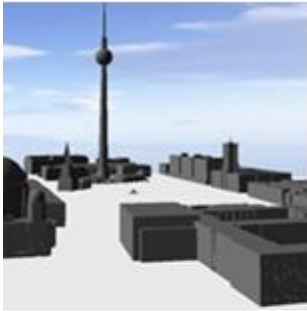


3D City Database for CityGML

Texture Atlas Creator

Standalone Tool and Java API

30 June 2011



**Institute for Geodesy and Geoinformation Science
Technische Universität Berlin**

Babak Naderi

Javier Herreruella

Claus Nagel

Thomas H. Kolbe

(Page intentionally left blank)

Content

1	DISCLAIMER.....	4
2	TEXTURE ATLAS CREATOR.....	5
	2.1 <i>Definition.....</i>	5
	2.2 <i>How to Use It.....</i>	5
	2.2.1 Main Features	7
3	REQUIREMENTS	7
4	APPENDIXES.....	8
	<i>Appendix A: Example.....</i>	8
	<i>Appendix B: Sample Texture Atlases</i>	11
5	REFERENCES.....	14

1 DISCLAIMER

The Texture Atlas Creator tool and API developed by the Institute for Geodesy and Geoinformation Science (IGG) at the Technische Universität Berlin is free software under the GNU Lesser General Public License Version 3.0. See the file LICENSE shipped together with the software for more details. For a copy of the GNU Lesser General Public License see the files COPYING and COPYING.LESSER or visit <http://www.gnu.org/licenses/>.

THE SOFTWARE IS PROVIDED BY IGG "AS IS" AND "WITH ALL FAULTS." IGG MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE QUALITY, SAFETY OR SUITABILITY OF THE SOFTWARE, EITHER EXPRESSED OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

IGG MAKES NO REPRESENTATIONS OR WARRANTIES AS TO THE TRUTH, ACCURACY OR COMPLETENESS OF ANY STATEMENTS, INFORMATION OR MATERIALS CONCERNING THE SOFTWARE THAT IS CONTAINED ON AND WITHIN ANY OF THE WEBSITES OWNED AND OPERATED BY IGG.

IN NO EVENT WILL IGG BE LIABLE FOR ANY INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES HOWEVER THEY MAY ARISE AND EVEN IF IGG HAVE BEEN PREVIOUSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

2 Texture Atlas Creator

2.1 Definition

The Texture Atlas Creator is a standalone tool and API for modifying CityGML files and making texture atlases for CityGML buildings. A texture atlas is a large image that contains many smaller sub-images, each of them being a texture for some part of a building object. The Texture Atlas Creator will automatically pack several (as many as possible) texture images of a building together into one or several atlases (depending on atlas' size settings) and adapt texture coordinates accordingly. As a result, the overall size of the resources (both CityGML and texture image files) will be significantly decreased. In addition, using texture atlases will considerably decrease the loading time of modified CityGML files in viewers. For more information about textures in CityGML specification, please see [1].

2.2 How to Use It

The Texture Atlas Creator program must be executed from the shell. It does not have a graphical user interface. The command line syntax is:

```
java [options] -jar tac.jar [/A <algorithm>] [/W <max_width> /H  
<max_height>] <input> <output>
```

A batch file for Windows systems is included. The command line syntax of this batch file is:

```
tac [/A <algorithm>] [/W <max_width> /H <max_height>] <input>  
<output>
```

In the following lines the command's options are explained:

- *<algorithm>* (*optional*): points to the packing algorithm that should be used during atlas creation. The selected algorithm will affect the position of each texture in the atlas(es) during the packing process. Default algorithm is set to be TPIM when not specified. Following algorithms are available in the Texture Atlas Creator program:
 - *TPIM*: It is a customized version of the Touching Perimeter algorithm as a heuristic two-dimensional bin packing supporting rotation of textures. During tests 83.35% of the resulting atlas surface was occupied. This algorithm is based on source code developed and released to the public by Jukka Jylänki.

TPIM sorts items according to descending area values. It initializes a bin with a maximum acceptable size and packs one item at a time. In the case that it is not possible to add a new item to the current bin, a new bin will be initialized. The first item packed in a bin is always placed in the bottom left corner. However in the resulting atlas the origin will be in the top left corner. Each item is

packed in a way that its bottom and left edges are touching either the bin or the edge of another item.

