Received October 1, 2014, accepted October 15, 2014, date of publication November 13, 2014, date of current version January 23, 2015.

Digital Object Identifier 10.1109/ACCESS.2014.2370758

Using 3-D Video Game Technology in Channel Modeling

ANDRÉS NAVARRO CADAVID[1], (Senior Member, IEEE), DINAEL GUEVARA IBARRA[2], (Senior Member, IEEE), AND SEBASTIAN LONDOÑO SALCEDO[1], (Student Member, IEEE)

[1]Universidad Icesi, Cali 760031, Colombia
[2]Universidad Francisco de Paula Santander, Cúcuta 540001, Colombia

Corresponding author: A. N. Cadavid (anavarro@icesi.edu.co)

ABSTRACT We present here a paper on the potential use of game engines and graphic technologies for 3-D ray-based technologies used to simulate multipath channels. Our approach harnesses the power of video game development engines to provide an urban 3-D ray-based model for exploration and analysis of multipath channels for a wide frequency range in different complex outdoor and indoor scenarios. Game technologies offer a variety of options for exploiting the capabilities of graphical processing units and provide high performance in computing time with accurate results for channel modeling in current and future wireless technologies. We show the usefulness of this approach using our 3-D ray-based system in different applications.

INDEX TERMS Ray-tracing, game engines, graphic technologies, channel estimation.

I. INTRODUCTION

Ray-tracing (RT) and ray-launching (RL) techniques are widely used to obtain multipath channel parameters in a variety of frequency ranges and operational outdoor and indoor environments. The multipath channel model is the representation of a complex phenomenon involving several mechanisms of interaction between the radio wave and the environment [1]. Different RL techniques allow the characterization of the radio channel in time and spatial domains for the design and implementation of mobile radio [2] and broadcasting systems [3], [4]. Besides, this multidimensional characterization can be used theoretically to evaluate the performance of the multiple input multiple output (MIMO) technologies and orthogonal-frequency division multiplexing (OFDM) transmission scheme which have been taken into consideration in wireless systems development.

In principle, there are two ways to derive a model of the wireless channel. One possibility is to use extensive measurement campaigns, which are time-consuming and expensive. The other is to use models capable of simulating the actual multipath propagation process, such as ray-launching and ray-tracing models [1].

The application of RL techniques to derive multipath channel parameters has two important features. The first is the flexibility of simulation for a wide frequency range and large numbers of scenarios. The other is its ability to identify the different paths in the propagation channel with their main individual characteristics, limited only by the computational capacity available. These properties give researchers the opportunity to validate new techniques or technologies under different scenarios and at specific frequencies, or to understand more deeply the behavior of the radio channel when searching for new techniques to their performance. However, ray launching has some limitations or drawbacks associated with the definition of the scenario. For higher accuracy, a more detailed scenario is required with precise descriptions of the objects in the scenario and a large number of objects like trees, urban furniture, etc. The implementation of such complex scenarios with a large number of objects has some disadvantages: first, they entail large databases because of the level of detail and number of objects with their electromagnetic characteristics; second, high computational complexity and increased computational effort because of the considered effects of reflections, diffractions and scattering; third, increased computational effort because of a high angular resolution and a large number of iterations. In typical ray-based tools the number of iterations is limited to five combinations in total, because of the computational complexity. However, in complex environments, both indoor and outdoor, it is necessary to increase the number of interactions and therefore the computational complexity and time required for doing calculations. Typically, the number of diffractions is limited to two, also because of the computational effort required, but for the most reliable information about the power delay profile, it is necessary to increment the number of diffractions considered in new models.

On the other hand, one of the main limitations of ray-tracing systems is the lack of interoperability between different scenarios, which makes the comparison of results between different ray-based tools difficult.

All these requirements mean a demand for the computing power necessary to make calculations in an adequate time lapse. GPU technology, used in recent years to develop high performance applications in scientific fields and general purposes [6], [7], is a very good candidate for ray- based models and for channel modeling using different numerical techniques, as has been demonstrated by the different applications existing in the field. Initially GPU was developed simply to accelerate graphic processing, but now the most advanced graphic cards have hundreds, even thousands, of processing cores and several gigabytes in their memory, and they are used to implement parallel processing algorithms to achieve high performance. Some authors have used GPU for radio propagation simulations and some RF simulations [7]; however, the use of GPU necessitates programs using specific languages and techniques, as in the case of NVIDIA CUDA [15] and OpenCL [25].

Most of the processing capabilities and physics effects implemented in graphics cards are quite suitable for wireless channel simulation, because most of the illumination and shooting effects highly optimized in most common games are basically the same in concept as the physical phenomena required for radio channel estimation and propagation. This fact and the physical effects implemented in game engines are perhaps the main motivation for this article.

