Open Source 3D Game Engines for Serious Games Modeling

Andres Navarro, Juan Vicente Pradilla and Octavio Rios
Universidad Icesi,
Colombia

1. Introduction

In this chapter we will review some tools and open source Game Engines used for modeling of real scenarios in serious games for training. One of the typical uses of serious games (3D serious games) is specialized training in dangerous tasks or when the training is quite expensive. However, typical games use artificial scenarios, created by artists and created according to the restrictions imposed by the Game engine used.

In our experience, some tasks require the use of a real scenario like a city, forest area, etc, and most of this information is available as Digital Terrain Models in Geographic Information Systems (GIS). The problem here is that GIS formats are not compatible with 3D formats used in Game engines. Then we have to solve the problem of convert the GIS format to a 3D format supported by the Game Engine.

On the other side of the story, there are different Game Engines in the market and different 3D formats both in the commercial world and in the Open Source world. For working with Open source tools, we have to consider tools like Blender and Game Engines like OGRE or JMonkey. Using these tools, we can model real scenarios based on GIS information, but being aware of some important considerations that will be exposed in the following pages.

During the writing of this chapter, many of the commercial game engines available have released some free versions, in some cases releasing source code for some applications. This is a very important event, because the users can develop serious games and training games using powerful tools according to specific needs. This chapter focuses in free game engines, which may be closed or open source (the code is provided with the software for modification or extension).

In the next pages we will describe some aspects of Serious Games and its importance in this modern time, and then we explain the basics of Open Source Game Engines and the recently released free Game Engines, its characteristics and some basic information about them. After that, we will explain some considerations about the conversion of Digital Terrain Maps in GIS format to 3D models useful for Game Engines. Finally, some conclusions and remarks.

2. The concept of serious games

Serious games have always been an important aspect of human development. They contribute to the acquisition of skills and knowledge that can be applied to daily activities and used to deal with the regular challenges of life.

The game as such doesn't aim anything but "the pleasure of being played" (Huizinga, 1938) any other collateral effect is beyond its goal.
The "serious game" makes reference to any game with a different goal than playing fun. Therefore, when a playful experience is used to teach or train, it can be considered a serious game. This particular kind of games take the consequences in great account, they are training process, development guides (Bruner, 1989).
The concept has received greater attention thanks to the growing interest of education centers for the use of computer games to teach. Nowadays is a common trend for companies to train their employees with help of scenarios simulated with computer games. Teaching institutions are using virtual environments to better impart concepts and procedures.
This is an area where 3D game engines offer a number of tools to achieve greater impact in a multimedia serious game. They support detailed visual representations and real-time interaction with some parts of the game.
They add a multisensory perception of objects that are difficult to access in the real life, and they can shed some light on complex concepts represented as familiar objects at game level.

3. Game engines

Game engines are toolkits aimed to ease the development of videogames, acting as a super-structure of several development efforts. They are also normally packed with a set of tools to be used in the design and coding stages.
A regular game engine provides: scripting, imagery rendering, artificial intelligence, physics, animation, cinematic, network access and resource management.

- Scripting: Let developers to write little pieces of code to control certain parts of the game.
- Imagery rendering: It's the core of visual part of the game. Handles lights, shadows, ray tracing, and rendering of 3D objects.
- Artificial Intelligence: Brings the world and characters of the game to life, through a set of routines that makes possible the interaction with the game environment.
- Animation: Adds behavior to objects, through transformations, skeletons, deformations and dynamics.
- Physics: Provides realistic interaction between objects and with their environment. Plus, character moment, fluid simulation, and "soft bodies".
- Cinematic: Adds the possibility to include video within the game to capture the attention of the player.
- Network access: Provides support to deploy the game in a network environment, be client-server or peer-to-peer.
- Resource management: A fundamental issue for the game engines is the efficient use of the computer resources (CPU, graphics card, memory, storage, hardware) and the load of game related resources (animations, shaders, 3d objects, sound, etc.)

Game engines can be classified according to several criteria, being one of them the type of licensing: commercial and freeware.

4. Open Source Game Engines

There exists different Open Source Game Engines, but perhaps the most popular one is OGRE. Ogre is an open source Game Engine written in C, with different extensions and characteristics.

Other interesting Game Engine that we will revise in deep is JMonkey. This is a Java based Game Engine with similar characteristics as Ogre, but more friendly with some new engineers' generation, familiarized with Java and with the advantages that Java offers.
Currently, serious games are being developed with aid of modeling and simulation tools, in hope of touching the players in a deeper way. This enlarged impact may largely contribute to the goals of the game, while keeping a pleasant visual environment and realistic interaction.

4.1 Jmonkey

JMonkeyEngine – jME (current version 3 alpha 4). It's a free and opensource engine distributed in the terms of BSD licence. It can be used to create games that run in any platform with a Java Virtual Machine (Windows, MAC OS y/o Linux.). It uses a few *native* libraries to improve performance through the Java Native Interface (JNA).
Games are coded in the Java programming language using the Java Standard Edition (J2SE) libraries.
jME is built with a modular architecture on top of OpenGL (Open Graphics Library), version 2, although the development roadmap points for a change to version 3 in the forth version of the engine. The combination of these two technologies offers a huge potential in terms of independence and functionality.
jME provides full scale physics support using jBullet. A shaders pack fully integrated with the engine, along with several other features, offering stunning graphical possibilities.
It also features an integrated system for the creation of Graphic user interfaces (GUI's) using XML (eXtensible Markup Language) through the Nifty GUI, and an advanced resource manager which comes handy to organize materials, textures, models, GUI's, and sound used in the game.
Finally, jME comes bundled with an Integrated Development environment (IDE) based on the Netbeans platform, for easier asset handling, terrain creation, 3D models manipulation, shaders creation, and game coding in Java.

