Eurographics Workshop on Urban Data Modelling and Visualisation (2015)
F. Biljecki and V. Tourre (Editors)

Far-LoD: Level of Detail for Massive Sky View Factor Calculations in Large Cities

D. Muñoz[1†], B. Beckers[2‡], G. Besuievsky[1§] and G. Patow[1¶]

[1]ViRVIG-UdG
[2]UTC-Compiegne

Abstract

In many applications, such as in urban physical simulations or in the study of the effect of the solar impact at different scales, models with different levels of detail are required. In this paper we propose an efficient system for quickly computing the Sky View Factor (SVF) for any point inside a large city. To do that, we embed the city into a regular grid, and for each cell we select a subset of the geometry consisting of a square area centered on the cell and including it. Then, we remove the selected geometry from the city model and we project the rest onto a panoramic image (in our case, the sides of a box). Later, when several SVF evaluations are required, we only need to determine the cell that the evaluation point belongs to, and compute the SVF with the cell's geometry plus the environment map. To test our system, we perform several evaluations inside a cell's area, and compare the results with the ground truth SVF evaluation. Our results show the feasibility of the method and its advantages when used for a large set of computations. We show that our tool provides a way to handle the complexity of urban scale models, and specifically to study the sensitivity of the geometry.

1. Introduction

The Sky View Factor (SVF) and similar measures are of crucial importance in urban planning, architecture and related fields. Traditional measuring techniques involve projecting the surrounding building geometries onto a half-sphere located at the measuring point, and computing the ratio of blocked vs. unblocked parts of the sky.

However, its computation is particularly challenging when evaluated inside a complex urban landscape, where distant skyscrapers can block substantial part of the incoming daylight illumination. See Figure 1. Including this geometry, however important it is, may result in an explosion of complex evaluations that can render forbidding computational times if several evaluations are requested (e.g., for averages over whole facades).

This paper introduces a technique for simplifying the computations of the SVF in large cities by simplifying the evaluation of distant geometry. For this, the city is partitioned using a regular grid and a set of localized *environment maps*, each associated to one cell, is used. In consequence, the evaluation of a SVF inside a grid cell simply amounts to the union of the evaluations of the occlusions by the geometric part inside the cell and the occlusions by the buildings represented in the environment map, thus considerably reducing computational evaluation costs.

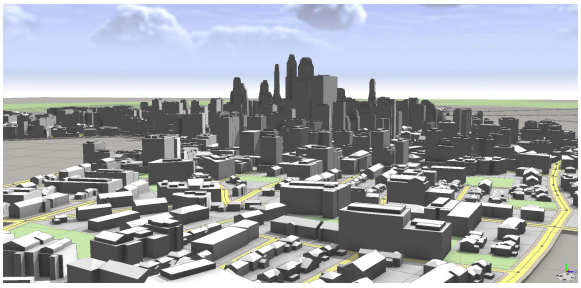

Figure 1: *An example of a complex urban landscape where computations of the SVF can be complex and time consuming.*

† davidmunoznovoa@gmail.com
‡ benoit.beckers@utc.fr
§ gonzalo@imae.udg.edu
¶ dagush@imae.udg.edu

© The Eurographics Association 2015.

