GAME OF RENDERS

The Use of Game Engines for Architectural Visualization

ANTÓNIO LEITÃO¹, RENATA CASTELO-BRANCO² and
GUILHERME SANTOS³

¹,²,³INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
¹,²,³{antonio.menezes.leitao|renata.castelo.branco|guilherme.j.santos}@tecnico.ulisboa.pt

Abstract. Good visualization mechanisms offer architects, and their clients, a better grasp of how their designs are going to turn out when built, and the experience one might have inside the constructions. This also helps the architect orient the design in a more informed manner. However, typically used modeling tools do not offer satisfactory visualization solutions. The operations available to view and navigate through the 3D space are flawed in terms of speed, interactivity, and real-time rendering quality. To solve this issue, we propose the coupling of a portable algorithmic design framework with a Game Engine (GE) to support interactive visualization of architectural models and increase the rendering performance of the framework. We explain in detail this integration, and we evaluate this workflow by implementing a case study and comparing the performance of the GE to architectural modeling tools.

Keywords. Algorithmic Design; Game Engine; Interactive Visualization.

1. Introduction

Rendered images and films have become a requisite in any architectural project (Ng et al. 2006). Not only are these the preferred means of visualization for clients, but architects are also beginning to rely on these more realistic visualization mechanisms for their design process (Shiratuddin and Thabet 2002).

However, architectural modeling tools do not offer satisfactory visualization solutions (Moloney and Harvey 2004, Pelosi 2010) and the operations available to view and navigate through the 3D space have problems regarding speed, interactivity, and real-time rendering. On the other hand, all of these are basic traits of Game Engines (GE), which have been developing solutions for these problems for quite some time (Indraprastha and Shinozaki 2009).

Given the advantages of GEs, ways of importing architectural models have already been developed. Nevertheless, the majority of these processes is still tiresome and error-prone. We propose to solve this problem by combining a GE with a portable Algorithmic Design (AD) approach.
2. Visualization in Architecture

The rapid development of digital technologies, allied to an ever-growing eagerness to innovate, has been pushing architectural projects towards more complex design solutions (Hensel and Nilsson 2016), which can hardly be represented by two-dimensional drawings only. As a result, the hegemony of 3D models has long been established.

Regarding the software used to produce 3D models, the industry is divided into two paradigms: Computer-Aided Design (CAD) and Building Information Modeling (BIM) (Kolarevic 2004). While the former is best suited for free-form modeling (Zboinska 2015), the latter was conceived as a collaborative platform for the various specialties involved in a project (Kensek and Noble 2014). CAD geometry is free of BIM constraints and, thus, faster to generate and manipulate than BIM geometry. BIM objects, on the other hand, have associated semantics, which facilitates and automates many of the tasks that have to be performed. However, this does not come without a loss of modeling freedom and performance.

In regard to the visualization mechanisms available, neither paradigm offers a satisfactory solution (Moloney and Harvey 2004), particularly regarding navigation through the 3D space in real-time (Indraprastha and Shinozaki 2009, Pelosi 2010).

2.1. GAME ENGINES

Good visualization mechanisms offer architects, and their clients, a better grasp of how their designs are going to turn out when built and the experience one might have inside the constructions. This also helps the architect orient the design in a more informed manner (Shiratuddin and Thabet 2002).

In order to have realistic representations, architects typically couple modeling tools with render engines, presented as independent tools or plug-ins for CAD/BIM tools (Castelo Branco and Leitão 2018). However, these rendering engines require extensive work to guarantee good rendering results and they take considerable time to produce just one single rendered image. This last problem also means that they cannot offer a satisfactory answer regarding interactivity.