In order to avoid interoperability issues between the different gaming platforms, most game development is based on the use of game engines. Game engines are a suitable platform for providing the flexibility, extensibility and reusability of the algorithms, method and techniques which facilitate the efficient use of GPUs in managing large databases of real models of complex environments and the parallel processing of ray-based techniques [6]. Additionally, game engines typically use standard file formats for the definition of scenarios, facilitating the information interchange of 3D scenarios between different tools.

Currently, there are more than 20 game engines [26], used mainly for the development of games for different OS platforms like Windows, IOS, Android or game consoles like xBox or similar. Another use of game engines is the simulation of real environments to allow a user to experiment many times with the state of a mission or task, as in a game. This kind of application is known as a serious game.

In the case of ray-based channel simulation, it is necessary to have 3D models with all the points, objects and textures needed to be imported or exported in different systems.

In the present article, we discuss the potential of game technologies, especially game engines for multipath channel estimation in complex urban and indoor scenarios. We will show some simulation results obtained with a game-based tool for RL simulation, in order to illustrate the application of the game engine. For this purpose, we use the algorithms, methods and techniques developed in a game engine and the potential and capabilities of a GPU.

This paper is organized as follows. In section II we briefly relate the GPU evolution; in section III we discuss game engines and graphics technology; in section IV we describe some applications of game technologies to radio propagation and channel modeling; in section V we present a proposal for a 3D file format for interoperability between 3D ray-based tools; in section VI we describe some scenarios employed to test a game engine ray-based tool; in section VII we describe new graphics technologies for real-time radio propagation and channel modeling; section VIII concludes.

II. GPU EVOLUTION

According to [21], the development of GPU technology started with the first graphics accelerators, which were developed by NIVIDIA (Riva128 in 1997) and ATI in the mid-1990s. Since that time, evolution has been fast and has given more graphics power following Moore's law. The first GPU developed by NVIDIA was the GeForce 256, launched in 1999, which contains 3 million transistors. The current GPU from this manufacturer, the Fermi, has 3 billion transistors and the latest NVIDIA chip, the Kepler processor, has 7.1 billion transistors.

In 2006, NVIDIA launched the first CUDA system with the GeForce 8800, containing 681 million transistors and 128 CUDA cores. It brought high performance computing to the desktop and increased the 3D graphics capabilities that can be used for ray-based technologies. Current Fermi processors have 512 CUDA cores and TESLA cards, like the K40, and the latest Kepler processors have 2880 CUDA cores.

The evolution of graphics cards was accompanied by that of the capabilities for different graphics processing like shading, color blending, texture, and antialiasing, as well as the number of polygons and physics effects incorporated in the different games. Evolution of 3G graphics technology is associated with the evolution of reference games like Quake 3 and Halo. These games exploited to the maximum the capabilities of graphics technologies.

However, graphics technologies have not only fostered an increasing number of transistors or introduced parallel processing capabilities. Functionalities like physics processing units have been incorporated in graphics cards, and are very useful for channel simulation. Physics processing units will be discussed in section III.

III. GAME ENGINES AND GRAPHICS TECHNOLOGY

A. 3D GAME ENGINES

A game engine is a system designed for the creation and development of video games. It includes a rendering engine, a physics engine for collision detection, and an efficient memory management system. One of the characteristics of the game engine is the efficient implementation of ray-tracing algorithms and the use of GPU power for rendering, regardless of the manufacturer of the GPU. The physics engine is intended to optimally implement effects like optical physics as reflections, refractions and diffraction. It is possible to combine different physics engines with game engines in order to obtain optimal results and reduce computation time.

New technologies like CUDA and OpenCL allow us to exploit the GPU graphics' computational power to improve channel characterization computation time, in order to obtain coverage and channel results in affordable time comparable to semi-deterministic propagation models. Currently, CUDA is a technology independent of game engines and has not been implemented in game engines; however, it is possible to implement such technology in most open source game engines by increasing computing capacity for real-time gaming and visualization.

We used the 3D RT techniques in combination with our urban 3D model to identify propagation parameters and derive multipath channel parameters [8], [9]. Both models are based on game engines. The identified propagation parameters between the transmitter and the receiver are: time delay of arrival (TDoA), full polarimetric transmission matrix, direction of departure (DoD), direction of arrival (DoA), delay spread, among others. Introducing the gains of the transmitting and receiving antenna and their complex directional pattern, we estimated the parameters of: attenuation, depolarization, phase shift, delay, angle of arrival and angle of departure. Thus, this is represented as a power delay profile (PDP). Then, we can use the PDP and its statistics for different developed applications.

B. RAY TRACING IN GRAPHICS AND PHYSICS

In computer graphics, ray tracing is a rendering technique for generating realistic images by tracing the path of light through a 3D scene. A render engine consists of a ''faceless'' computer program that interacts with a host 3D application to provide specific ray-tracing capabilities ''on demand'' [16]. The technique is capable of producing a very high degree of visual realism, usually higher than that of other common rendering methods but at a greater computational cost. Ray tracing is capable of simulating a wide variety of optical effects, such as reflection and refraction, scattering and light scattering. Optical ray tracing describes a method for producing visual images constructed in 3D computer graphics environments, with more photorealism than either ray casting or scanline rendering techniques.

