Web 3D Indoor Authoring and VR Exploration via Texture Baking Service

Federico Spini, Enrico Marino, Michele D'Antimi, Edoardo Carra, Alberto Paoluzzi*
Università Roma Tre, Rome, Italy

Figure 1: *Image from real-time navigation of a quasi-photorealistic indoor VR environment. No light or material nodes are in the final scenegraph, but just geometry and textures, automatically baked on a remote Web service via a physically-based render engine. Both scene authoring and navigation are in the browser, without any software or plugin installation.*

Abstract

An interactive user-experience with virtual environments of high visual-quality usually demands either authoring tools of movie-industry level and a server farm, or a native application on the client, or both. Conversely, we introduce here a novel 3D workflow, targeting specifically the largest needs of high-quality VR experience. Our workflow architecture incorporates both authoring and navigation of quasi-photorealistic scenes directly on the Web platform, thus including mobile and VR devices, and is specifically addressed to the common use by the street man. No software installation is needed. Both simplified authoring and exploration tools are Three.js based, and are mutually connected by a Web service that automatically produces high quality textures, which include lighting effects, multiple soft shadows, reflections and refractions, by exploiting the Blender's physically-based Cycles render engine.

Keywords: Web Rendering, Virtual Environments, Graphics Processing, 3D Visualization, Baking Service

Concepts: •**Human-centered computing** → **Interaction paradigms;** •**Computing methodologies** → **Graphics systems and interfaces;** •**Information systems** → *World Wide Web;*

1 Introduction

This work introduces a novel web-based architecture to support both the authoring of synthetic 3D indoor environments and the virtual walkthrough of such scenes with quasi-photorealistic quality. The proposed architecture implements a linear workflow composed by three consecutive steps, namely (i) *authoring*, (ii) *processing* and (iii) *exploring*.

*e-mail: [spini | marino | dantimi | carra | paoluzzi]@dia.uniroma3.it

SIGGRAPH 2016 Posters, July 24-28, 2016, Anaheim, CA
ISBN: 978-1-4503-ABCD-E/16/07
DOI: http://doi.acm.org/10.1145/9999997.9999999

Authoring and/or editing (i) and scene walkthrough (iii) have been implemented as serverless web applications. Processing (ii) — here called *baking* — is performed as a remote web service accessed via HTTP protocol, and is responsible for a variety of off-line computations, performed with the aim of making the scene available for browsers running on commodity hardware, while maintaining a near-photorealistic perceived quality.

In particular, we pursue the near-photorealism of the indoor scene by focusing on special lighting interactions, *i.e.* on accurate soft shadows, light reflections and refractions which, in accordance with [Joon 2010] and [Wang and Doube 2011], return an enhanced perceived quality. Unfortunately, a native WebGL implementation of these techniques, by simply exploiting the graphics pipeline in the browser, results too slow to be executed on most part of the wide range of devices equipped with a browser, and especially on those with limited graphics power. The approach we discuss in this paper aims to overcome this impediment, and to provide a from-good to very-good 3D experience on every kind of device supporting WebGL, *i.e.* on almost the totality of modern computing appliances.

So, holding our the focus on modeling and interactively visiting *indoor* environments, where a critical condition holds, *i.e.* the stillness of the scene, we are enabled to statically precompute, once and for all, the aforementioned light interactions. This process goes under the name of *baking*, and take place during the *processing* phase, storing the result in suitable web objects, ready to be provided on demand. The hypothesis of stillness holds in a variety of use-patterns, and in particular for VR environments used for e-commerce, museum walkthroughs, and many other applications.

The resulting web suite can find several uses, ranging from culture (e.g. virtual museums), to social trade (e.g. enabling people to visit places they are unable to reach), to some prominent commercial use (e.g. real-estate, home staging), to security and serious games applications. The main goal of this project was to broaden the accessibility of 3D synthetic indoor environments, both for designers, which are asked to face the web authoring tool as a simplified CAD editor, and for explorers, which may move inside the scene in manners well tested by the videogames world.

The remainder of this document is organized as follows. Section 2 provides an overview of related work. Section 3 presents the proposed architecture and tools. Finally, Section 4 contains some conclusive remarks and figures future developments.