GEs, on the other hand, are optimized to provide, precisely, realism and interactiveness in large indoor or outdoor environments. In GEs, model visualizations benefit from dynamic lighting and shadowing, which can also be improved with programmable characteristics, such as specularity, reflection, refraction, or polished surfaces (Andreoli et al. 2005). The fact that they are physics-based engines (Fritsch and Kada 2004, Juang et al. 2011) facilitates the manipulation of objects in the scene to create ambiances. For instance, for the production of films, it is particularly important for people, vehicles, and vegetation to have realistic behavior and interactions (Andreoli et al. 2005). In this sense, artificial intelligence, originally meant to improve the agent’s and other non-player characters’ behavior in scene, can highly enhance the perceived realism of walkthrough movies/animations of the model.
2.2. OBSTACLES

There is, nonetheless, an issue with this solution: most GEs were not originally conceived to support architectural projects. Naturally, given the advantages GEs bring to architectural visualization, ways of importing models from CAD and BIM tools have been developed. Even so, the majority of these processes is still tiresome and error-prone (Boeykens 2011). Furthermore, since the geometry produced in these tools was not thought for real-time visualization, as are the objects used in GEs, the result may be an almost complete loss of performance from the GE, which partially defeats the point.

Our solution intends to solve this issue by integrating a GE in an AD tool - one where the architect converts his design intent into a computer program, which is then responsible for generating the 3D model, instead of directly manipulating geometry in 3D modeling tools.

3. Algorithmic Design

AD defines the creation of architectural designs through algorithmic descriptions (Gerber and Ibañez 2014), i.e., computer programs with sets of rigorous modeling instructions for the machine to perform (Burry 2011). This means that, differently from traditional design approaches, with AD the architect does not build the model directly, but instead, builds the program that builds the digital model.

This allows him to (1) model more complex geometries that would take a considerable amount of time to produce otherwise; (2) automate repetitive and time-consuming tasks, relieving him from tedious and error-prone work (Burry 2011); and (3) effortlessly generate diverse design solutions, since the resulting algorithmic descriptions are parametric (Woodbury 2010).

3.1. PORTABILITY

AD presents yet another important advantage: algorithmic-based descriptions, if properly implemented, stand on their own, independently of the tools that may be used to generate the resulting model. Provided we have good translation mechanisms, capable of converting the programming operations used by the architect to the modeling operations recognized by each tool, we can devise programs that benefit from different modeling paradigms.

Design approaches have been developed with this premise in mind, integrating different paradigms and workflows in the same design process, and allowing the architect to program his intent in one single programming environment (Castelo-Branco and Leitão 2017). Using such an approach, architects can combine assets from different tools, namely CAD tools for design concept and geometry experimentation, BIM tools for the generation of more detailed models and project documentation, and they can also incorporate analysis of the design in any stage of the process, possibly integrating optimization algorithms in order to find the best performing design (Aguiar et al. 2017, Belém and Leitão 2018).
3.2. EXTENSION TO VISUALIZATION

Some experiments have also been made regarding the production of specific visualization models and the effortless generation of adaptable visualizations (Castelo-Branco and Leitão 2018). This work included, alongside the model’s description, the parametric descriptions of the rendering tasks using cinematographic techniques. To this end, parametric pre-modeled objects from BIM libraries were used directly in the program, saving the architect the time he would take to model them himself.

However, several problems arose from this approach. First and most imposing, BIM models are heavily filled with semantic information, which makes them very appealing when dealing with constructive detail and producing documentation, but makes them quite hard to manage and nearly impossible to interact with in complex projects. This issue is aggravated in proportion to the amount of objects we place in the model, and it escalates when producing visualizations, as the render engines tend to take inordinate amounts of time to process all this information, often resulting in long and torturous rendering processes.

4. Real Time Visualization

As a solution to the presented problem, we claim that AD tools should be capable of generating, from the same program, not only equivalent CAD, BIM, and analytical models, but also a specialized model suitable for a GE. This allows the architect to benefit from all the visualization advantages GEs have to offer, using a workflow that frees him from import/export problems. Figure 1 presents a scheme of the proposed workflow.

![Diagram](image)

Figure 1: AD workflow, with geometry generated in CAD and BIM modeling tools, analysis tools, and in a game engine (detailed version for rendering, and simplified version for fast interaction).