In physics, ray tracing is a method for calculating the path of waves or particles through a system with regions of varying propagation velocity, absorption characteristics, and reflecting surfaces [20]. Under these circumstances, wave fronts may bend, change direction, or reflect off surfaces, complicating analysis. Ray tracing solves the problem by repeatedly advancing idealized narrow beams called rays through the medium in discrete numbers. Recent work shows that raw computational capability has now advanced to the point where it is reasonable to consider using ray-tracing visibility in real-time graphics systems. Also, in the last few years, some efficient ray-tracing algorithms have been proposed in the literature, as well as similar techniques like photon mapping [16].

C. PHYSICS PROCESSING UNIT

Most ray-tracing engines included in game engines are complemented with a physics engine and a physics processing unit (PPU), commonly used in state-of-the-art GPU cards. Some state-of- the-art physics engines like NVIDIA PhysX use the computational power of PPU to reduce the computation time in particle effects, which also can be used to reduce computation time in channel parameter estimation.

A physics engine is computer software that provides an approximate simulation of certain physical systems, such as rigid body dynamics (including collision detection), soft body dynamics, and fluid dynamics, for use in the domains of computer graphics, video games and film. Their main uses are in video games (typically as middleware), in which case the simulations are in real time. The term is sometimes used more generally to describe any software system for simulating physical phenomena, such as high-performance scientific simulation.

A Physics Processing Unit (PPU) is a dedicated microprocessor designed to handle and accelerate the calculations of physics, especially in the physics engine of video games. Examples of calculations involving a PPU might include rigid body dynamics, soft body dynamics, collision detection, fluid dynamics, hair and clothing simulation, finite element analysis, and fracturing of objects. The idea is that specialized processors offload time-consuming tasks from a computer's CPU, much like how a GPU performs graphics operations offloading CPU's load. Initially, this technology was developed by a small company called Ageia, which was acquired by NVIDIA. Currently, PPUs are incorporated in NVIDIA series 8xxx and above graphic cards. NVIDIA uses the commercial term PhysX to refer to an SDK middleware used to accelerate some physics effects in games allowing real-time simulation of physical effects. Some of these effects are the same as those required in ray-tracing propagation models, and therefore propagation models can be improved using technologies like PPU and PhysX combined in a game engine.

D. NVIDIA CUDA

CUDA is the acronym of Compute Unified Device Architecture. CUDA was introduced by NVIDIA in 2006 with the GeForce 8800, featuring the first unified graphics and computing GPU architecture programmable in C with the CUDA parallel computing model, in addition to using DX10 and OpenGL [21]. Unlike previous generations that partitioned computing resources into vertex and pixel shaders, the CUDA architecture included a unified shader pipeline, allowing each and every arithmetic logic unit on the chip to be marshaled by a program intended to perform general purpose computations. Because NVIDIA intended to use this new type of graphics processors for general purpose computing, these ALUs were built to comply with IEEE requirements for single-precision floating-point arithmetic and were designed to use an instruction set tailored for general computation [28].

E. OPENCL

OpenCL is an open standard maintained by the Khronos group with the backing of major graphics hardware vendors as well as large computer industry vendors interested in off-loading computations to GPUs. OpenCL (open computing language) is designed for general purpose parallel programming across CPUs, GPUs and other processors, giving software developers portable and efficient access to the power of these heterogeneous processing platforms [29].

The main advantage of OpenCL over CUDA is that it supports graphics cards from AMD and NVIDIA, as well as embedded platforms; facilitating interoperability for GPU-based applications.

IV. APPLICATIONS OF GAME TECHNOLOGIES IN CHANNEL MODELING

A. POWER PREDICTION

This application used the RT and PDP values to estimate the propagation model and power prediction [10], [11]. Using a game engine, we can reduce computation time from days to hours, as shown in [10] and [11], compared with published results. Also, we can easily implement five interactions, whereas most ray-tracing models use two or three interactions. Results are better than those obtained by using traditional semi-deterministic propagation models, but computation time is still high, making this application unattractive for commercial use.

B. RAY-TRACING MODEL CALIBRATION

This application allows us to adjust the ray-tracing model parameters in the permittivity values by using field measurements. This is achieved by analysis of the behavior of the statistical variation of standard deviation, correlation coefficient and the average error between the values of estimated and measured path loss data [12], [13].

Ray-tracing calibration is not common because of the high computation time of most models. This application solves a typical problem found with ray-based models, as is the lack of information about constitutive parameter values for different materials in complex environments like urban areas. Calibration of constitutive parameters using power measurements is a powerful tool for the acquisition of more real information about building materials which can be used in the simulation of channel parameters with greater confidence.