Each potential position for the new item will be scored as the amount of its touching edges. Touching the bin edges is more valuable in order to avoid inhomogeneous shape of bin. For each candidate (position) the score will be calculated twice (normal orientation and 90 degree rotated) and the highest value will be taken. For more information about Touching Perimeter algorithm please refer to [2].

- *TPIM_WOR*: *TPIM_WITHOUT_ROTATION* is an extension of *TPIM* algorithm which does not rotate textures.
- *SLEA*: Sleator's algorithm is a two-dimensional bin packing. It starts by packing all the items wider than half of the bin's width on top of each other. Then the remaining items will be sorted according to descending height values. The bin will be split in two halves. Next items will be placed from left to right in the left half and when it cannot accommodate further items, the left half will be closed and items will be placed in the right half. Next level of halves will be started with different bottom height according to the maximum height in previous level of left and right side. For more information please refer to [3].
- *NFDH*: The Next-Fit Decreasing Height (NFDH) algorithm starts by sorting items according to descending height values. Then it packs the next item, left justified, on the current level if it fits. Otherwise, the level will be closed, and a new level will be created (as a horizontal line drawn on the top of the tallest item packed on the current level), then the item is packed, left justified, on it [4].
- *FFDH*: The First-Fit Decreasing Height (FFDH) algorithm starts by sorting items according to descending height values. It packs the next item, left justified, on the first level where it fits. If there is not any level to accommodate it, a new level is created as in NFDH [4].
- *<max_width>* (*optional*) specifies the maximum acceptable width of output texture atlases. Default value is 2048. In case that this field is not specified in shell command the default value will be used.
- *<max_height>* (*optional*) specifies the maximum acceptable height of output texture atlases. Default value is 2048. In case that this field is not specified in shell command the default value will be used.
- *<input>* points to an input CityGML file or a folder. In the case of using folder, all sub folders will be parsed recursively and all founded CityGML files will be modified.
- *<output>* points to an output CityGML file or a folder. In the case of using a folder for input, the folder structure will be reproduced in the output folder.

2.2.1 Main Features

In the following, the main features of the Texture Atlas Creator are listed:

- Program will group textures having similar properties for each building and then create one or more texture atlases from each group. Transparency value of the images is one of the factors being considered in grouping textures.
- Maximum atlas size can be specified as an input argument. The program will guaranty that the generated texture atlases have a size equal or less than the specified argument. In the case that a texture's size is larger than the maximum atlas size, the texture will be resized to follow the condition of maximum atlas size.
- A folder may be specified as an input. As a result it will be parsed recursively and program will automatically run on all valid CityGML files founded in sub-folders.
- Detailed output will be shown in console.
- The SGI RGB Image format is supported by this program. An encoder is developed based on file format specification version 1.00 written by Paul Haeberli from Silicon Graphics Computer Systems. The encoder supports most of RGB images, but not all of them. For more information about the file format please refer to [5]. In addition to RGB, more image formats are also supported for input like JPEG and PNG.
- Texture atlases will be created in two image format depending on the transparency value of entire images (JPEG format for nontransparent and PNG format for images including transparent pixels). If the input CityGML file refers to a texture file whose format is not understood by the Texture Atlas Creator, the file will be copied as is in the corresponding place of the output folder.

Notes:

- Textures with wrapping coordinates (have a coordinate outside the range [0,1]) will not be added to texture atlases. The corresponding image file will be copied as is and its texture coordinates will not be changed.
- `TexCoordGen` and `GeoreferencedTexture` are not supported.
- Input CityGML files MUST be well-formed and validate based on the official schema.

3 REQUIREMENTS

Java JRE or JDK version 1.5 or later is needed. In case of using the batch file for executing the program 1GB main memory is needed. If that amount of memory is not available use the pure java syntax (first method) or modify the batch file by changing `-Xms1024m` and `-Xmx1024m` to the maximum possible size depending on your system.

4 APPENDIXES

Appendix A: Example

Here is a step-by-step guide for running the Texture Atlas Creator.

- Open a shell window and write the following command:

```
tac /A TPIM /W 2048 /H 2048 c:/test_case c:/results
```

- Figure 1 is a shell snapshot of the Texture Atlas Creator running. At first the program will recursively parse the input folder (path: “c:/test_case”) to find all available CityGML files. Then it will start modifying them one by one.