We implemented this workflow on top of a new AD tool - Khepri. Khepri is an updated version of Rosetta, an AD tool that already integrated several CAD, BIM, and analysis backends (Leitão and Lopes 2011, Feist et al. 2016, Aguiar et al. 2017). Khepri allows architects to model their designs using Julia, a programming
language with both a smooth learning curve and the capacity for large-scale development, while guaranteeing fast execution of the computationally-intensive algorithms required in computational geometry.

4.1. IMPLEMENTATION

In the current implementation we decided to explore Unity, a popular GE that ensures: (1) high visual quality; (2) high performance; (3) platform independency; (4) assets availability; (5) up-to-date documentation; (6) good usability (Hocking 2015); and (7) low cost. Furthermore, Unity allows us to integrate current state-of-the-art methods of visualization such as Virtual Reality (VR), thus, bringing a new level of interactiveness and immersive experiences to architectural projects (Nandavar 2018).

Our implementation entails both the use of Unity’s primitive Game Objects to construct entities that can be algorithmically transformed and placed in the scene, and the construction of complex geometries as custom elements built in Khepri. We also use furniture assets, provided by the Unity’s Asset Store and other external sources, as well as a collection of physically-based rendered materials to ensure high visual quality. This kind of materials has impact on performance, specially since most architectural designs are composed by a great number of them.

In order to optimize Unity’s visualization performance, all decorative objects that share the same mesh and materials, as is the case of most furniture, were converted to Prefabs, which are essentially pre-built Game Objects that can be copied and added efficiently to the scene. This efficiency allows for both faster generation of models and low latency in live model manipulation and navigation.

Our solution supports various types of interactiveness through custom scripts: (1) a static overview with multiple viewpoints of the scene; (2) a free-fly mode where the user can fly through the scene and pass-through any objects; (3) and a walk mode where the user can explore the scene at walking speed and collide with the various elements. Unity also allows us to straightforwardly integrate VR technologies for an immersive experience. SteamVR provides a user-friendly interface to use any VR hardware, covering six degrees of freedom of head and hands tracking, and various types of locomotion for navigation (Murray 2017).

Finally, physics simulation can also be integrated into our projects, allowing us to add different kinds of scene navigation, such as wheel-chair navigation for accessibility simulation (Boeykens 2011) and other interesting experiences with object interaction. Some other aspects may also be worth exploring for simulation purposes, namely the sound and weather features available in modern GEs.

4.2. TRADEOFF ANALYSIS

The interactiveness offered and the improved generation and rendering speed are due to various underlying real-time optimizations in the rendering pipeline. These techniques include (1) occlusion culling, which inhibits the rendering of any object obscured by other elements; (2) frustum culling, which disables the rendering of any objects outside the camera’s view frustum; (3) Level of Detail (LOD), the rendering of objects in-sight but far away from the camera with reduced detail;
These optimization techniques can, nevertheless, result in poorer rendering results, when compared to dedicated architectural render engines. The visual outcome presents some over-simplifications in material processing, reflections, light and shadow casting at long distance, among others. These are necessary tradeoffs in order to guarantee all the aforementioned benefits, but GEs also give us the flexibility to fine-tune them to our needs. This means we can choose to hinder some of these performance optimization techniques for better rendering results, keeping in mind that this will decrease generation and navigation speed, as well as the real-time render capacity. Material specularity, for instance, was one of the bargains with greatest impact on performance.

The tradeoffs demanded by the GE workflow also include considerable modeling limitations, particularly, the lack of Boolean operations, which can, nevertheless, be implemented with the help of external libraries.

Finally, GEs do not provide the smart geometry treatment solutions BIM tools do, which correct and perfect modeling bugs introduced by their users. Examples include wall intersections and corners, for which BIM tools calculate smooth finishes, even if the geometry is not perfectly aligned; and Z-fighting issues, which BIM tools solve by removing superimposed geometry.