C. DELAY SPREAD CONTROL HIDING FIRST HIT OBJECTS

This application used 3D RT and the statistics of the power delay profile (PDP) to identify the scatterers from first hit which contribute the most to the channel excess delay. Once identified, these scatterers can be hidden to control the delay spread [14].

This application has the potential to help LTE, LTE-A operators to improve the coverage/capacity mixture in specific areas of a city by means of simulations, before deploying infrastructure. In this way, it is possible to obtain not only coverage information but multidimensional information, required in present and future wireless technologies.

D. CHANNEL DELAY SPREAD

Ray-launching algorithms are suitable for estimating channel parameters.

The most important channel characteristics can be estimated by obtaining the path parameters based on the frequency response $H(f, t)$ and the time-variant channel-impulse response of the channel $h(t, \tau)$.

The path parameters for the propagation between the transmitter and the receiver are defined by $n = 1,\ldots,N(t)$ propagation paths [30]. The identified propagation path parameters between the transmitter and the receiver are:

$\tau_n(t)$: time delay of arrival (TDA) of path;
$\bar{T}_n(t)$: full polarimetric transmission matrix of path;
$\Omega_{T,n}(t)$: direction of departure (DoD) of path;
$\Omega_{R,n}(t)$: direction of arrival (DoA) of path.

At each interaction of the ray with an obstacle, the field strength is multiplied by a dyadic propagation transfer factor, which accounts for the actual propagation effect and for a change in divergence caused by the interaction. Cascading all transfer factors (and therefore all occurring propagation phenomena) leads to the full polarimetric transmission matrix $\bar{T}_n(t)$, which together with the path length (time delay $\tau_n(t)$) characterizes the field strength of the ray. $\Omega_{T,n}(t)$ and $\Omega_{R,n}(t)$ are represented in colatitude and longitude (spherical coordinates). Introducing the gains of the transmitting and receiving antenna G_R and G_T and their complex directional pattern $\vec{C}_R$ and $\vec{C}_T$ the frequency response of the channel is:

$$\begin{aligned} H(f,t) &= \sqrt{\left(\frac{C_o}{4\pi f_c}\right)^2 G_R G_T} \cdot \sum_{n=1}^{N(t)} \vec{C}_R(\Omega_{R,n}(t)) \\ &\quad \cdot \bar{T}(t)\vec{C}_T(\Omega_{T,n}(t))e^{-j2\pi f \tau_n(t)} \\ &= \sum_{n=1}^{N(t)} A_n(t)e^{-j2\pi f \tau_n(t)} \end{aligned} \quad (1)$$

where C_o is the vacuum speed of light and $\mathrm{f_c}$ is the center frequency of the system $A_n(t)$ represents the complex amplitude of the n^{th} multipath component and incorporates the properties of the transmitter and receiver antenna.

The low-pass impulse response of the channel $h(t, \tau)$ is obtained by the inverse Fourier transform of (1).

$$h(\tau, t) = \sum_{n=1}^{N(t)} A_n(t)e^{-j2\pi f_c \tau_n(t)}\delta(\tau - \tau_n(t)) \quad (2)$$

Thus, the channel model could be represented as a power delay profile (PDP) expressed by (2). This is easily estimated by a ray-launching technique implemented with a game engine, using the native ray-launching algorithms implemented in most game engines and optimized in GPUs.

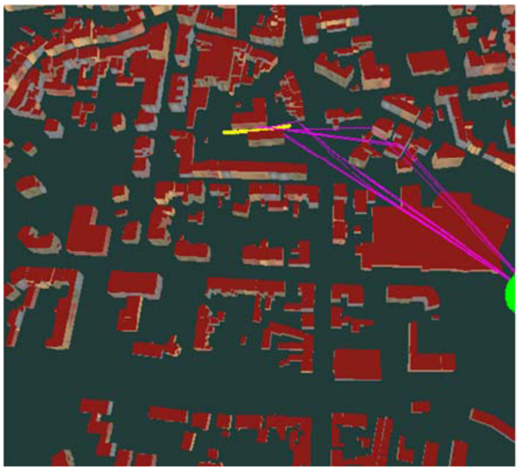


FIGURE 1. Reflected and diffracted rays (fuchsia) for a reception point (yellow).

Fig. 1 shows the results of a ray-tracing procedure using the game engine, with reflections and diffractions, for an estimated reception point. We used parameter values TDA, full polarimetric transmission matrix, DoD and DoA for simulated delay spread and estimated wide-band characteristics of the propagation channel. Our previous work fighas shown that these techniques are appropriate for obtaining multipath parameters with high accuracy and fast processing [8], [9], [11].

Characterization of wideband radio channels in time domain is obtained from the PDP and first and second order moments, the mean excess delay and RMS delay spread.