DOI: 10.2312/udmv.20151341

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

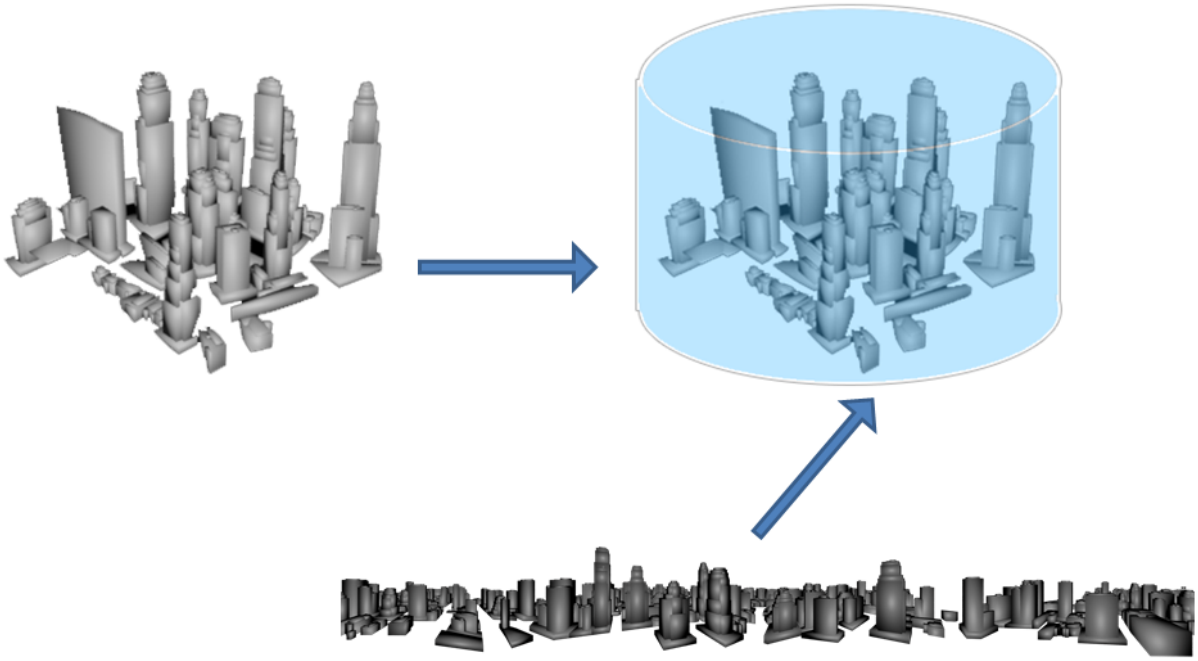

Figure 2: *Overview of our method. Given a cell in a city, we encode its surroundings in an environment map and compute the SVF from the geometry in the cell plus its corresponding environment map.*

2. Previous Work

Previous work on level of detail for urban models can be found in the area of urban generalization, like the cartographic generalization proposed by Anders [And05], or the face collapse from known constructive structures as walls and roofs [RCT*06]. The CityGML standard [Kol09], proposes the definition and usage of five different LoD levels, but does not provide a mechanism to generate them, nor an adaptive LoD scheme.

For procedural modeling, Parish and Müller [PM01] presented an initial proposal intended for city generation based on the L-system recursive nature. Automatic LoD-generation is obtained by starting from the building envelope as axiom, and the output of each rule iteration represents a refining step in the building generation. Although it is simple and automatic, this approach does not provide control on geometric building details. Through a similar approach, in the CityEngine system [Esr14], LoDs can be added manually in the grammar-rules by using a switch-case scheme for controlling the insertion of the geometry. Recently, new approaches were proposed to integrate LoDs mechanism in the procedural processing. In [BP13a], a rewriting method of the rulesets for the buildings has been developed for further replacing the geometric operators, which produced the right level of detail for each asset according to some user-defined criteria. In [BP13b], the authors proposed a highest level of detail by enabling selection, from entire buildings up to whole blocks, for geometric reduction. Later, Besuievsky et al. [BBBP14] presented a configurable LoD technique intended for daylight simulation, specifically tailored at procedural urban models. These works focus more on solving rendering problems, whereas in our approach we target more on the model preparation for simulation analysis. An example of such simulation was presented in 2014, when Roure et al. [RBP14] presented a method for computing hierarchical radiosity for procedurally modeled urban environments.

The technique presented in this paper is deeply related with the efforts done in the computer Graphics field with environment maps based on the early works by Greene [Gre86]. One of the first extensions was presented by Shade et al. [SLS*96], where a hierarchical image caching technique was used for accelerated walkthroughs of complex environments. Later, Décoret et al. [DSSD99] accelerated rendering computations by the use of multi-layered impostors. Then, Jeschke et al. [JW02, JWS02] introduced the use of layered environment-map impostors, used for navigating arbitrary scenes, and Eisemann and Décoret [ED07] presented an accurate analysis on exact error bounds for view-dependent simplification while interactively navigating arbitrary scenes. Finally, Umenhoffer et al. [UPSK08] used layered environment maps to compute robust multiple specular reflections and refractions using the capabilities of modern GPUs.

© The Eurographics Association 2015.

The sky view factor is widely used as an important parameter in modeling thermal phenomena, such as the urban heat island [Ung04]. In addition to the important use in all aspects of urban climatology, also plays a crucial role in forest climatology [HPE01] and human biometeorology [LMH10]. Further, it can be used in a variety of new fields, such as renewable energy sources and urban planning [LMH10].

