T. Fischer, K. De Biswas, J.J. Ham, R. Naka, W.X. Huang, *Beyond Codes and Pixels: Proceedings of the 17th International Conference on Computer-Aided Architectural Design Research in Asia,* 647–656.

REAL-TIME RENDERING OF LARGE BUILDING INFORMATION MODELS

Current state vs. state-of-the-art

MIKAEL JOHANSSON AND MATTIAS ROUPÉ
Chalmers University of Technology, Gothenburg, Sweden
jomi@chalmers.se

Abstract. With the use of Building Information Models (BIM), real-time 3D visualisations have become a natural tool in order to communicate ideas and share information between all involved parties in a project. Currently, several different BIM viewers are available for the purpose of interactive presentations and design reviews. However, as BIMs become larger and more detailed, it provides a challenge for available software solutions to manage them interactively. In this paper we present our findings from analysing three commonly used BIM viewers - Tekla BIMSight, Autodesk Navisworks and Solibri Model Viewer - in terms of real-time rendering performance. In addition we have developed a prototype BIM viewer to test modern approaches for efficient real-time rendering. Specifically, we have implemented the latest version of the Coherent Hierarchical Culling algorithm. Our results show that existing BIM viewers all share limitations in their ability to handle large and complex BIMs interactively. However, for the same test models, our prototype BIM viewer enables smooth real-time performance with no visual artefacts. The results from our tests thus shows that the technology to enable correct real-time rendering of large and complex BIMs is already accessible, but are currently not utilised by any of the tested BIM viewers.

Keywords. 3D graphics; BIM; real-time rendering.

1. Introduction

With the creation of Building Information Models (BIM), the content produced by architects and designers have evolved from traditional 2D drawings

and sketches to parametric, object-oriented 3D-models that can describe any building or facility in detail. As a natural extension, BIM further simplifies the use of real-time visualisations as a tool to communicate ideas and share information between all involved parties in a project. Currently, several different BIM viewers – both commercial and free – are available for the purpose of interactive presentations and design reviews. During these sessions it is desirable that the software can provide a smooth and interactive experience that is trustworthy. In other words, the system should be able to provide a sufficiently high frame rate without sacrificing visual correctness. However, based on our own practical experience with several of the available BIM viewers we have observed that this is not the case when large and detailed BIMs are used. Although some of these viewers have functionality to maintain interactive navigation by guaranteeing a certain frame rate, this is realised by presenting a visualisation that is incorrect. As BIMs tends to become larger and more detailed as the technology matures, we initiated a more comprehensive study within this area.

In this paper we present our findings from analysing three commonly used BIM viewers – Tekla BIMSight, Autodesk Navisworks and Solibri Model Viewer – in terms of real-time rendering performance. The focus is on large and complex BIMs and we have used several different test models to stress-test the available software solutions. In addition, we have developed a prototype BIM viewer to use as a reference and to test modern approaches for real-time rendering of large and complex 3D-models. Specifically, we have aimed for a solution that can provide both interactivity and correctness, even for large and detailed BIMs.

The rest of this paper is organised as follows. In the next section we briefly discuss a number of acceleration techniques for efficient real-time rendering. Section 3 presents our findings from analysing the existing BIM viewers in terms of real-time rendering performance. In Section 4 we motivate the choice of acceleration technique for our prototype BIM viewer and in Section 5 we present a performance analysis of it. Finally, Section 6 concludes the paper.

2. Acceleration techniques for real-time rendering

An important property for any type of real-time rendering system is its ability to maintain a sufficiently high frame rate. Although this number is highly dependent on the type of application, it is generally considered that it should at least be 20 Hz (Bittner 2002). Regardless, a too low or fluctuating frame rate will make navigation and other interaction tasks more challenging and may also cause participants to lose orientation or even feel sick (Yuan et al. 1997, Mortensen et al. 2008). However, depending on the amount of data to

be visualised this is not always an easy task to achieve. Even if the performance of computers and graphic processing units (GPUs) has increased tremendously during the last years there is always an upper limit in the amount of 3D-data that the technology supports out-of-the-box. Fortunately, a number of acceleration techniques exist that allows us to go beyond this limitation (Akenine-Möller et al. 2008). These techniques can basically be assigned into one of the following three categories: *GPU-centric optimisations*, *Level-of-detail* or *visibility culling*.

