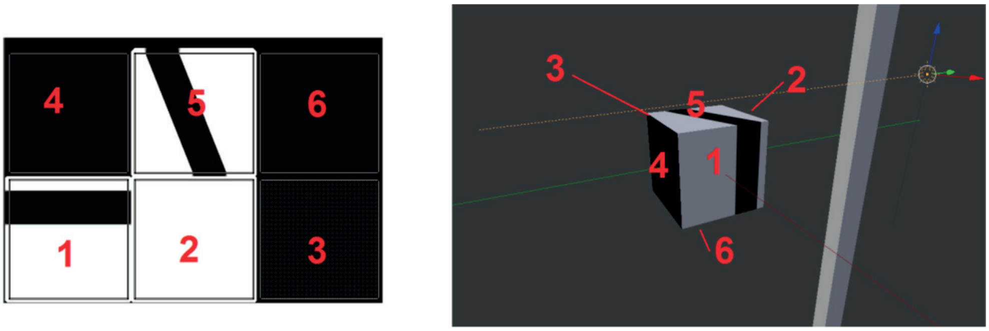

Fig. 2. Texture mapping diagram. An example based on generating shadows using the "bake" function in the Blender software

We can see that the areas are marked on the left side, in which the texture will be put on individual planes of the object. Additionally, there are small gaps among overlapping planes in the image. While combining the 2D image with the 3D object, the software transforms texture pixels and puts them in appropriate positions on the 3D model according to the mapping. Since we work with a cube, the image has 6 parts, which are put on the cube at appropriate places. It is in the figure that the walls 3 and 2 from the rear part of the object as well as wall 6 from the bottom are invisible.

If there is at least one source of light in a scene, which generates shadows, we have the possibility to generate shadow maps for 3D objects. In the Blender software, shading is saved in the form of an image with two values for the pixels: if there is no shadow, it will be white and if there is shadow, it will be black. In Figure 2 on the left, we can see an example of an image with information on the shading generated for the whole 3D object. A big advantage of the algorithm is taking into account all spatial elements in the 3D scene while generating shadows. It is in the example that there is a second object, which causes the appearance of shadows on the analyzed object.

Before starting "baking texture", an image has to be created, on which the 3D object will be mapped. The resolution of an image can be selected freely but it should be taken into account that the too high resolution will cause a substantial increase in the time of the algorithm's work and will increase the use of the computer memory. Additionally, an automatic distribution of the 3D object's planes on the texture through UV mapping has to be carried out [1]. The software analyzes the scene itself and we obtain a texture for the specified object as a result.

3. Application Scheme

It is not a problem to perform the operation of determining shading for a single object at one moment of time, so it can be done manually [2]. If there are more objects then the manual generation of shadows will be inefficient. Additionally, if we want to make classification for a whole day, we have to change the position of the sun from time to time and carry out the analysis all over again saving the previous results in a certain way. Therefore, there has to be written a script for the automation of the analysis, an application coordinating actions carried out by hand along with the performance classification.

The first step, while carrying out the analysis, is to load 3D objects. The Blender software environment supports a wide range of popular formats, in which three-dimensional objects are saved. Among them there are files: *.obj, *.dae, *.svg, *.vrl, *.3ds, which set the standard for 3D data exchange. Each program supporting three-dimensional data is able to save them at least one in the above-mentioned formats. While creating the algorithm, the author used objects saved in the format *.obj imported from the AutoCAD software and from the format *.vrl originating from the export of objects from the ArcScene software. When importing 3D data, the attention should be paid to the coordinate system. The Y-axis in the Blender software sets the north direction, the X-axis the east direction and the Z-axis shows the height. A change of any coordinate, while 3D objects are imported, will cause incorrect generation of shadows [9].

A flowchart for the work of the script is shown in Figure 3. It can be noticed that there are two main loops in the program: the loop of the object selection for analysis and the loop of the time selection.