Urban Physics Simulation

City models are complex systems of physical objects that can be considered as an interface between building and territory, where the main physical parameters are deeply and complexly modified. Such information is needed at all scales: at lower scale by intervention (pedestrian comfort, building thermal efficiency), as boundary conditions, and at upper scale (meteorology, climate). This requirement suggests a unified simulation approach able to represent the physical behavior of the system at each scale with a level of detail required for a given accuracy of the simulation.

Current numerical computational methods, as for example the Finite Element Method, allows to model phenomena at individual scales such as a building or a block street, with different degrees of accuracy. But fully integrated multiscale approaches are still an open research subject [Bec12]. Beyond the dimensionality that overtakes computer capacities in memory and processing time, adaptive level of detail with respect to the analysis needs comes as an imperative requirement to deal with the problem.

Concerning solar energy simulation, defining the optimal LoD at the neighborhood scale is not a simple problem and most of the approaches are taken from an empirical perspective. In [RBPB12], a study of the sensitivity of the geometry used is carried out taking into account the solar flux computation, where for a neighborhood-scale model, different levels of detail elements (windows and roofs) are evaluated.

3. Preprocessing

Figure 2 gives an overview of the proposed method. Basically, the method consists of partitioning the city into a grid of cells, and for each one, we select an associated subset of the urban geometry, together with a cylindrical panoramic image of the geometry of the city after excluding the selected cell geometry. We call that cells *cores* (or *viewcells*).

To further improve the final quality of the results, for each cell we actually keep the geometry of a subset of $N \times N$ cells centered in the selected cell (N is odd in our implementation, to guarantee symmetry). See Figure 3.

Later, when evaluating the SVF for any position inside the cell, the actual geometry associated with the cell is used, together with the environment map of the rest of the city, to compute the final SVF value. Obviously, this introduces an error that is easy to control by adjusting both the size of each cell and the number N of cells taken to compute the cell's geometry. Please note that this approach assumes no overhanging geometry, but extensions to include this case are simple to imagine. Also, for regular cities, the possibility of aligning the grid with the street directions would naturally reduce any approximation error.

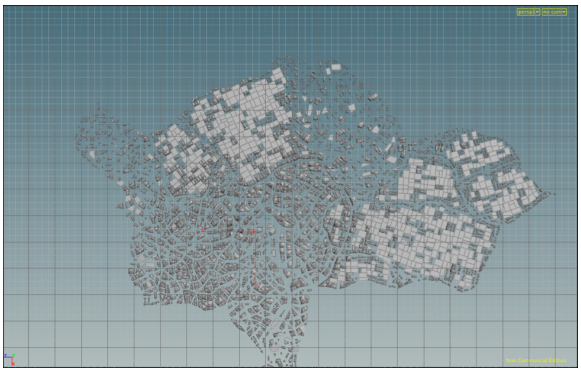

Figure 3: *To generate the cells, we embed our city in a regular grid of cells.*

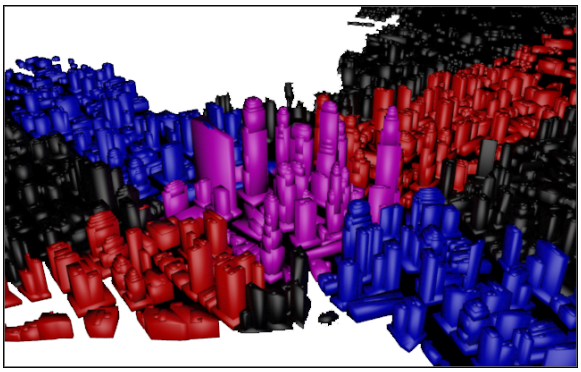

Figure 4: *Given a city and a slice width (red/blue buildings), we select a cell (purple) based on the intersection of two perpendicular lice widths.*

Given a large city, a regular grid is superimposed, and the buildings inside each cell are identified, as can be seen in Figure 4. Then, we generate two sets of buildings: those associated to and within the cell, and those of the whole city excluding the ones in the first set. See Figure 5.

The final step in our pre-computation is to build the Environment Map. In a classic implementation, the environment (i.e., the buildings in our second set) is projected onto the six faces of a cube and stored as six square textures or unfolded into six square regions of a single texture. However, as we are interested in computing the SVF, the projections on the top and bottom sides are not needed, as they would result in empty and full projections, respectively. At this point it

© The Eurographics Association 2015.