GPU-centric optimisation refers to several different strategies that are used in order to maximise the raw processing power of modern GPUs. Often, the impact on performance can be huge if the 3D-data is arranged and sorted in an optimal way before sent to the GPU. A typical example is to arrange the draw-order of objects by material properties. This will minimise costly state changes, such as switching shaders or textures, and let the GPU work with as few interrupts as possible. However, although efficient, the common characteristic of GPU-centric optimisations is that they do not reduce the amount of data that has to be processed by the GPU. As such, the approach is not indefinitely scalable.

Compared to GPU-centric optimisations, a more scalable approach is to consider Level-of-detail (LOD). With LOD, the main idea is to reduce the complexity of a 3D object representation when the object is far away from the current viewpoint. In such a situation the object becomes small on screen and a less detailed representation can be sufficient in order to give the same visual impression. The simplified version of the object is often created by reducing the number of triangles, replacing geometric features with textures or a combination of both. Regardless of simplification strategy the end result is an object that is less stressful for the GPU to process and therefore the real-time performance of the visualisation is increased.

Finally, visibility culling refers to a category of techniques where the idea is to improve performance by only sending objects that are potentially visible to the GPU. Today, most real-time rendering systems implement at least view frustum culling, where objects are discarded if they are found to be outside the visible region of space, as defined by the virtual camera. Although this approach can give a huge performance increase, it does not take into account a situation when objects in a 3D environment are entirely behind other opaque objects. In order to reduce workload in such a situation some form of occlusion culling has to be utilised. With occlusion culling the idea is to discard objects that are guaranteed to be hidden in the final image. The technique is considerably harder to implement compared to view frustum culling, but has the potential to increase performance substantially for 3D environments with

a lot of occlusion. As such, it is a viable acceleration technique for real-time rendering of building models and will be further discussed in Section 4.

3. Performance analysis and comparison of existing BIM viewers

In order to analyse and compare the existing BIM viewers in terms of real-time rendering performance, four different BIMs were used (Figure 1). Except for the A model, they all represent planned or existing buildings. The A model is an initial test model that, although not actually planned or build, may be considered representative for a student housing building. All four models were created in Autodesk Revit Architecture 2012 and exported to the IFC file format (BuildingSMART 2011). In Table 1, related statistics for all models are presented.

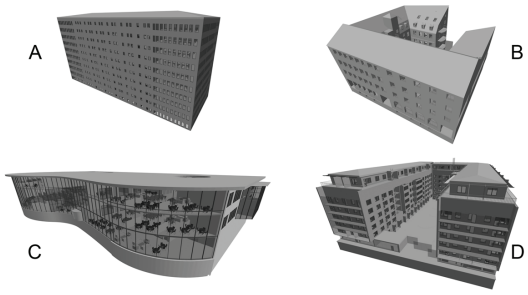

Figure 1. The four different test models.

TABLE 1. Statistics for the different test models.

Model	Type of building	Number of objects	Number of triangles
A	Student housing	5,683	9,559,028
B	Apartment building	2,151	1,474,586
C	Library / Culture building	6,224	4,102,959
D	Student housing	15,861	9,958,143

To measure the actual performance, FRAPS was used (FRAPS 2011). FRAPS is a software that "hooks into" the graphics driver in order to measure frame rate. The frame rate (in frames per second) is then displayed as a numerical value in the viewport of the currently active viewer. Unfortunately, FRAPS was unable to work together with the Tekla BIMSight viewer. For the Tekla BIMSight viewer we instead use a modified version of the GLIntercept software (GLIntercept 2011) that allows us to measure frame rates similar to

FRAPS. All the performance tests were performed on a computer equipped with an Intel i7 2.93 GHz CPU, 4 GB of RAM and an Nvidia GeForce 460 GTX GPU. The operating system was Windows 7 x64 and the viewport size was set to 1200 × 800 pixels.

In order to make a fair comparison of the different viewers in terms of performance, any acceleration technique that results in an incorrect visualisation is turned off. Specifically, both Navisworks and Solibri implements functionality in order to guarantee interactive frame rates during navigation. This is achieved by simply stop drawing objects once a certain rendering time has been reached. Although the rejection of objects appears to be based on its relative "importance" for the current view (size, type and distance to viewpoint) the approach does not guarantee a correct visualisation and often results in severe visual "popping" during navigation. Figure 2 shows an example of this in Navisworks for one of the tested building models when the desired frame rate has been set to 20 Hz.