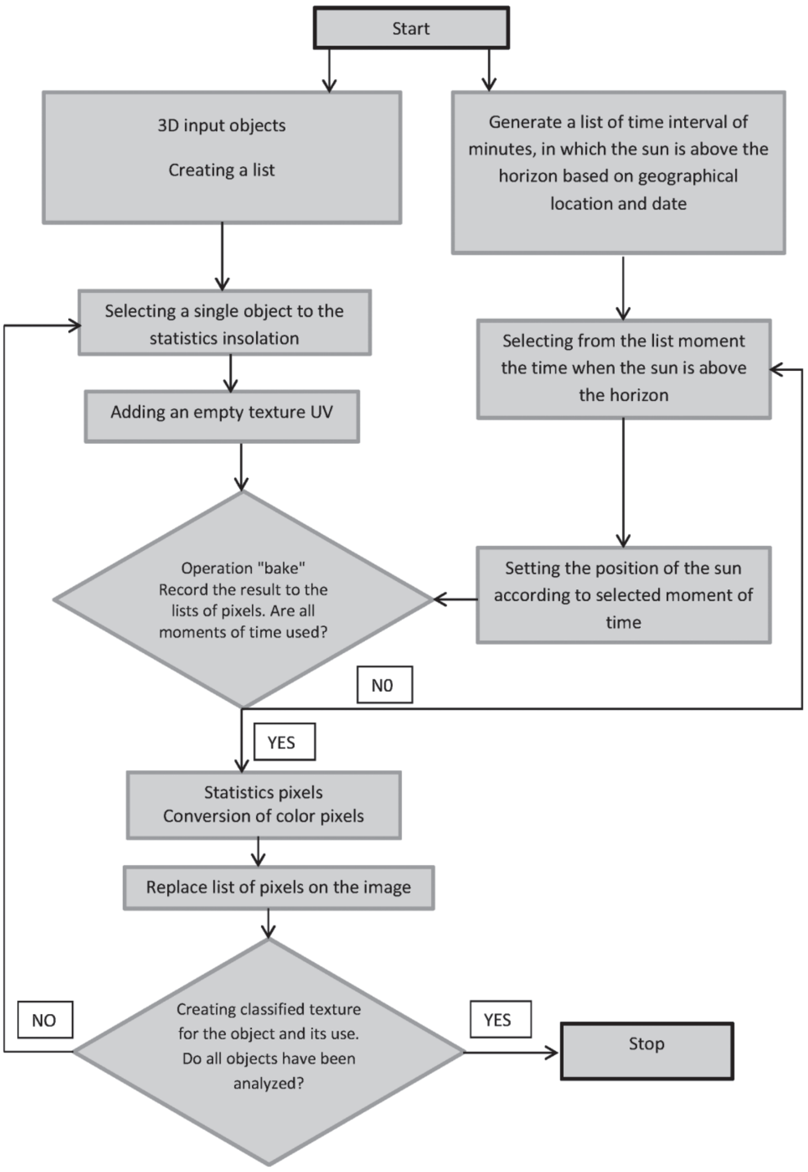
Start
3D input objects
Creating a list
Generate a list of time interval of minutes, in which the sun is above the horizon based on geographical location and date
Selecting a single object to the statistics insolation
Selecting from the list moment the time when the sun is above the horizon
Adding an empty texture UV
Operation "bake" Record the result to the lists of pixels. Are all moments of time used?
Setting the position of the sun according to selected moment of time
NO
YES
Statistics pixels Conversion of color pixels
Replace list of pixels on the image
NO
Creating classified texture for the object and its use. Do all objects have been analyzed?
YES
Stop

Fig. 3. Scheme of the algorithm shading daily

Each separate spatial object is analyzed in the order that has been loaded in the list of items. The division of the analysis into single elements is needed due to the use of the "baking textures" function for one object. Each "baking" has to be performed on a new texture, which is put on the spatial object. After "baking" the image is converted into a variable in the form of a list of image pixel values. This operation allows for carrying out any classification, among others of the time length of falling sunlight on the given item. The second loop causes generating shadows for every moment, in which the sun is above the horizon. These moments are saved in the form of a list and show following minutes from sunrise to sunset. Since they are given in minutes, our time list can be freely filtered; for example, we can select a point in time every 15 minutes in order to speed the work of the algorithm up.