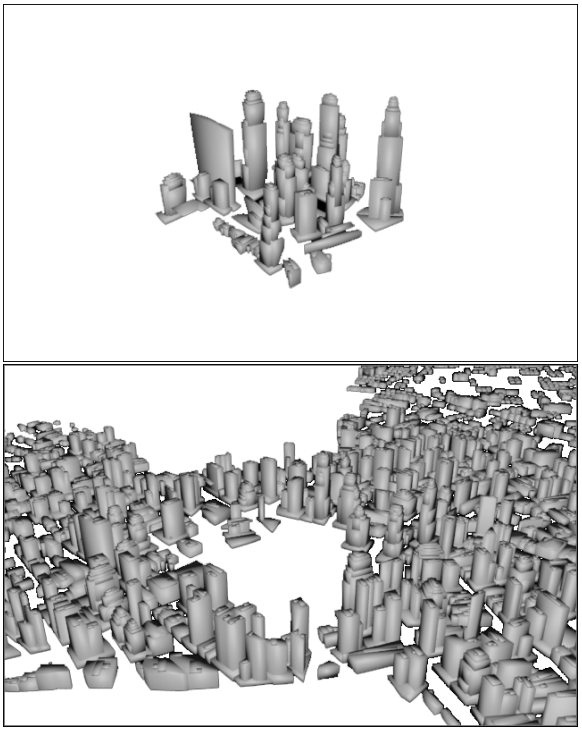

Figure 5: *Given a city, we partition the original set of buildings into two disjoint sets: Top, the ones that are associated with a cell, and Bottom, the rest of the buildings in the city.*

is important to recall that we deleted all the buildings in the proximity of the projection point, which guarantees that the top projection will always return an image of the empty sky, and thus can be neglected for computations. Also, observe that an Environment map is a projection onto a surface that is considered to be at a large distance of its center of projection, or that the cube it is projected onto is large with respect to the size of the environment. This will be an important point later on to understand the nature of our approximation and the error that can be obtained from it. See Figure 6.

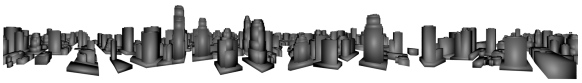

Figure 6: *An environment map computed from the non-selected buildings in a cell, as can be seen in Figure 5.*

4. SVF computations

For the correct computation of the SVF, we used the accurate rule for disk and hemisphere partition into equal-area cells presented by Beckers and Beckers [BB14]. For this, we generated a number of samples following that distribution, and, for each sample, traced a ray from the current center of projection. See Figure 7.

We can describe the process with the procedure described in Listing 1. The method takes as input the already mentioned sample distribution, the geometry for a given cell and its corresponding environment map. Then, for each sample it evaluates its intersection (from the projection point) with the city geometry and then verifies if the intersection hits a building. If not, the environment map is checked, and a possible intersection with a building (actually, its image projected onto the environment map) is tested. Again, a building is an occluder, so any hit with a building in the geometry or on the environment map should be computed as occlusor.

```
1  def computeSVF(distributionSamples,
       cellGeometry, cellEnvMap):
2    hits = 0
3    for r in distributionSamples:
4      t = r.intersects(cellGeometry)
5      if t is a building:
6        hits++
7      else
8        t = r.instersect(cellEnvMap)
9        if t is a building:
10         hits++
11   return hits/length(distributionSamples)
```

Listing 1: *Procedure to compute the SVF from a given sample distribution.*

One important observation is that we actually do not use the full sphere of directions, only the upper half which represent the samples that are meaningful for the computation of the SVF [BB14].

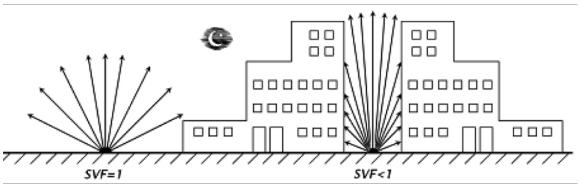

Figure 7: *The SVF is computed by tracing rays from two different centers of projection.*

5. Results and Discussion

Figure 8 shows the geometry hit from the projection point represented by a small half-sphere in the middle of the building geometry. Only a reduced set of polygons is selected for the actual computations, while the rest of the rays either hit the buildings in the environment map or contribute to the view factor.

© The Eurographics Association 2015.