5. Evaluation

In order to evaluate the validity of our proposed approach we chose to redo the model and render sequences developed in previous work, only using the GE backend to do so, and compare the results in terms of navigability, time per render frame, and quality of generated image.

The building used as case study is the Astana National Library (ANL) project, designed by BIG architects in 2008, for Kazakhstan. ANL is characterized by complex and somewhat repetitive geometry, which makes it a great candidate for the use of algorithmic approaches. Furthermore, the building’s shape replicates a Moebius strip, with the outer skin wrapping over itself, and the inner space organized in a double loop. Navigating any model of this project in typical CAD and BIM tools is a rather hard and disorienting task, which can be greatly improved in a dedicated visualization backend, such as the GE.

5.1. ASSETS

For decorative purposes, several pre-modeled components were added to the algorithmic description of ANL. The previous visualization work rested on the use of BIM library objects, particularly in ArchiCAD’s library. For this evaluation, we mimicked the process by augmenting our algorithmic tool’s library with similar assets. We used several assets from the Web, plus a few parametric assets developed in Khepri. This last approach presents an advantage compared to the BIM one, since the objects modeled using Khepri’s operations are portable, unlike specific BIM objects, which can only be found and used within specific BIM tools.

The objects used to populate the library include doors, staircases, elevators, bookshelves, tables, and chairs. Figure 2 presents some of the BIM objects used
on previous work and some of the assets used in the current evaluation.

![Figure 2. A - Some of ArchiCAD’s pre-modeled objects; B - Assets used in Unity.](image)

### 5.2. BENCHMARKING RENDERING TIME

After filling the library’s core with furniture, we proceeded to the remaking of the algorithmically defined film sequences inside the library’s space, originally described in Castelo-Branco and Leitão (2018). The previous films, available at web.ist.utl.pt/antonio.menezes.leitao/AAV, were produced using ArchiCAD’s render engine: CineRender. The sequences produced with the new visualization backend, Unity, are available at web.ist.utl.pt/antonio.menezes.leitao/RTV. The timing measurements of the previously generated sequences and the ones reproduced in Unity are shown in Table 1. The time per frame columns in each render set show the average time to render each frame for both backends, BIM and GE. Both sets were rendered in a workstation dual Xeon E5-2670, with 64 GB RAM, a 512 GB SSD, and a NVIDIA Quadro 6000 GPU.

Table 1. Comparison of render production times in ArchiCAD and Unity. Time presented in: HH - Hours; MM - Minutes; SS - Seconds; 00 - Milliseconds.

<table>
<thead>
<tr>
<th>RENDER SEQUENCE</th>
<th>ARCHICAD RENDER</th>
<th>UNITY RENDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Dolly-zoom forh</td>
<td>1620 x 880</td>
<td>00:56:57:00</td>
</tr>
<tr>
<td>2 Tracking aerial</td>
<td>1620 x 880</td>
<td>01:16:03:07</td>
</tr>
<tr>
<td>3 Tracking parallel</td>
<td>1620 x 880</td>
<td>00:40:37:20</td>
</tr>
<tr>
<td>4 Walkthrough outer ring</td>
<td>1680 x 1050</td>
<td>01:48:22:70</td>
</tr>
<tr>
<td>Walkthrough inner ring</td>
<td>1820 x 980</td>
<td>00:34:44:15</td>
</tr>
<tr>
<td>5 Tracking line</td>
<td>1820 x 880</td>
<td>00:13:56:40</td>
</tr>
<tr>
<td>6 Tilting</td>
<td>1820 x 980</td>
<td>00:59:11:14</td>
</tr>
<tr>
<td>7 Tracking fly-over</td>
<td>1820 x 980</td>
<td>00:04:48:97</td>
</tr>
<tr>
<td>8 Panning sideways</td>
<td>800 x 600</td>
<td>00:46:36:14</td>
</tr>
<tr>
<td>9 Dolly-zoom back</td>
<td>1620 x 880</td>
<td>01:21:34:74</td>
</tr>
</tbody>
</table>