V. 3D FILE FORMAT FOR SCENARIO INTEROPERABILITY IN RAY-BASED SYSTEMS

The use of game engines and GPU for ray-based systems has an additional and not so sophisticated advantage; the possibility of using existing 3D file formats used in the game community for the definition of game scenarios. In the next paragraphs, we will discuss the requirements and modifications for a standard scenario which can be used in 3D ray-based systems, using game technologies.

This proposal is based on the well-known XML format and the format used by most game engines and 3D tools.

A. XML FORMAT: BRIEF RECAPITULATION

XML is the acronym of eXtensible Markup Language and was developed by the World Wide Web Consortium (W3C) and used to store and manipulate data in an interoperable way. Derived originally from SGML, XML allows grammar from specific languages to be defined to structure big documents. XML also supports data bases and is useful for connecting different applications or integrating information.

1) ADVANTAGES OF XML

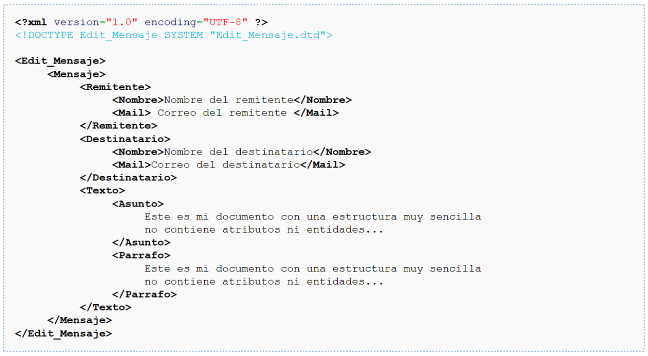
It is extensible: after its design and operation, it is possible to extend XML to add new labels and functionalities.

It is not necessary to create a specific analyzer for each version of XML. It allows the use of any of the available analyzers. In this way, it is possible to develop applications faster and bug free.

It is easy to understand and process, improving compatibility and interoperability between different applications. It is possible to connect applications from different platforms regardless of data sources.

XML transforms data on information, because it gives significance to the data and associates the data with a context.

Fig. 2 shows an example of XML format.



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
     <Mensaje>
          <Remitente>
               <Nombre>Nombre del remitente</Nombre>
               <Mail> Correo del remitente </Mail>
          </Remitente>
          <Destinatario>
               <Nombre>Nombre del destinatario</Nombre>
               <Mail>Correo del destinatario</Mail>
          </Destinatario>
          <Texto>
               <Asunto>
                    Este es mi documento con una estructura muy sencilla
                    no contiene atributos ni entidades...
               </Asunto>
               <Parrafo>
                    Este es mi documento con una estructura muy sencilla
                    no contiene atributos ni entidades...
               </Parrafo>
          </Texto>
     </Mensaje>
</Edit_Mensaje>
```

FIGURE 2. XML example.

2) XML DOCUMENT PARTS

Prologue: it is not mandatory and indicates XML version, document type, comments.

Body: shows that the body of the document only contains root element.

Elements: can contain more elements, characters or both.

Attributes: indicates special characteristics and allows definition.

Predefined entities: entities are designed to represent special characters which do not need to be decoded by the XML parser.

CDATA sections: it is an XML construction designed to specify data using any character not to be interpreted as a marker.

Comments: information to the programmers about the document.

B. XML SUPPORT IN GAME ENGINES FOR 3D SCENARIOS

In the gaming world XML is commonly used for the definition of 3D scenarios in order to allow interoperability between different game engines. In Table I, a summary of game engines and their support for XML is shown.

As shown in the table, commercial game engines do not natively support XML. However, open source engines do natively support XML and all engines support OBJ. It is quite

TABLE 1. Engines with XML support.

Game engine	Unity	CryEngine	Unreal	Jmonkey	Ogre3D
Version	Commercia l	Commercial	Commercia l	OpenSource	OpenSource
XML native support	N	N	N	S	S
OBJ support	S	S	S	S	S
Requires third-party software to convert from XML (Maya, 3DS Max, Softimage, Blender)	S	S	S	N	N

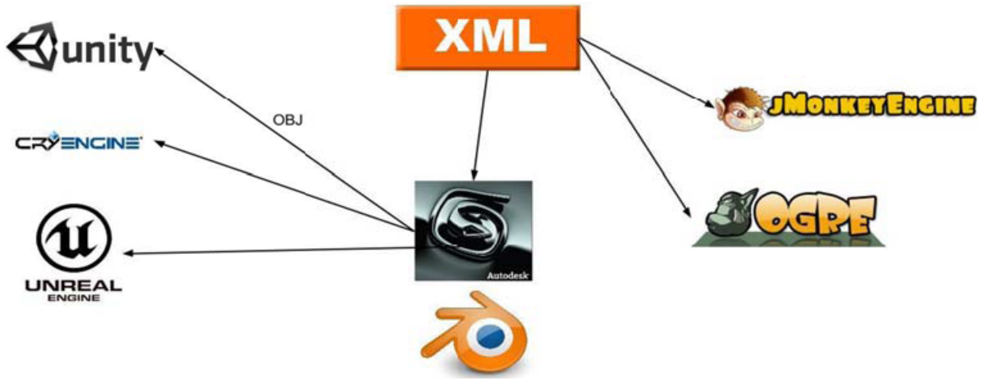


FIGURE 3. Information flow for different engines and 3D tools.

easy to convert from XML to OBJ using 3D design software like Blender.

C. PROPOSED FORMAT FOR 3D SCENARIOS IN RAY TRACING

As shown in the table , all game engines support the OBJ format natively, because it is the common format in the gaming world which supports mesh objects, as well as surfaces and bounding for physics engines.

For ray-tracing engines for channel simulation, we propose the use of OBJ type objects, in order to allow interoperability between different ray-tracing platforms. The initial structure of the proposed scenario is as follows:

main entity - mesh
sub-entities – submeshes