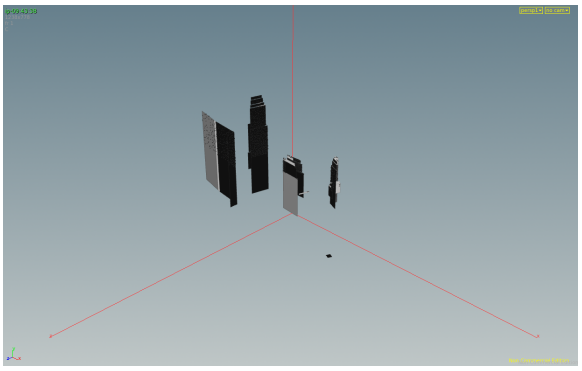

Figure 8: *The actual geometry that is intersected from a projection point located inside a dense urban area. Observe that only a fraction of the actual geometry contributes to the final result.*

We can study the time complexity of the algorithm by considering that the cost of tracing rays on a number R of objects is of the order $O(f(R))$, with f an arbitrary function depending on the acceleration data structure used. In our case, this cost is limited to a reduced number r of buildings, with $r << R$, thus resulting in a net result of $O(f(r)) + O(1)$, because evaluating the environment map requires constant computational time. Our preliminary results show reductions to $1/3$ even $1/4$ of the computational time with respect to a full ray-tracing solution for a city with 136482 polygons. However, the computational load in this case was transferred to memory consumption: The evaluation of several SVFs for a given cell requires a constant memory proportional to the number of buildings associated to the cell, which we can assume roughly proportional to its size $r = CellSize^2 * N^2$, plus the memory needed for the environment map. For a full city, we must take into account that every cell requires r buildings and an environment map, thus considerably increasing the associated memory costs. Fortunately, this is largely compensated when several SVFs are needed inside a reduced number of cells, which reduces storage costs to those of these cells.

With respect to to the accuracy of our computations. in our preliminary experiments, for $cellSize = 50m \times 50m = 250m^2$ and $N = 3$, we have observed a maximum error below 10%, and also that increasing $cellSize$ to 100×100 also reduces this error to below 5%, which confirms our previous analysis.

At this point, it is important to mention the error introduced by the use of the environment map in the approximations. As mentioned above, environment maps in general consist of an image, projected onto a given projection point, of a scene that is assumed to be very large with respect to the measuring area, in our case a cell. This corresponds to an approximation where two different rays, leaving two different positions but with the same directions, will end up hitting the same point on the environment map. This is the so called parallax error . In our case, this means that two identical samples computed from two different points will result in the same occlusion value. For projection points that coincide with the one used for the environment map construction, this error is null, and the only possible remaining error is the one of the finite-area pixels on the panoramic image, which we found to be negligible. For SVF computations inside the cell but not on the same projection point, this error will increase because of the mismatch between the projection point used and the one for the environment map. However, this error is controlled by defining the geometry associated with a given cell as the geometry of the cell itself plus its $N \times N$ neighbors. Also, this error is easy to control by the user by setting appropriate values for the cell size and for N. Actually, in the limit of the cell size going to 0, or when N is a large number, the associated error becomes 0: in the first case, the projection point will be the same as the one the environment map was taken from, and, in the second (i.e., N being very large), the geometry associated to the cell would become close to the full city itself, while the environment map would represent a smaller fraction of the outer city. In both cases, the net result is that the error becomes 0, at the expense of an increased computational cost.

One drawback of our technique is the large memory costs associated with storing, for each cell, its associated geometry and environment maps. However, according to our experience, this increased storage cost is largely outweighted by the increase in speed for the computation of a large number of Sky View Factors. If only a single (or a reduced number of) calculation is required, it is clearly better to directly compute the SVF with the full city geometry.

6. Conclusions

We have presented a method to approximate, within a user specified range, the SVF in a large complex urban landscape. This is accomplished by embedding the city into a regular grid, and approximating the areas of the city far from the SVF computation point with an environment map associated with the cell. In our approximation, the user has an effective control of the error by selecting the cell size, the number of cells surrounding the SVF projection point to be kept as geometry for accurate computations, or both. This allows the user to find the appropriate balance between speed of computation and accuracy of the results.

One of the most promising avenues for future work is finding ways to take advantage of the observation made above, that is, that only a limited number of geometric elements surrounding a projection point do have an actual influence on the computations. This would require a computation of the involved geometry, which can be a complex task for highly detailed city models. Also, an accurate error metric should

© The Eurographics Association 2015.

be made in order to automatically determine *N* and *cellSize* depending on the maximum parallax error allowable.

Acknowledgements

This work was partially funded by the project TIN2014-52211-C2-2-R from Ministerio de Economía y Competitividad, Spain.