Apart from the main algorithm, a model for determining the position of the sun for a certain time and certain geographic coordinates was developed. It is based on the operations contained in the NOAA ESRL formula presented in the Python language [4]. It is an exact, which takes into account among others refraction of sunlight passing through the atmosphere. The program also includes smaller algorithms that improve the performance of the entire application: among others the method for filtering the list, for conversion of a list into an image and vice versa, for preparation for "baking" shadows in the texture [2]. Additionally, the author has developed a formula for the transition from a numerical aggregate data of shading into the outcome texture presenting the shading time using any number of time intervals and colors.

4. Summary

In the era of the information society the demand for each type of information is huge, especially, when it is applied to us. Without any doubt information on shading is required by planners and architects but also by ordinary people, who are interested in sunlight falling onto particular buildings. The knowledge, if the sunlight will not be blocked by surrounding buildings or urban infrastructure is valuable and provides us with answers to many questions.

Without any doubt the presented way of carrying out an analysis of shading based on the use of the Blender 2.69 program, will allow for obtaining information on shading time. An absolute advantage of the application is its gratuitousness; blender is available under the GPL license so all scripts, including the ones written by the author, are distributed free of charge. It causes that the access to this application is extremely wide.

The author has tested the work of the script for six output resolutions of the texture: 10 × 10 pixels, 100 × 100 pixels, 200 × 200 pixels, 300 × 300 pixels, 400 × 400 pixels and 500 × 500 pixels. While executing the script, the generation of shadows was set with an interval of 15 minutes that means that the sun changed its position with a specified interval during the day. Additionally, the author examined, how the script behaved at different intervals between generated shadows. 1-minute, 5-minutes, 10-minutes, 15-minutes intervals were taken into account. In Figure 4 we can see a difference in the appearance of the scene after classification of the result, whereas Table 1 shows, how the time needed to generate shadows varies depending on the resolution of the resulting texture, Table 2 shows how the time needed to generate shadows varies depending on the time interval. We can notice that although the interval of the analysis does not significantly affect the generated results, but if the time interval is too big, and the setting of too small size of the texture will cause blurring of details.