```
    <mesh>
     <submeshes>
       <submesh material="table"
usesharedvertices="false" use32bitindexes="false"
operationtype="triangle_list">

<materialPropagation = “0.21”>
       <faces count="904">
         <face v1="0" v2="1" v3="2" />
         <face v1="0" v2="3" v3="1" />
         <face v1="3" v2="4" v3="1" />
</submesh material>
</submeshes>
</mesh>
```

The XML fragment above shows how the proposed XML format includes material as a descriptive element of the 3D, with the possibility of including constitutive parameters in any standard 3D scenario. In terms of the existing systems for developing 3D scenarios, the constitutive parameter is optional and does not affect the standard model used in 3D tools.

In Fig 3, the process of converting from XML to 3D tools or vice versa and exporting to OBJ format is shown. This process allows generating 3D scenarios from 3D tools, exporting to XML and using the scenario in different ray-tracing tools, using OBJ or proprietary formats used by existing ray-tracing tools.

For example, in Fig 4 we show a 3D indoor scenario developed in Blender and exported to OBJ.

In Fig 5, we show another indoor scenario, modeled in Blender and exported to OBJ but with more details than the scenario from Fig 4. In this scenario, we have included windows, door, a projection screen and a lot of details of chairs.

VI. SCENARIOS

We have used the urban Cost 2100 Cali Realistic Reference Scenario and a sub-urban scenario obtained from the main campus of the Polytechnic University of Valencia, Spain [12]. In the Cost 2100 Cali Reference Scenario we simulate mobile telephone cellular services at a frequency of 900 MHz and DVB-T2 services at a frequency of 479 and 473MHz. In Valencia Scenario we simulate DVB services for mobile terminals at a frequency of 496 MHz.

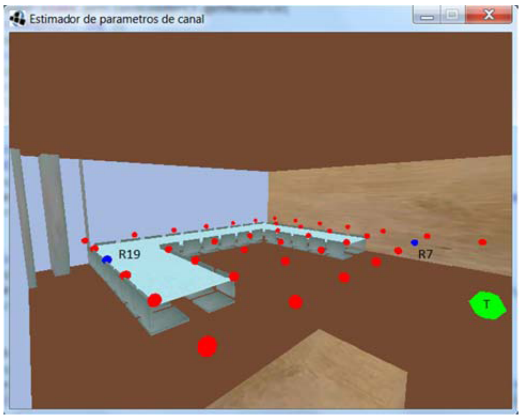


FIGURE 4. **3D scenario built in Blender and exported to OBJ.**

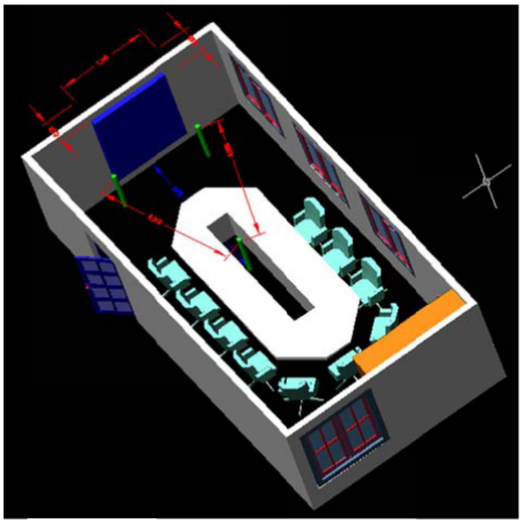


FIGURE 5. **Detailed model of an indoor environment in OBJ format.**

Also, we have performed simulations using the Cost 2100 Ilmenau Realistic Scenario (Fig. 1) and an Indoor Scenario in Valencia (Fig. 4) [31], using channel measurement information to compare simulation data and measured data.

VII. NEW GRAPHICS TECHNOLOGY AND ITS USE IN CHANNEL MODELING

New developments in game technologies and GPU processing power can be used to improve the capacity for channel simulation and simulation.

Technologies like Nvidia Optix or similar open sources like OpenRT or PovRay, or PhysiX engine, would increase the computational capacity for ray-tracing techniques not only for the game industry but also for real-time radio propagation and channel modeling.

The NVIDIA OptiX Ray-tracing Engine [19] is a programmable ray-tracing framework that helps software developers to build ray-tracing applications in a fraction of the time of conventional methods, running exceedingly fast on NVIDIA GPUs. OptiX can be extended beyond image creation by enabling rays to gather and carry custom payloads. The data fed to OptiX are also programmable, enabling custom shading techniques, programmable intersection for procedural definitions, and programmable cameras for customized ray dispatching. This flexibility enables OptiX to accelerate ray-traced rendering algorithms ranging from the highly interactive to the ultra-realistic, while also accommodating disciplines such as acoustics, ballistics, collision analysis, radiation reflectance, and volume calculations - wherever intensive ray-tracing calculations are employed. It can be incorporated in many game engines and exploit the parallelization capabilities of CUDA technology for real-time ray-based channel estimation.

For most people from the world of propagation and channel modeling, these new technologies mean a change of paradigm and new programming skills. Perhaps it will be necessary to involve people from the IT community with experience in game engines and parallel programming in order to fully exploit such game technologies in radio channel issues for channel modeling.