References

[And05] ANDERS K.-H.: Level of detail generation of 3d building groups by aggregation and typification. In *In: Proceedings of the XXII International Cartographic Conference, La Coruna* (2005). 2

[BB14] BECKERS B., BECKERS P.: Sky vault partition for computing daylight availability and shortwave energy budget on an urban scale. *Lightning Research and Technology 46*, 6 (Nov. 2014), 716–728. 4

[BBBP14] BESUIEVSKY G., BARROSO S., BECKERS B., PATOW G.: A Configurable LoD for Procedural Urban Models intended for Daylight Simulation. In *Eurographics Workshop on Urban Data Modelling and Visualisation* (2014), Besuievsky G., Tourre V., (Eds.), The Eurographics Association. doi:10.2312/udmv.20141073. 2

[Bec12] BECKERS B. (Ed.): *Solar Energy at Urban Scale*. Wiley, 2012. 3

[BP13a] BESUIEVSKY G., PATOW G.: Customizable lod for procedural architecture. *Computer Graphics Forum 32*, 8 (2013). 2

[BP13b] BESUIEVSKY G., PATOW G.: The skylineengine system. In *XXIII Congreso Español De Informática Gráfica, CEIG2013* (Madrid, Spain, 2013), pp. 29–36. 2

[DSSD99] DÉCORET X., SILLION F. X., SCHAUFLER G., DORSEY J.: Multi-layered impostors for accelerated rendering. *Comput. Graph. Forum 18*, 3 (1999), 61–73. 2

[ED07] EISEMANN E., DÉCORET X.: On exact error bounds for view-dependent simplification. *Computer Graphics Forum 26*, 2 (2007), 202–213. 2

[Esr14] ESRI: Cityengine, 2014. http://www.esri.com/software/cityengine. 2

[Gre86] GREENE N.: Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl. 6*, 11 (Nov. 1986), 21–29. 2

[HPE01] HOLMER B., POSTGÅRD U., ERIKSSON M.:. *Theoretical and Applied Climatology 68*, 1-2 (2001). 3

[JW02] JESCHKE S., WIMMER M.: *An Error Metric for Layered Environment Map Impostors*. Tech. rep., Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11 / 186, A-1040 Vienna, Austria, 2002. 2

[JWS02] JESCHKE S., WIMMER M., SCHUMANN H.: Layered environment-map impostors for arbitrary scenes. In *Proceedings of Graphics Interface 2002* (May 2002), St"urzlinger W., McCool M., (Eds.), AK Peters Ltd., pp. 1–8. 2

[Kol09] KOLBE T. H.: Representing and exchanging 3d city models with citygml. In *Lecture Notes in Geoinformation and Cartography* (2009), Springer Verlag, p. 20. 2

[LMH10] LIN T.-P., MATZARAKIS A., HWANG R.-L.: Shading effect on long-term outdoor thermal comfort. *Building and Environment 45*, 1 (2010), 213 – 221. International Symposium on the Interaction between Human and Building Environment Special Issue Section. 3

[PM01] PARISH Y. I. H., MÜLLER P.: Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001), Press, pp. 301–308. 2

[RBP14] ROURE F., BESUIEVSKY G., PATOW G.: Hierarchical Radiosity for Procedural Urban Environments. In *Eurographics Workshop on Urban Data Modelling and Visualisation* (2014), Besuievsky G., Tourre V., (Eds.), The Eurographics Association. 2

[RBPB12] RODRIGUEZ D., BESUIEVSKY G., PATOW G., BECKERS B.: Procedural models to better compute solar flux at the neighbourhood scale. In *Proceedings of Flow modeling for urban development* (2012). 3

[RCT*06] RAU J.-Y., CHEN L.-C., TSAI F., HSIAO K.-H., HSU W.-C.: Lod generation for 3d polyhedral building model. In *Advances in Image and Video Technology*, Lecture Notes in Computer Science. 2006. 2

[SLS*96] SHADE J., LISCHINSKI D., SALESIN D., DEROSE T., SNYDER J. M.: Hierarchical image caching for accelerated walkthroughs of complex environments. In *SIGGRAPH* (1996), pp. 75–82. 2

[Ung04] UNGER J.: Intra-urban relationship between surface geometry and urban heat island: review and new approach. *Climate Research 27*, 3 (2004), 253–264. 3

[UPSK08] UMENHOFFER T., PATOW G., SZIRMAY-KALOS L.: Robust multiple specular reflections and refractions. In *GPU Gems 3*, Nguyen H., (Ed.). Addison-Wesley, 2008, pp. 387–407. 2

© The Eurographics Association 2015.