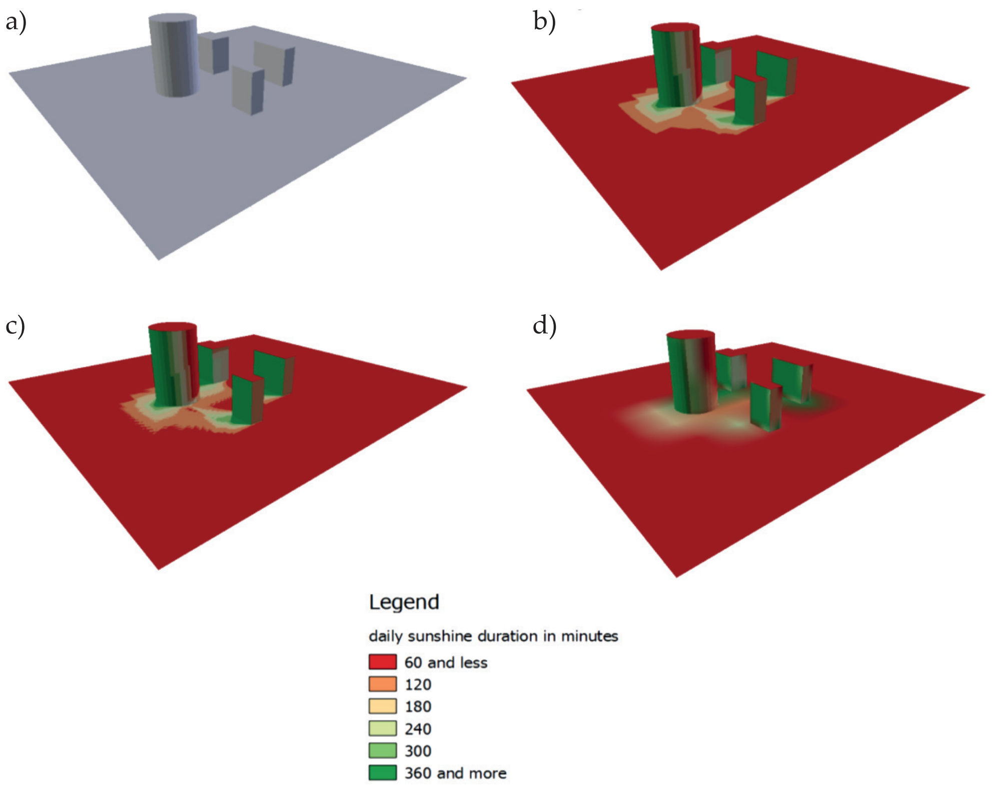

Fig. 4. Different settings generate shading: a) no shading analyses; b) resolution of the texture was set to 500 × 500 pixels, time interval 15 minutes; c) resolution of the texture was set to 500 × 500 pixels, time interval 1 minutes; d) resolution of the texture was set to 10 × 10 pixels, time interval 15 minutes

Table 1. Analysis time, generate shadows varies depending on the resolution of the resulting texture

No.	The number of texture pixels	Analysis time [s]
1	100	16
2	10,000	18
3	40,000	23
4	90,000	29
5	160,000	37
6	250,000	56

The longest work time of the script falls on the analysis of the pixel list, therefore by increasing the resolution, we increase the length of the list significantly. For the texture size of 10 × 10, the list has only 100 elements; when we increase the size to 50 × 50 pixels the list is increased to 2,500 elements; after increasing the size to 100 × 100 pixels the list grows to 10,000 elements. Probably, the speed of shading generation could be increased significantly by introducing operations on matrices. The author did not find a solution to operate on matrices in the environment of the blender software [8]. The latest version of Python 3.3.0 does not allow for proper use of the available Python libraries written for older versions [7].

Table 2. Analysis time, generate shadows varies depending on the time interval

Time interval [minutes]	Number of time moments during the analyzed day (e.g. December 21st)	Analysis time [s]
1	551	833
5	110	170
10	55	85
15	37	56

As a result of the shading analysis, we obtain visualization, with which we can easily distinguish areas on the objects that are shaded for a certain time. We can influence the pallet of colors and a specified number of intervals. We can mark any time interval that we are interested in, in which shadows appear, with any color. We have a possibility to save a 3D scene, which presents the results of the analysis, in any format and display it on any computer, even without the Blender software.

Besides normal tests for generating shading, the author tested the work of the script on real data, which were taken from a part of Manhattan, a district of New York. The data were imported as a *.shp file with the data on height of the buildings [10]. Moreover, the author compared the results with the commercial solutions of the Bentley Company in the form of the Microstation V8i application and the application Shadow Analysis for Google SketchUp. The results of this comparison can be seen in Figures 5–7.

Summing up, by combining the Python language with the Blender application, it is possible to carry out day- analysis for shading of a 3D city. It has advantages because there are no purchase costs for the person performing the analysis; there is a wide range of formats of input spatial data; the presentation of the analysis can be modified flexibly, the number of ranges and their colors can be set; the visualized scene can be exported to many formats. The disadvantage is the time of the analysis. The script is an effective method for automating shading analysis and for the evaluation of different options for shaping development in urban areas. The author wants to develop the application in order to improve the quality of the generated spatial analysis as well as to increase the efficiency of the algorithm.