VIII. CONCLUSIONS

Game technologies have an important role in the development of new tools for real-time channel estimation. The combination of game engines and GPU technologies and architectures like CUDA and OpenCL provides powerful tools and scripting for GPU exploitation.

Using Open Source 3D Game Engines, it is possible to exploit available technologies like NVIDIA Optix for real-time ray-tracing/launching, NVIDIA PhysX for physics simulation and CUDA or OpenCL, in order to improve computation time and obtain real-time radio propagation and channel modeling for the 4G and 5G wireless technologies.

The combination of such techniques with better ray-tracing algorithms implemented in game engines and parallelization techniques would permit real-time channel simulations for future wireless systems, as well as multidimensional coverage/capacity/channel maps in 3D for both indoor and outdoor and HetNet environments.

REFERENCES

[1] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.
[2] L. M. Correia, *Mobile Broadband Multimedia Networks: Techniques, Models and Tools for 4G*. Oxford, U.K.: Elsevier, 2006.
[3] Y. Corre and Y. Lostanlen, "Characterization of the wideband wireless channel in the context of DVB systems," in *Proc. IEEE 19th Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Cannes, France, Sep. 2008, pp. 1–5.
[4] *Digital Video Broadcasting (DVB); Implementation Guidelines for a Second Generation Digital Terrestrial Television Broadcasting System (DVB-T2)*, document ETSI TS 102 831 V1.2.1, 2012.
[5] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, "A survey of commercial & open source unmanned vehicle simulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 852–857.
[6] A. Leist, D. P. Playne, and K. A. Hawick, "Exploiting graphical processing units for data-parallel scientific applications," *ACM J. Concurrency Comput., Pract. Exper.*, vol. 21, no. 18, pp. 2400–2437, Dec. 2009.
[7] G. de la Roche, A. Valcarce, D. López-Pérez, E. Liu, and J. Zhang, "Coverage prediction and system level simulation of WIMAX femtocells," in *Proc. Minutes 6th Cost 2100 Manage. Committee Meeting*, Lille, France, Oct. 2008.

[8] A. N. Cadavid and D. G. Ibarra, "Using game engines in ray tracing physics," in *Proc. IEEE Latin-Amer. Conf. Commun. (LATINCOM)*, Bogotá, Colombia, Sep. 2010, pp. 1–5.
[9] A. Navarro and D. Guevara, "Using game engines for wideband channel estimation parameters in Andean cities," in *Proc. 4th Eur. Conf. Antennas Propag. (EuCAP)*, Barcelona, Spain, Apr. 2010, pp. 1–5.
[10] A. Navarro and D. Guevara, "Applicability of game engine for ray tracing techniques in a complex urban environment," in *Proc. IEEE 72nd Veh. Technol. Conf. Fall (VTC-Fall)*, Ottawa, ON, Canada, Sep. 2010, pp. 1–5.
[11] A. Navarro, D. Guevara, N. Cardona, and J. J. Gimenez, "DVB coverage prediction using game engine based ray-tracing techniques," in *Proc. IEEE 74th Veh. Technol. Conf. (VTC Fall)*, San Francisco, CA, USA, Sep. 2011, pp. 1–5.
[12] A. Navarro, D. Guevara, N. Cardona, and J. Lopez, "Measurement-based ray-tracing models calibration in urban environments," in *Proc. IEEE Antennas Propag. Soc. Int. Symp. (APSURSI)*, Chicago, IL, USA, Jul. 2012, pp. 1–2.
[13] A. Navarro, D. Guevara, and M. V. Africano, "Calibración basada en medidas para modelos de trazado de rayos en 3D para ambientes exteriores urbanos andinos," *Rev. Sistemas Telemática*, vol. 10, no. 21, pp. 43–62, 2012.
[14] A. Navarro, D. Guevara, D. Tami, and N. Cardona, "Delay spread control hiding first hit objects," in *Proc. 7th Eur. Conf. Antennas Propag. (EuCap)*, Gothenburg, Sweden, Apr. 2013, pp. 1870–1873.
[15] *What is CUDA?* [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html, accessed Sep. 15, 2014.