2 Related Work

Several solutions have been proposed in past years to let "mobile" devices — and even ancient PDAs — render 3D models. Several authors, including [Lamberti et al. 2003; Diepstraten et al. 2004; Quax et al. 2006; Doellner et al. 2012] faced the problem by proposing a remote rendering approach. The common idea was to perform the scene rendering remotely, either on a large server or on a server cluster. Conversely, our approach relies only partially (and on-demand) on remote pre-computations, but the scene rendering is essentially performed client-side.

The work of [dos Santos et al. 2009] specifically addresses the problem of virtual environment rendering on machines with limited graphic computational power, directly from a web perspective. They propose to employ a 3D warping technique for generating views locally in the client, while 3D rendering of the scene is again done remotely, on demand, but the proposed approach has no photorealistic concerns.

Several software products can be found on the Internet, in general orbiting around the interior design industry, and covering — typically only partially — the same problem presented in this work. Both small companies[1,2] and big players like Autodesk[3] but also related players such as Ikea[4], introduced web based simplified CAD softwares, that allow the user to design an indoor space and furnish it by choosing furniture from predefined libraries, in order to produce photorealistic single pictures of the environment. Virtual indoor walk-throughs on the web, with remarkable photorealistic quality, are instead allowed by Shapespark[5]. In this case the authoring process is not in the direct control of the user, since they write internally the widget code for web navigation. CGCloud[6] provides high-quality virtual walk-throughs. In this case the Unreal Engine is used for the rendering, but a platform-specific application must be downloaded and executed on the client. The same apply to the well know Google's Sketchup[7].

Hence, as far as we know, no web-based workflow or interactive framework currently support both the authoring process and the fluid navigation of 3D synthetic indoor environments, ensuring a good perceived quality on rendering large scenes in the browser, even on underperforming devices.

3 System Architecture

The main contribution of this work is the definition of a workflow designed specifically for the web platform, which allows for creating, editing and exploring synthetic 3D indoor environments. The three phases, (i) *authoring*, (ii) *processing* and (iii) *exploring* are designed to be executed linearly, although a feedback loop between phases (i) and (ii) can take place. An iterative feedback may so apply to the design of the 3D scene and, since the baking process may require some long time, depending on the requested quality, the designer may get a quick and coarse rendering preview.

Figure 2 shows the modular components implementing this approach, respectively named *Web 3D Editor*, *Baking Service*, and *Web VR Explorer*. The second component is an HTTP web service, while the others are server-less WebGL-based HTML5 web applications, using the well-known *Three.js* 3D library. This design implies that the system does not require any installation of additional software or plugin in order to be used.

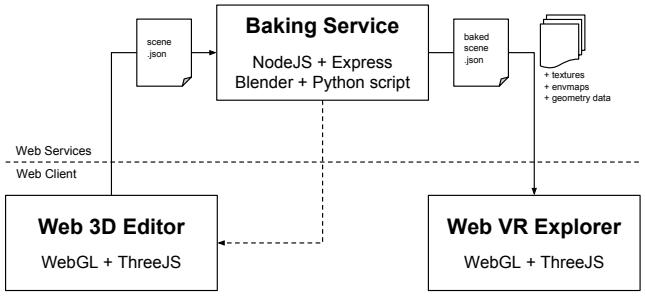

Figure 2: *System architecture.*

Enabling context Indoor scenes are inherently static. Forms of movement could only be introduced by dynamic objects placed in the scene (*e.g.* a running fan). We leave out such kind of objects, thus obtaining a steady scene. Lost of dynamism is not crucial for a good general perception of the surrounding environment. This one is clearly a necessary precondition for a massive a-priori computation of light interactions with scene objects. The a-posteriori introduction on the client of some animated objects in the rendered scene is also possible, and will be further investigated.