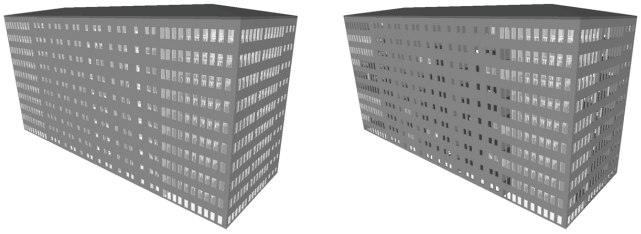

Figure 2. Visual artefacts (right) when forcing 20 Hz in Navisworks.

For all of the tests we have focused on the worst case scenario in terms of rendering performance. For all viewers this means a viewpoint where the complete building model is visible. In addition we have selected two interior viewpoints in all of our test models. These viewpoints represent neither the worst nor the best case in terms of performance, and should just be seen as typical locations inside each building. However, they do allow for a fair comparison between the tested viewers. In Figure 3, one of the selected view points (VP1) is shown for each of the models.

The results from our performance tests are presented in Table 2. Here, WC represents worst case, VP1 viewpoint one and VP2 viewpoint two for all of the tested models and viewers. The frame rates presented are consistent for the different viewpoints and any single spike or drop in frame rate has been omitted. Among the tested viewers no one except Navisworks appears to utilise any occlusion culling. However, during our tests it became clear that the occlusion culling in Navisworks is far from optimal and actually lowers

the frame rate in some of the tested models. We therefore present the performance results from Navisworks in two versions; occlusion culling activated (OCA), and occlusion culling deactivated (OCD).

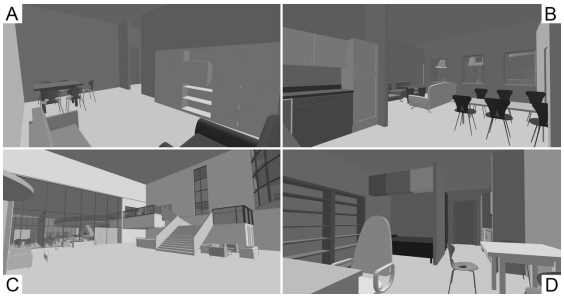

Figure 3. Selected viewpoint (VP1) for the different test models.

TABLE 2. Frame rates (in frames per second) for the four tested BIM viewers for models A to D: worst case (WC), viewpoint 1 (VP1) and viewpoint 2 (VP2).

BIM viewer	A (WC)	A (VP1)	A (VP2)	B (WC)	B (VP1)	B (VP2)	C (WC)	C (VP1)	C (VP2)	D (WC)	D (VP1)	D (VP2)
Tekla BIMSight	<1	5	3	5	12	7	3	7	10	<1	2	2
Autodesk Navisworks (OCA)	4	32	23	12	29	30	5	16	23	4	22	16
Autodesk Navisworks (OCD)	1	5	4	17	27	30	8	17	23	2	4	4
Solibri Model Viewer	26	59	52	62	80	129	23	43	42	8	11	16

When comparing the results it becomes clear that the Solibri Model Viewer offers the best performance among the tested BIM viewers. The tests further reveals a far from optimal implementation of occlusion culling in the Navisworks viewer, as it actually lowers the frame rate for many of the models. As far as we can tell, the higher performance of the Solibri Model Viewer probably comes from GPU-centric optimisations. As none of the tested viewers (except Navisworks) supports occlusion culling or LOD, the difference between them can only depend on their ability to utilise the GPU efficiently.

However, even if the Solibri Model Viewer turned out to offer the best performance, it was not able to provide sufficiently high frame rates for all of our test models. Although these models can be considered large and detailed, they are far from being a worst case scenario, even today. Taking future directions for the use of BIMs into account it should be safe to assume even larger models tomorrow. As such, a more performance efficient solution for real-time rendering of BIMs is required. The next section addresses the situation further by analysing potential acceleration techniques for our prototype BIM viewer.