[16] D. Buck. (Aug. 2001). *What is Ray-Tracing?* [Online]. Available: http://wiki.povray.org/content/Documentation:Tutorial_Section_1#What_is_Ray-Tracing.3F
[17] *Yafaray—Home*. [Online]. Available: http://www.yafaray.org/, accessed May 2, 2014.
[18] *Gforce/Physx/Technology*. [Online]. Available: http://www.geforce.com/hardware/technology/physx/technology, accessed May 2, 2014.
[19] (2014). *NVIDIA OptiX Ray Tracing Engine*. [Online]. Available: http://developer.nvidia.com/optix
[20] (Sep. 15, 2014), *Ray Tracing (Physics)*. [Online]. Available: http://en.wikipedia.org/wiki/Ray_tracing_(physics)
[21] J. Nickolls and W. J. Dally, "The GPU computing era," *IEEE Micro*, vol. 30, no. 2, pp. 56–69, Mar./Apr. 2010.
[22] K. Karimi, N. Dickson, and F. Hamze. (May 16, 2010). "A performance comparison of CUDA and OpenCL." [Online]. Available: http://arxiv.org/abs/1005.2581v3
[23] A. González *et al.*, "The impact of the multi-core revolution on signal processing," *Waves*, pp. 74–84, 2010.
[24] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proc. IEEE*, vol. 96, no. 5, pp. 879–899, May 2008.
[25] P. Du, R. Weber, P. Luszczek, S. Tomov, G. Peterson, and J. Dongarra, "From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming," *Parallel Comput.*, vol. 38, no. 8, pp. 391–407, 2012.
[26] (2012). *100 Highest Rated Game Engines—Mod DB*. http://www.moddb.com/engines/rated, accessed May 18, 2014.
[27] (2007). *XML—Wikipedia*. http://es.wikipedia.org/wiki/Extensible_Markup_Language, accessed May 18, 2014.
[28] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Reading, MA, USA: Addison-Wesley, 2010.
[29] A. Munshi, Ed., *The OpenCL specification. Version: 1.1. Document Revision: 44*. Beaverton, OR, USA: Khronos Group, Jun. 2010.
[30] M. Narandžić *et al.*, "On a characterisation of large-scale channel parameters for distributed (multi-link) MIMO—The impact of power level differences," in *Proc. 4th Eur. Conf. Antennas Propag. (EuCAP)*, Apr. 2010, pp. 3–7.
[31] A. Navarro, D. Guevara, N. Cardon, and J. Gimenez, "Using 3D game engines and GPU for ray launching based channel modeling in indoor," in *Proc. 31st URSI General Assembly Sci. Symp.*, Beijing, China, Aug. 2014, pp. 1–4.

ANDRÉS NAVARRO CADAVID (M'95–SM'11) received the Electronic Engineering degree and the master's degree in technology management from Universidad Pontificia Bolivariana, Medellín, Colombia, in 1993 and 1999, respectively, and the Ph.D. degree in telecommunications from the Universitat Politécnica de Valencia, Valencia, Spain, in 2003. He is an Advisor of the National Innovation Program on Electronics, Telecommunications and Informatics with Colombian R&D + I System and the Spectrum Management Committee for Colombian Spectrum Agency. He has been the Director of the i2t Research Group with Universidad Icesi, Cali, Colombia, since 1999. His research interests are spectrum management, radio propagation, and m-health.

DINAEL GUEVARA IBARRA (M'04–SM'14) received the B.S. degree in electrical engineering from the Universidad Industrial de Santander, Bucaramanga, Colombia, in 1989, the M.S. degree in engineering telecommunications from the Universidad Nacional Experimental Politécnica Antonio José de Sucre, Barquisimeto, Venezuela, in 2006, and the Ph.D. degree from Universidad Pontificia Bolivariana, Medellín, Colombia, in 2012.

He is currently a Professor of Electronics Engineering with the Department of Electrical and Electronics, Universidad Francisco de Paula Santander, Cúcuta, Colombia.

SEBASTIAN LONDOÑO SALCEDO (M'08) received the System Engineering degree from the Universidad Icesi, Cali, Colombia, in 2010, where he is currently pursuing the master's degree in research. He was the World Champion for Microsoft Competition EmbeddedSpark (2011), a Mentor at Colombian program Apps.co for entrepreneurs (2013), and is also a Junior Researcher of the i2t Research Group with the project SafeCandy. His research interests are UAVs, serious games, security in android, and neurodegenerative diseases.