Scene Representation The scene produced by the Web 3D Editor is transmitted to the baking service as a JSON document, the data-interchange format of the Web. It consists of a linearized version of the Three.js scenegraph data-structure maintained by the Editor, augmented with some more attributes needed for the baking process. This representation contains both geometry data and texture images as base64 encoded data URLs. As side effect of the baking process, some minor customizations are applied to optimize the bandwidth consumption, relying on the browsers cache. In particular (a) the geometric data are stored in separated binary buffer files, linked via HTTP URL in the JSON document; (b) the image textures and shadow maps are merged together in the so called *lightmapped texture* and again stored separately in binary images file as well as *envmaps* to encode the reflections and refractions. By using binary files directly, we are enabled to leverage the possibility to send and receive binary data using the JavaScript type Array Buffer via XMLHttp Request Level 2.

3.1 Baking Service

The Baking Service is a remote web service that takes a JSON scene representation as its input, computes the lightmapped textures, the envmaps for reflections and refractions and stores the enhanced graphic information in a format that is compliant with the input, so enabling the feedback authoring loop.

Lightmapped Textures This is the term used to denote the enhanced texture image, whose construction is exemplified in Figure 5. The figure shows the texture of a parquet shadowed by desktops and office chairs. The baked lightmap is blended on top of the object texture, thus making pointless a dynamic light computation at runtime, during the interactive exploration of the environment.

Interface The service exposes a simple yet expressive REST-like HTTP API: **`POST`** `/bake`, to request a baking job; **`GET`** `/jobs`, to get info about the currently running jobs; **`GET`** `/jobs/{id}`,

[1] Room Sketcher http://http://www.roomsketcher.com
[2] Floorplanner http://www.floorplanner.com
[3] Autodesk Homestyler http://www.homestyler.com/floorplan
[4] IKEA http://www.ikea.com/ie/en/customer-service/planning-tools/
[5] Shapespark http://www.shapespark.com
[6] CG Cloud http://cgcloud.pro
[7] Sketchup http://www.sketchup.com/

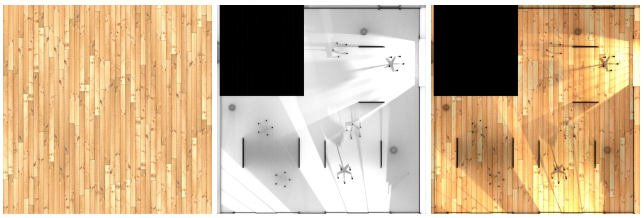

Figure 3: *Lightmapped texture creation.*

to get informed about the state of a particular running job (referenced by the `id`); **`DELETE`** `/jobs/{id}` to stop and delete a running job. A single integer value enclosed in the **`POST`** baking request synthetically controls the expected output quality of the lightmapped textures.

APIs are authenticated and associated to the calling user via JSON Web Token. The requests are handled by a *Node.js* web server, using *Express.js* for its simplicity and clearness in routing requests.

Figure 4: *Comparison between some details with or without the application of the lightmapped textures and reflection envmaps.*

Process The baking process is ultimately executed by Blender, and precisely by its *Cycles* render engine, which implements a physically-based ray-tracing algorithm. This opensource Blender tool can be controlled and configured through Python scripts, and implements out-of-the-box a CUDA-based parallel rendering. Blender and Cycles have a vast and active community of users and developers, strongly orienting our choice of the baking engine.

The Baking Service controls Blender via a Python script. First the input JSON scene is parsed. Since the input is essentially a Three.js scenegraph exported from the Editor, each parsed entity must be translated into a similar Blender's data-structure. Although the Blender expressiveness — for example in terms of applicable materials — is by far larger than the corresponding Three.js one, the mapping between analogous structures can be simply and intuitively defined. We performed an empirical study to identify the best performing mapping of parameter values.

Once the input has been parsed and mapped into Blender, the real baking process can take place: for each object inside the scene the corresponding lightmapped texture is created (or "baked") and stored. When the process is completed the user who sent the job is notified via email, and provided with a link, either to directly access the Web VR Explorer, or to iterate again in the Web Editor. A comparison of texture imagess before and after the baking process is shown in Figures 4. It is easy to appreciate the enhanced realism.