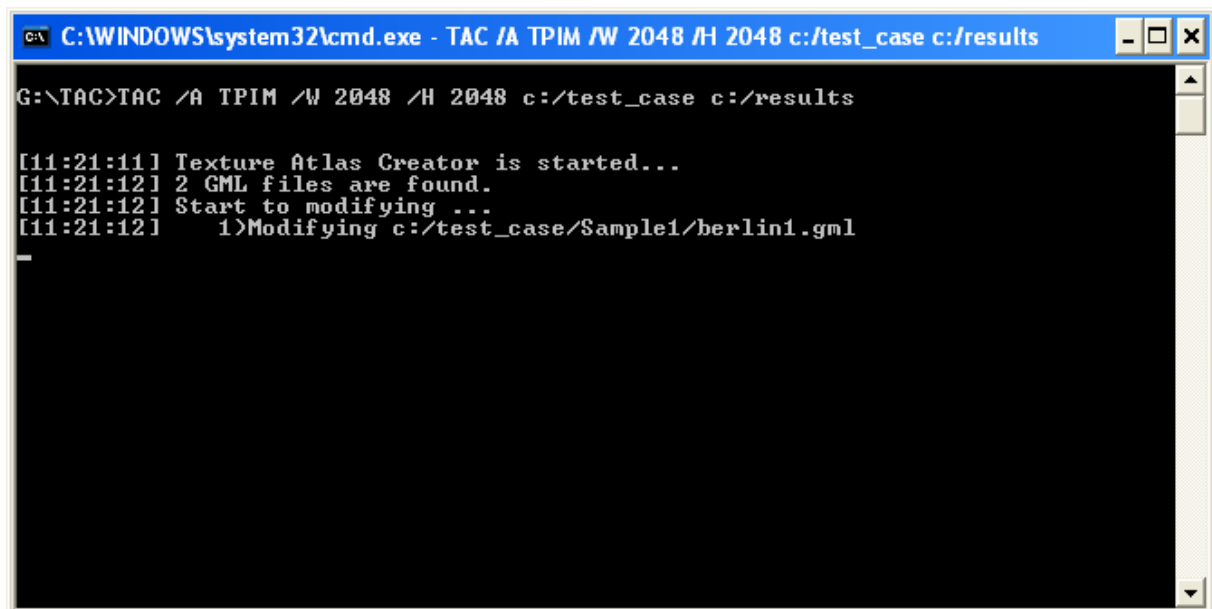


Figure 1: Shell snapshot of the Texture Atlas Creator running.

- Figure 2 shows a snapshot of the next step. The program is modifying buildings included in the first CityGML file one by one. As a result one or more texture atlases will be created for each building.
- Figure 3 shows a snapshot of the console when some textures with wrapping coordinates are detected. In this case the program will show an information message on the console and skip the texture.
- Figure 3 also shows the console message when modification of the first CityGML file is finished. The modified file is written into the output folder and the same procedure is started for the next CityGML file in the input directory.


```

C:\WINDOWS\system32\cmd.exe - TAC /A TPIM /W 2048 /H 2048 c:/test_case c:/results

G:\TAC>TAC /A TPIM /W 2048 /H 2048 c:/test_case c:/results

[11:21:11] Texture Atlas Creator is started...
[11:21:12] 2 GML files are found.
[11:21:12] Start to modifying ...
[11:21:12] 1>Modifying c:/test_case/Sample1/berlin1.gml
[11:21:18] Contains 19 building(s).
[11:22:58] Working on 1st building.<BLDG_0003000000f521db>
[11:23:00] Working on 2nd building.<BLDG_00030000f0028da8a>
[11:23:02] Working on 3rd building.<BLDG_00030000f0025072f>
[11:23:03] Working on 4th building.<BLDG_00030000a001ce4b3>