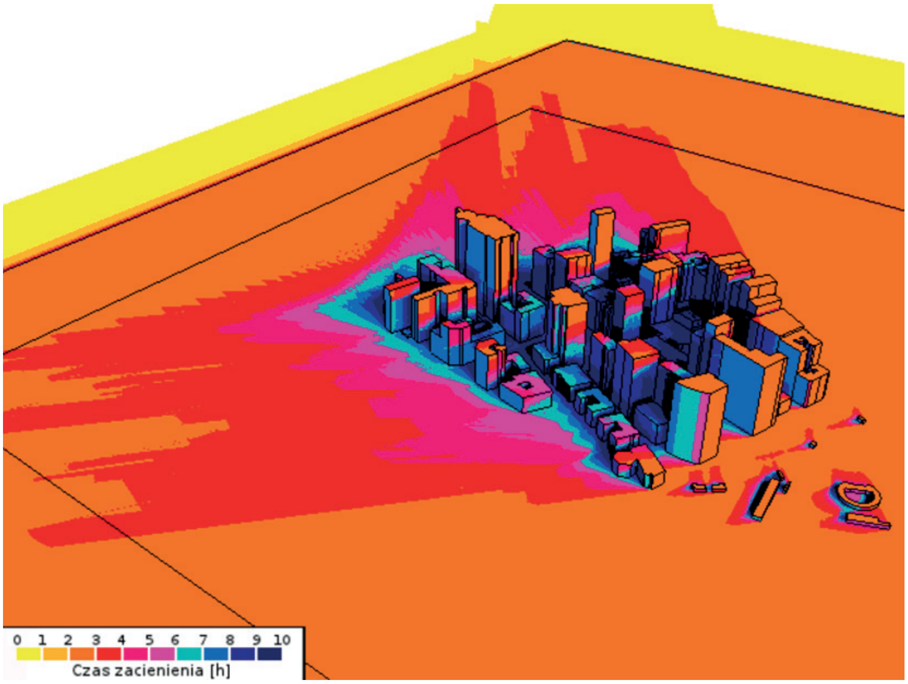

Fig. 5. Daily shading analysis.
Analysis was performed with the Shadow Analysis TM for Google SketchUp program
Source: [6]

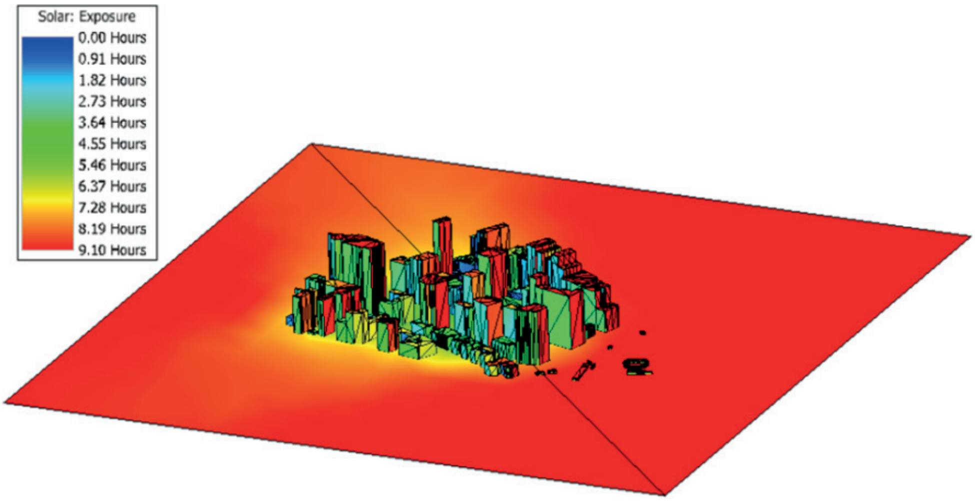

Fig. 6. Daily shading analysis.
Analysis was performed with the Bentley Microstation V8i TM application