Metrics In the following we provide some indicative metrics to understand the magnitude of processing time. To bake the "Modern Green House" scene, the one shown in Figures 4 and 7, it takes $4\,h$ and $38\,min$ of computation using CUDA drivers, on a machine equipped with a 3.6 GHz Intel Core i7-3820 processor, 16 GB of RAM and NVIDIA GeForce GTX 670 (2 GB of VRAM) as graphic processor. The scene is composed by 206 objects with 8 lights. It is worth noting that it is not unusual for an interior designer or an architect to wait for a time this long during a render stage of a detailed, photorealistic picture. With the same order of magnitude of time our system allows instead for a complete virtual environment exploration.

3.2 Web 3D Editor

The Editor is essentially a simplified CAD server-less web application, including many interaction tools the designer may be accustomed to.

In general, the Editor allows for the following operations: (i) *add/remove* objects (basic objects such as cube, sphere, torus, etc., generated on-the-fly, or complex objects represented in the standard 3D format `.obj`) and lights (ambient light, directional light, point light, spot light) in the scene to build the scenegraph; (ii) *translate/rotate/scale* objects/lights; (iii) *change materials* and specific parameters related to each kind of object/light; (iv) *import/export* the scene in a JSON document; (v) *request* via HTTP the Baking Service to process the scene and make it quasi-photorealistic; *get-info* about the status of a request; or *abort* it.

Furthermore, the Editor may be equipped with a collection of free 3D models, imported in various formats, including `.obj` and `.mtl`, and often representing pieces of furniture, so allowing the designer for exploiting the very large amount of Web 3D repositories of resources.

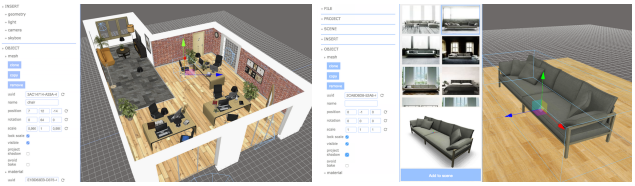

Figure 5: *Images from Web 3D Editor and its catalog.*

3.3 Web VR Explorer

The Web VR Explorer[8] allows to navigate in first person the photorealistic scene, processed by the Baking Service. Virtual tour experience is enhanced by collision detection, which avoids walking through objects and spaces, and enables the viewer to go up and down stairs (see Figure 7, first row). A 2D map can be also switched on to facilitate user orientation in large scenes (see Figure 7, second row). The stereoscopic view for wearable devices (see Figure 8) supports appliances like Google Cardboard.

Our approach to virtual walk-through of inner environments is supported by a static 3D model and allows an unrestricted user interac-

[8] Demo available at http://cvdlab.github.com/bak3d-demo.

tion [Shi and Hsu 2015]. Various techniques have been adopted to maintain a high framerate. To remove the lights in the scene avoids computationally expensive real-time calculation of light effects, since photorealistic lighting informations are directly stamped on textures, resulting in a lightweight render loop. A smart scheme of collision detection drastically reduces the number of objects to be considered for casted rays. The objects of the scene are organized in an octree, dynamically computed only once at scene loading time [Ericson 2004]. Hence the intersection test are yet performed at every frame redrawing, but only against a limited number of objects, *i.e.* those present in the same octree cell of the observer and closer to her.

Metrics In our tests we took as a reference point for consumer standard hardware the MacBook Air 13-inch, Mid 2012. On this machine, "Modern Green House" scene exploration using the browser Google Chrome, is performed constantly at 60 fps.