```

Figure 2: Shell snapshot of the Texture Atlas Creator running. During modification of each CityGML file, buildings will be modified one by one and one or more texture atlases will be created for each of them.

```

C:\WINDOWS\system32\cmd.exe - TAC /A TPIM /W 2048 /H 2048 c:/test_case c:/results

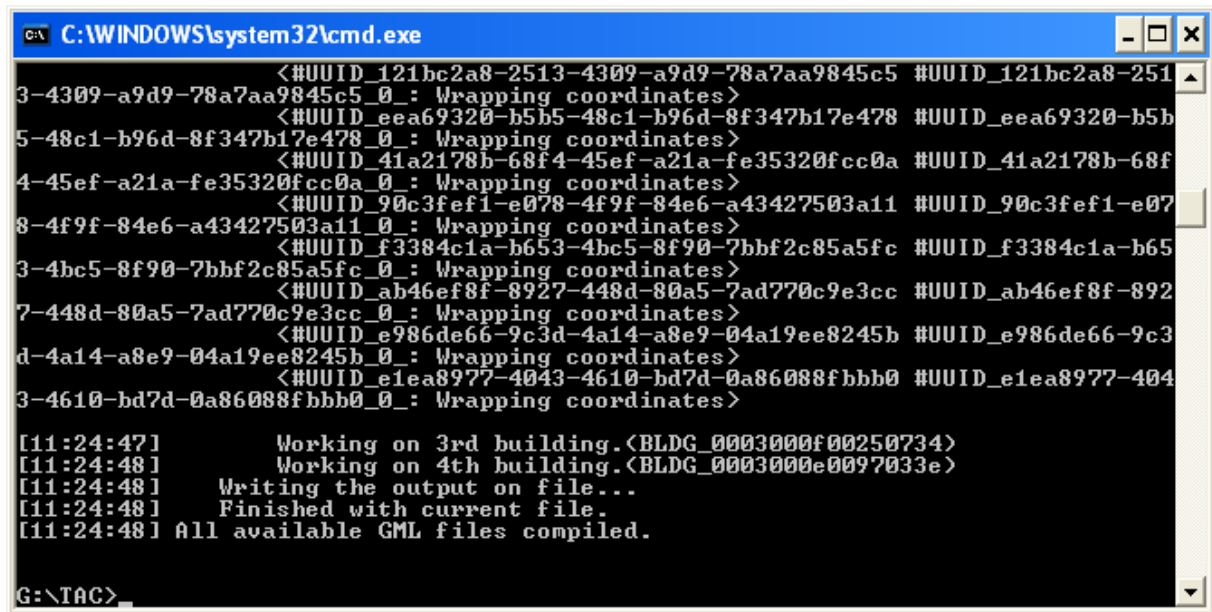
<#UUID_f3384c1a-b653-4bc5-8f90-7bbf2c85a5fc #UUID_f3384c1a-b653-4bc5-8f90-7bbf2c85a5fc_0_: Wrapping coordinates>
<#UUID_ab46ef8f-8927-448d-80a5-7ad770c9e3cc #UUID_ab46ef8f-8927-448d-80a5-7ad770c9e3cc_0_: Wrapping coordinates>
<#UUID_e986de66-9c3d-4a14-a8e9-04a19ee8245b #UUID_e986de66-9c3d-4a14-a8e9-04a19ee8245b_0_: Wrapping coordinates>
<#UUID_e1ea8977-4043-4610-bd7d-0a86088fbbb0 #UUID_e1ea8977-4043-4610-bd7d-0a86088fbbb0_0_: Wrapping coordinates>

[11:23:38] Working on 16th building.<BLDG_00030000e00858137>
[11:23:39] Working on 17th building.<BLDG_00030000a0011778e>
[11:23:39] Working on 18th building.<BLDG_00030000f00093e91>
[11:23:40] Working on 19th building.<BLDG_0003000000f51d29>
[11:23:40] Writing the output on file...
[11:23:41] Finished with current file.
[11:23:41] 2>Modifying c:/test_case/Sample2/berlin2.gml

```

Figure 3: Shell snapshot of the Texture Atlas Creator running. If any wrapping coordinates are found, the corresponding message will be shown in the console. After finishing the modification of all buildings included in the first file, the modified CityGML file will be written into the output folder. Then same procedure will be started for the next CityGML file.