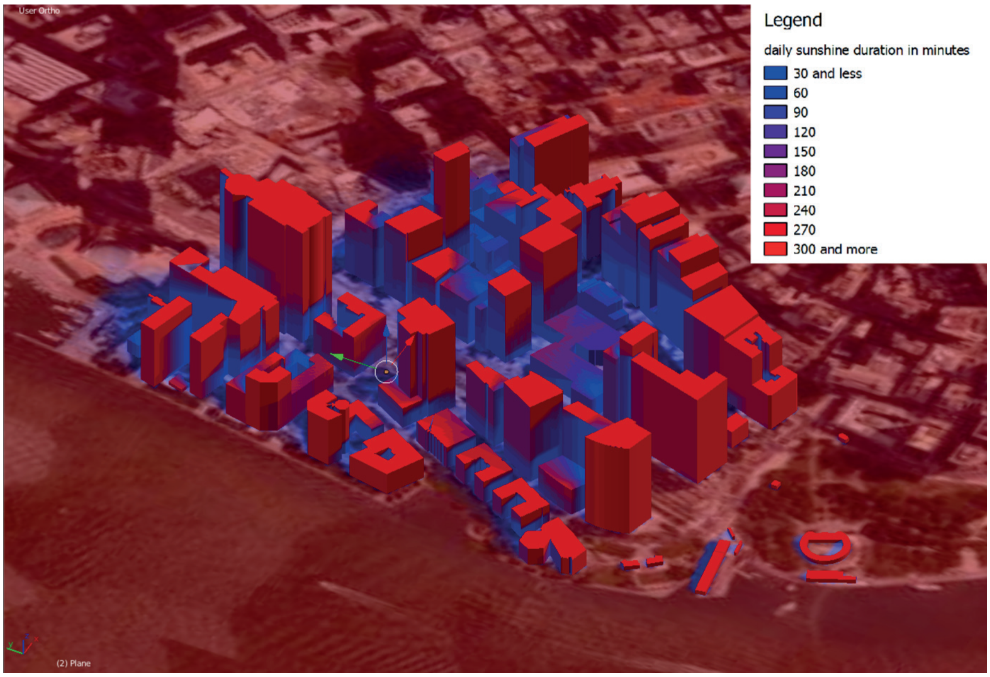

Fig. 7. Daily shading analysis. The analysis was performed in the algorithm of the author. In addition, added photorealistic ground texture. The color palette of visualization has been defined by the author

References

[1] Arłukowicz P.: *Polski kurs Blendera. Funkcja Bake*. [on-line:] http://polskikurs-blendera.pl/ [access: 15.12.2014].
[2] Blender: *Blender manual*. [on-line:] http://wiki.blender.org/index.php/Doc:2.6/Manual [access: 18.01.2015].
[3] Cook R., Porter T., Carpenter L.: *Distributed Ray Tracing*. Computer Graphics, vol. 18, no. 3, 1984, pp. 137–145.
[4] Earth System Research Laboratory: *Solar Calculation Details*. [on-line:] http://www.esrl.noaa.gov/gmd/grad/solcalc/calcdetails.html [access: 10.01.2015].
[5] Institute for Communication Technologies and Embedded Systems (ICE): *Ray-Tracing*. [on-line:] http://www.ice.rwth-aachen.de/research/tools-projects/grace/ray-traycing/ [access: 20.12.2014].
[6] Janik T.: *Architecture Design and Computation. Shadow Analysis*. [on-line:] https://tomaszjaniak.wordpress.com/plugins-2/ [access: 01.12.2014].
[7] Langtangen H.P.: *Python scripting for computational science*. Springer, Berlin 2004.

[8] McKinney W..: *Python for Data Analysis*. O'Reilly Media, 2013.
[9] Mullen T.: *Blender: mistrzowskie animacje 3D*. Wydawnictwo Helion, Gliwice 2010.
[10] NYC Open Data, [on-line:] http://www.data.cityofnewyork.us [access: 01.12.2014].
[11] Twarowski M.: *Słońce w architekturze*. Arkady, Warszawa 1996.
[12] Xuewen L., Qiuhai H., Yan Z.: *Texture Baking Techniques on Construction of Virtual Reality Interactive Scenes*. Applied Mechanics and Materials, vol. 55–57, 2011, pp. 478–483.