4. A suitable acceleration technique for BIMs

As discussed previously, there are a number of techniques that can be utilised in order to accelerate real-time rendering. These have all strengths and weaknesses and a suitable choice is highly dependent on the type of 3D environment that it should be applied to. For instance, a vast, open landscape seen in a flight simulator is very different from a detailed city environment seen from the ground level when it comes to performance optimisation. For any type of real-time rendering system it is important to consider GPU-centric optimisations, such as efficient arrangement of draw order. Still, these techniques are only helpful up to a certain point. Once the 3D environment reaches a certain size in terms of amount of data we need an additional, more scalable approach. When considering BIMs, it typically contains a lot of rather small objects, such as furniture, lighting fixtures and sanitary equipment. When viewed from a distance these objects becomes very good candidates for LOD. However, even if this is a viable option in terms of accelerated rendering, it introduces problems in practice as it requires the existence of simplified 3D-models of all these objects. Although approaches exist to automate this (Garland and Heckbert 1997, Wang et al. 2009), it usually requires at least some user interaction and supervision in order to guarantee pleasant results for general 3D-models.

Another characteristic of BIMs is that they typically exhibit a lot of occlusion. For reasons that are obvious, a building is naturally divided into different zones, rooms and floors. Although some buildings, such as a concert hall or a library can be very open in their layout, there always exist areas that are separated, in terms of visibility, from the rest of the building. Given this, a suitable acceleration technique would therefore be to use any type of occlusion culling. Within this category several different algorithms exists that are mainly differentiated by whether they require time-consuming offline computations or not (Bittner and Wonka 2003, Cohen-Or et al. 2003). For the purpose of review sessions that require the visualisation to be initiated on-demand, an offline solution is clearly not the most suitable approach. For our prototype BIM viewer

we have therefore chosen to implement what is generally considered current state-of-the-art in terms of online visibility detection - the latest version of the Coherent Hierarchical Culling algorithm (Mattausch et al. 2008).

4.1. COHERENT HIERARCHICAL CULLING

Occlusion queries is a feature of modern GPUs that lets any application query the number of pixels that will end up on screen when rendering a specific set of geometries. This way, proxy-geometries can be used to test if any occlusion is present before the actual object is rendered. However, the use of occlusion queries introduces latency in the system which may lead to a decrease in performance if used naively. The initial Coherent Hierarchical Culling algorithm (CHC) (Bittner et al. 2004) makes us of temporal and spatial coherence in order to reduce this latency. In essence, this is realised by interleaving the rendering of objects with the issuing of queries while traversing the 3D scene (organised in a bounding volume hierarchy) in a front-to-back order. During traversal, queries are only issued for previously invisible interior nodes and for previously visible leaf nodes of the hierarchy. Still, for scenes with a low level of occlusion, the initial version of the algorithm can actually decrease performance (compared to only using frustum culling). To address this, CHC++, a revised version of the algorithm was developed (Mattausch et al. 2008). Although the core ideas remain the same, CHC++ introduces several optimisations which make it perform very well even in situations with low occlusion. For our prototype BIM viewer the revised version of the algorithm has been implemented, and in the following section we present a performance analysis of it when applied to the different test models.

5. Performance analysis of our prototype BIM viewer

For the performance test of our prototype BIM viewer we have used the same test models and interior view points as described in Section 3. As we utilise an efficient occlusion culling algorithm the performance is highly dependent on the view point we chose. For all of the models we therefore present the lowest frame rates that we have encountered during several interactive navigation sequences (Table 3). For the most complex model (D), we provide additional information about the rendering performance in Figure 4. Here, the frame rates are presented for a pre-defined navigation sequence when we are following a path around the building while the view direction is oriented towards the building. As the complete building is visible throughout the whole navigation sequence, albeit from different direction, it should be representative as a worst case scenario.

TABLE 3. Frame rates (in frames per second) for our prototype BIM viewer for models A to D: worst case (WC), viewpoint 1 (VP1) and viewpoint 2 (VP2).

BIM viewer	A (WC)	A (VP1)	A (VP2)	B (WC)	B (VP1)	B (VP2)	C (WC)	C (VP1)	C (VP2)	D (WC)	D (VP1)	D (VP2)
Prototype BIM viewer	100	800	700	260	500	450	120	285	400	94	550	300

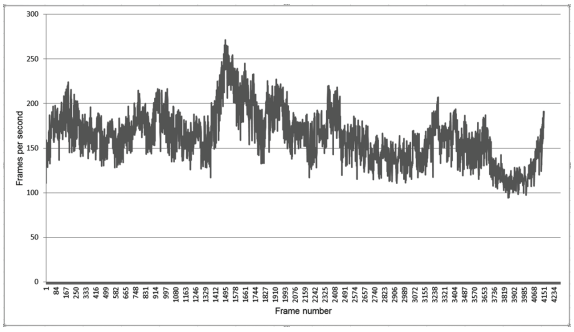

Figure 4. Frame rates (in frames per second) for our prototype BIM viewer for the D model during the pre-defined navigation sequence.