- After modifying the second CityGML file the program is finished since all files present in the input folder were processed (Figure 4). When looking into the output folder, the same structure as in the input folder will be found. Textures are replaced with texture atlases and CityGML files are modified.



```

C:\WINDOWS\system32\cmd.exe
<#UUID_121bc2a8-2513-4309-a9d9-78a7aa9845c5 #UUID_121bc2a8-251
3-4309-a9d9-78a7aa9845c5_0_: Wrapping coordinates>
<#UUID_eea69320-b5b5-48c1-b96d-8f347b17e478 #UUID_eea69320-b5b
5-48c1-b96d-8f347b17e478_0_: Wrapping coordinates>
<#UUID_41a2178b-68f4-45ef-a21a-fe35320fcc0a #UUID_41a2178b-68f
4-45ef-a21a-fe35320fcc0a_0_: Wrapping coordinates>
<#UUID_90c3fef1-e078-4f9f-84e6-a43427503a11 #UUID_90c3fef1-e07
8-4f9f-84e6-a43427503a11_0_: Wrapping coordinates>
<#UUID_f3384c1a-b653-4bc5-8f90-7bbf2c85a5fc #UUID_f3384c1a-b65
3-4bc5-8f90-7bbf2c85a5fc_0_: Wrapping coordinates>
<#UUID_ab46ef8f-8927-448d-80a5-7ad770c9e3cc #UUID_ab46ef8f-892
7-448d-80a5-7ad770c9e3cc_0_: Wrapping coordinates>
<#UUID_e986de66-9c3d-4a14-a8e9-04a19ee8245b #UUID_e986de66-9c3
d-4a14-a8e9-04a19ee8245b_0_: Wrapping coordinates>
<#UUID_e1ea8977-4043-4610-bd7d-0a86088fbbb0 #UUID_e1ea8977-404
3-4610-bd7d-0a86088fbbb0_0_: Wrapping coordinates>

[11:24:47]      Working on 3rd building.<BLDG_0003000f00250734>
[11:24:48]      Working on 4th building.<BLDG_0003000e0097033e>
[11:24:48]      Writing the output on file...
[11:24:48]      Finished with current file.
[11:24:48] All available GML files compiled.

G:\TAC>

```

Figure 4: Shell snapshot of the Texture Atlas Creator running. The program is finished.

Appendix B: Sample Texture Atlases

In the following, sample texture atlases generated by different algorithms are shown:

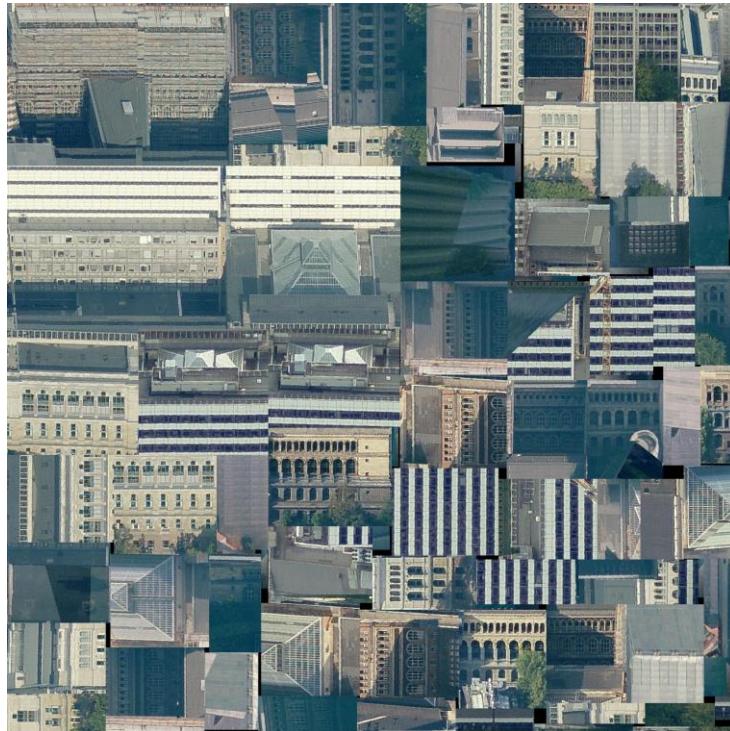


Figure 5 A texture atlas created by TPIM algorithm.