Figure 6: *"Open Space Office" demo from Web VR Explorer.*

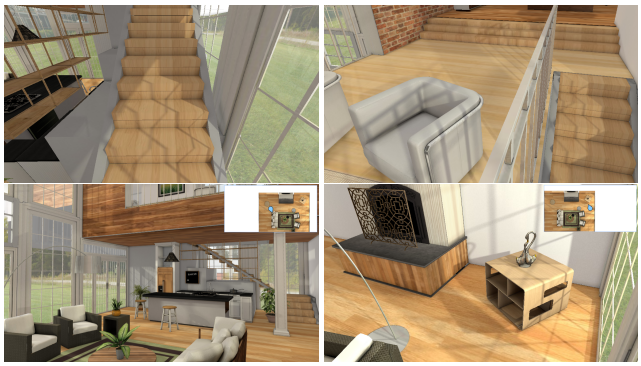

Figure 7: *"Modern Green House" demo from Web VR Explorer.*

4 Conclusions

In this paper we have discussed a strongly simplified visual 3D workflow to provide a high-level user-experience on web clients (desktop, tablet and mobile, reaching wearables like Google Cardboard). Very simple and compact 3D data structures are produced by the editor on the browser, automatically transmitted to a remote web baking service, and given back as a real-time enjoyable VR experience with high realism and frame rate. We are currently providing several optimizations, and are also working to automatize at maximum degree the generation of built environments using the novel LAR (Linear Algebraic Representation) data structure [Dicarlo et al. 2014]. The project discussed here is actually a component of a much larger programme to provide indoor mapping and indoor/outdoor 3D realistic models starting from (a) cadaster documents and/or building drawings, and (b) outdoor/indoor drone flights and their returned set of photographs and generated point clouds. The possible applications range from security enforcing inside small areas and public buildings, to e-commerce, to virtual access to cultural heritage, to serious games, and much more.

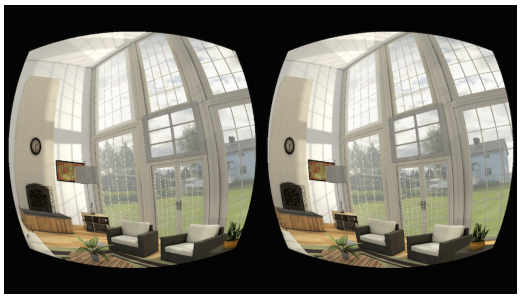

Figure 8: *Web VR Explorer stereoscopic view, enabling the use of VR devices such as Google Cardboard.*

Acknowledgements This work, made in the CVDLAB of the Roma Tre University, was partially supported by grant 2016 from SOGEI, the ICT company of the Italian Ministry of Economy and Finance.

References

DICARLO, A., PAOLUZZI, A., AND SHAPIRO, V. 2014. Linear algebraic representation for topological structures. *Comput. Aided Des. 46* (Jan.), 269–274.

DIEPSTRATEN, J., GORKE, M., AND ERTL, T. 2004. Remote line rendering for mobile devices. In *Computer Graphics International, 2004. Proceedings*, 454–461.

DOELLNER, J., HAGEDORN, B., AND KLIMKE, J. 2012. Server-based rendering of large 3d scenes for mobile devices using g-buffer cube maps. In *Proceedings of the 17th International Conference on 3D Web Technology*, ACM, New York, NY, USA, Web3D '12, 97–100.

DOS SANTOS, M. C., PEDRINI, H., AND BATTAIOLA, A. L. 2009. An architecture for rendering web 3d virtual environments on computers with limited graphic processing power. In *Proceedings of the XV Brazilian Symposium on Multimedia and the Web*, ACM, New York, NY, USA, WebMedia '09, 30:1–30:8.

ERICSON, C. 2004. *Real-Time Collision Detection*. CRC Press, Inc., Boca Raton, FL, USA.

JOON, J. S. 2010. Principles of photorealism to develop photorealistic visualisation for interface design: A review. In *Computer Graphics, Imaging and Visualization (CGIV), 2010 Seventh International Conference on*, 17–25.

LAMBERTI, F., ZUNINO, C., SANNA, A., FIUME, A., AND MANIEZZO, M. 2003. An accelerated remote graphics architecture for pdas. In *Proceedings of the Eighth International Conference on 3D Web Technology*, ACM, New York, NY, USA, Web3D '03, 55–ff.

QUAX, P., GEUNS, B., JEHAES, T., LAMOTTE, W., AND VANSICHEM, G. 2006. On the applicability of remote rendering of networked virtual environments on mobile devices. In *Systems and Networks Communications, 2006. ICSNC '06. International Conference on*, 16–16.

SHI, S., AND HSU, C.-H. 2015. A survey of interactive remote rendering systems. *ACM Comput. Surv. 47*, 4 (May), 57:1–57:29.

WANG, N., AND DOUBE, W. 2011. How real is really? a perceptually motivated system for quantifying visual realism in digital images. In *Multimedia and Signal Processing (CMSP), 2011 International Conference on*, vol. 2, 141–149.