There is a great difference in the average time each backend took to render one single frame. With Unity we managed to produce, in a matter of minutes, film sequences that took several days in ArchiCAD. Scene 2, for instance, was originally planned to have 800 frames. In ArchiCAD we were forced to stop the
process at 371 frames, since it was already over 470 hours (approximately 19 days). The same sequence was reproduced in Unity, with higher resolution, in under 9 minutes and the full sequence of 800 frames in under 22 minutes. This represents a considerable gain, not only in time, but also in electrical cost savings.

It is also important to note that, in an attempt to accelerate the rendering process, for some of the sequences above, the model was only partially generated in ArchiCAD. This is the case of sequence 7 where, since no furniture would appear in the tracking, we did not generate any of the aforementioned decorative objects. Moreover, a culling mechanism devised specifically for this project had also been implemented in order to accelerate the walkthrough sequences, which required the regeneration of the model every few frames.

In Unity’s case, none of these concerns were necessary on account of the GE’s native ability to deal with these issues. Hence, for any of these sequences, we generated the complete model, and the times per frame still proved to greatly beat those of the BIM tool. If we desired even faster results, we could have also chosen to inhibit some of the geometry in specific scenes, such as lights and furniture on outdoor scenes or outdoor detail in indoor scenes.

The generation time of the model is an important benchmark not represented in this table. The Unity backend can generate the entire ANL model within seconds, a record that not even the CAD backends could beat. Naturally, GE’s cannot compare to CAD tools at modeling free-form geometry. Nevertheless, these numbers suggest this could also be a very interesting tool to explore in early stages on the design process, if architects wish to see, in almost real time, the effect their changes to the program are having in the generated model.

5.3. BENCHMARKING RENDERING QUALITY

The image quality comparison (Figure 3) revealed slightly better light, shading, and reflection results on Cinerender’s part. However, the detail found in ArchiCAD’s renders can only be seen when the visualization is produced. Normal interaction with the model within the tool’s working environment does not present this quality. On the other hand, Unity is computing all this detail in real time, which means users can navigate the model with real-time rendered textures, lights, and shadow detail. As a result, the concept of producing a visualization in Unity is reduced to a mere screenshot, made in a fraction of a second.

Figure 3. ANL renders from ArchiCAD (on the left) and from Unity (on the right).
5.4. VIRTUAL REALITY

As mentioned in the Implementation section, we could easily couple our workflow with VR technologies, allowing us to wander through the model in a more immersive environment. Figure 4 illustrates this integration, which constitutes an advantage towards clients’ satisfaction.

6. Conclusion

We proposed coupling an AD framework with a GE to support interactive visualization of architectural models, increasing the visualization and rendering performance. With this framework, architects can generate their designs in different tools according to the projects’ needs: CAD, BIM, analysis, and GE tools. The fact that the process is algorithmic also brings forth advantages regarding parametric flexibility and the ability to change the model with little effort.

Regarding the integration of the GE in the AD framework we highlight several advantages: (1) increased generation speed; (2) real-time rendering of textures, lighting, and shadowing; (3) realistic navigation and interaction with objects in the scene; and (4) extension to VR. Furthermore, the presented workflow serves different design phases or purposes, namely controlling LOD for either fast generation and navigation or better rendering results.

It is relevant to mention that, despite the performance improvements obtained, very little optimization was done. We are currently working on profiling the GE backend to identify and remove the most relevant bottlenecks, improving not only the rendering times but also the navigation fluidness. We also plan on conducting user studies to infer the perceived quality of the rendered images; as well as exploring the potential of the VR extension. Additional user studies will be performed to analyze the impact of the approach in conceptual modeling phases and, in later stages, for qualitative analysis of the designed space.

Acknowledgments

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with references UID/CEC/50021/2019 and PTDC/ART-DAQ/31061/2017.
References