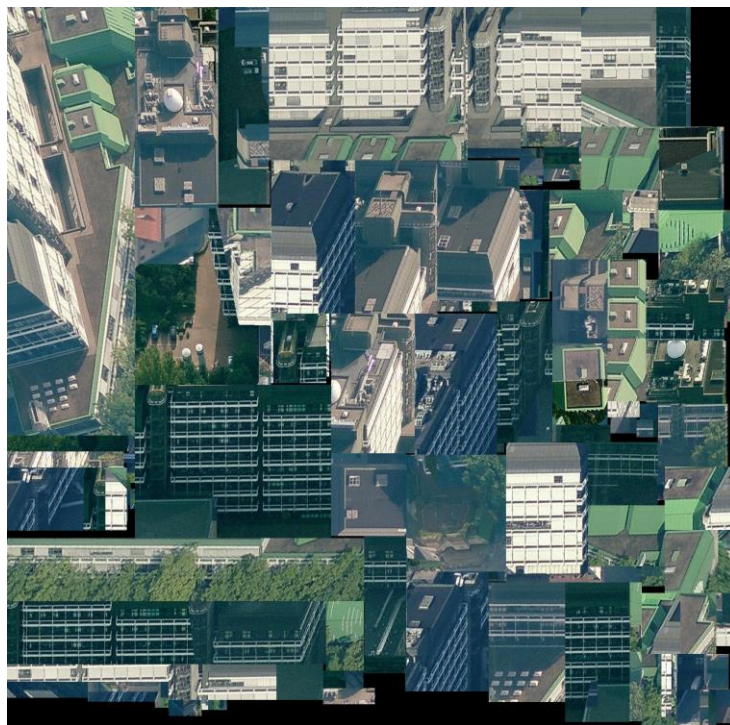


Figure 6 A texture atlas created by TPIM_WOR algorithm.

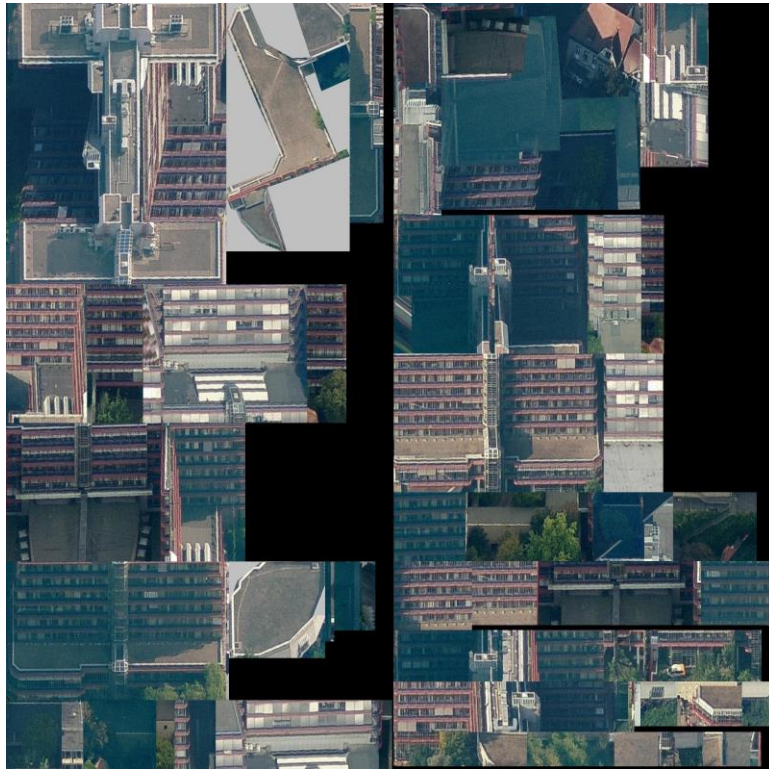


Figure 7 A texture atlas created by SLEA algorithm.



Figure 8 A texture atlas created by NFDH algorithm.



Figure 9 A texture atlas created by FFDH algorithm.

5 REFERENCES

- [1] Kolbe, T. H.; König, G.; Nagel, C.; Stadler, A. (2009): *3D-Geo-Database for CityGML*, Version 2.0.1, Documentation, April 24th. <http://www.3dcitydb.net>
- [2] Lodi A, Martello S, Vigo D (1999): *Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems*. INFORMS J on Computing;11:345-357.
- [3] D. Sleator (1980): *A 2.5 times optimal algorithm for packing in two dimensions*. Information Processing Letters 10, pp. 37-40.
- [4] Lodi, Andrea, Martello, Silvano and Monaci, Michele (2002): *Two-dimensional packing problems: A survey*, European Journal of Operational Research, 141, issue 2, p. 241-252.
- [5] The SGI RGB Image format, Weblink (accessed February 2012): <http://paulbourke.net/dataformats/sgirgb/sgiversion.html>