As seen by the result presented in Table 3 there is a huge difference when compared to that of the existing BIM viewers. This is especially true for the interior view points, but also for the worst case scenarios the performance is impressive. Even for the C model, which doesn't exhibit as much occlusion, the speed-up is almost 6 times compared to the Solibri Model Viewer. Furthermore, the data presented in Figure 4 reveals that, except for a small drop around frame 3900, the frame rate is above 100 frames per second throughout the complete navigation sequence. The tests thus confirm our assumption that an efficient occlusion culling algorithm is a suitable choice for real-time rendering of BIMs.

6. Conclusions

Regarding existing BIM viewers, our results show that they all share limitations in their ability to handle large and complex BIMs interactively. Although both Navisworks and Solibri Model Viewer implements functionality to maintain real-time performance during navigation, it is realised through rejection of objects that should, in fact, be visible. As a result, the presented visualisa-

tion is incorrect and exhibit visual popping. Without this acceleration technique enabled, Solibri turned out to offer the best performance. Still, it was not able to provide sufficiently high frame rates in all of our tests. However, for the same test models, our prototype BIM viewer enables smooth real-time performance with no visual artefacts. The results from our tests thus highlight the efficiency of the CHC++ algorithm and show that the technology to enable correct real-time rendering of large and complex BIMs is already accessible, but are currently not utilised by any of the tested BIM viewers.

References

Akenine-Möller, T., Haines, E. and Hoffmann, N.: 2008, *Real-Time Rendering*, 3rd ed., AK Peters, Wellesley.

Bittner, J.: 2002, *Hierarchical Techniques for Visibility Computations*, PhD thesis, Czech Technical University, Prague.

Bittner, J. and Wonka, P.: 2003, Visibility in computer graphics, *Environment and Planning B*, **30**(5), 729–756.

Bittner, J., Wimmer, M., Piringer, H. and Purgathofer, W.: 2004, Coherent hierarchical culling: hardware occlusion queries made useful, *Computer Graphics Forum*, **23**(3), 615–624.

BuildingSMART: 2011, "Model - Industry Foundation Classes (IFC) — buildingSMART". Available from: <http://buildingsmart.com/standards/buildingsmartstandards/ifc>.

Cohen-Or, D., Chrysanthou, Y. L., Silva, C. T. and Durand, F.: 2003, A survey of visibility for walkthrough applications, *IEEE Transactions on Visualization and Computer Graphics*, **9**(3), 412–431.

FRAPS: 2011, "Fraps: Real-time video capture and benchmarking". Available from: <http://www.fraps.com/>.

Garland, M. and Heckbert, P.: 1997, Surface simplification using quadric error metrics, *Proc. SIGGRAPH'97*, 209–216.

GLIntercept: 2011, "GLIntercept". Available from: <http://code.google.com/p/glintercept/>.

Mattausch, O., Bittner, J. and Wimmer, M.: 2008, CHC++: Coherent hierarchical culling revisited, *Computer Graphics Forum*, **27**(2), 221–230.

Mortensen, J., Yu, I., Khanna, P., Tecchia, F., Spanlang, B. and Marino, G.: 2008, Real-time global illumination for VR applications, *IEEE Computer Graphics and Applications*, **28**(6), 56–64.

Wang, H., Yin, G., Zhang, J., Wang, D. and Wang, J.: 2009, An efficient mesh simplification algorithm, *in* Ni, J. et al. (ed.), *Proc. 4th ICICSE Conference*, IEEE Computer Society Press, Washington DC, 60–63.

Yuan, P., Green, M. and Lau, R.: 1997, A framework of performance evaluation of real-time rendering algorithms in virtual reality, *in* Thalmann, D. et al. (eds.), *ACM Symposium on Virtual Reality Software and Technology 1997*, Lausanne, 51–58.