4.2 Ogre (just a render engine)

Web page: http://www.ogre3d.org/
Platforms: Windows, Linux y Mac
Licensing: MIT License
Free: yes
Opensource: yes
None official tool.
Programming language: C++
Alternative programming languages (ports): java (ogre4j), .Net (MOGRE), GM (GMOgre3D), Phyton (Phyton-Ogre)

4.2.1 Ogre3d (just the rendering engine)

Object- Oriented Graphics Rendering Engine - Ogre 3D is not a fully flagged game engine, but a 3D graphics toolkit. Nonetheless, given its relevance in the game developers community, has been included in this chapter.
Ogre 3D is scene oriented and it's coded in C++. The engine provides Libraries to avoid the difficulties associated with the use of OpenGL and Microsoft's Direct3D. Graphic User

Interfaces maybe created in an object oriented manner with the engine's API. Games are coded in C++, but ports had been made for other programming languages such as java (ogre4j), .Net (MOGRE), GM (GMOgre3D) y Phyton (Phyton-Ogre).
Ogre 3D is freeware and open source, it's licensed under the terms of MIT license, and has an active community around its use and development, the result is a continuous improvement. It has a large user-base that creates games for Linux, Mac OS and Windows.
Finally, the engine offers many plugins, provided by the open source community, for quick development of applications. Some of them:
Ogremax (visualization y exporting), MyGui (GUI creation), CrazyEddie's GUI system (GUI widgets creation), OgreSpeedTree (creation of trees and nature elements), NeoaxisEngine (multipurpose 3D engine), ParticleUniverse (a full-flagged system for particle based effects).

4.3 Unity

Web page: http://unity3d.com/
Platforms: Web, iOS, Android, Windows, MAC, Wii, Xbox360 y PlaySatation3
Licensing: Propietary
Free:Yes, the plug-in for end user. Paid for developers
Open source: No
Programming language: C#, javascript, Boo
Additional tools: IDE
Alternative programming languages (ports):
Unity (versión 3), advanced game engine with focus in games with complex graphical content. It offers a high quality editor for design and coding of the game. It support development of games in several platforms: Web (plugin), iOS, Android, Windows, MAC, Wii, Xbox360 y PlaySatation3.
Unity uses a pipeline based deferred rendering method for improved performance. It comes packed with over 100 shaders, ranging from the classics (diffuse, glossy, etc) to the top-notch kind (SelfilluminatedBumpedSpecular)
The engine features a brand-new technology called *surfaceshader*, that permits the creation of new shaders from the scratch, to be used in different applications and renders. This allows the user, by example, to generate an illumination pattern for his game, and being able to reuse it in another.
To minimize computer time for the readers, Unity uses batch processing. The render combines different geometries into parallelizable units; this reduces the load on the graphic drivers and increases flexibility. It is also optimized to work with OpenGL, allowing the user to use shaders on mobile devices.
Besides, the engine features OcclusionCulling, a technology developed by Unity in conjunction with Umbra to function on the Web, mobile devices and game consoles. It reduces the load reducing rendering objects, generated new ones on demand.
Deferred rendering has been added in version 3, it allows the handling of multiple illumination patterns in an efficient way, without the inconvenient of overload.
Real time shadowing is an extremely demanding task for CPU's and graphic cards. The engine uses state of the art techniques to balance the load in a manner it can be processed gracefully by not so last generation equipment.
Unity has a number of tools for modeling and terrain generation within the editor. It can completely generate a terrain from scratch including elements such as trees, bushes, rocks and many types of grass, which give games a much realistic feel.

In the physics department, the engine is based on Nvidia's PhysX, a specialized physics engine that let game creators to focus in design and interaction. It packs default physics for characters and vehicles.
This platform provides last generation tools, which make use of technologies such as FMOD, used to create audio in an interactive way. Application audio can be visualized in real time, filters are provided to improve quality of the final result.
Unity supports 3 scripting languages: Javascript, C#, and a Python dialect called Boo. The three of them can coexist with each other, and make use of .Net libraries for database access, regular expressions, XML, and such. As a high level language, Boo provides syntax for fast prototyping and development of actions an behaviors of objects within the application.
Unlike other engines, Unity offers effortless web deployment, thorough real time network processing, synchronization and remote procedure calls. Multi-player mode is an issue already solved by the engine, which provides a plug-in (add-on) to export the application to any modern web browser, offering unprecedented distribution and interoperability.

4.4 UDK – Unreal Development Kit

Unreal Development Kit - UDK is property of Epic Games Inc, is the well-known game engine behind games such as: Gears of War, X-Men Origins: Wolverine y Unreal Tournament 3. Version 3 has come as a surprise for the game development community, as it will be distributed under a non-commercial free to use license, and with a profit share license for commercial purposes.
UDK is one of the most advanced game engines in use today, and has plenty of tools for easy creation of astounding games, which are coded in the UnrealScript programming language, and can be run on the Xbox/Xbox360 and PlayStation2/3 consoles, and in any computer with Windows, Linux or MAC, and by mobile devices with iOS and Android operating systems.
For animation, the engine uses an "animated skeleton" approach, aiming for detailed control of muscles and movement of characters. It uses a multithreaded processing system called Gemini, making possible the creation of a large number of environment elements with high quality photo-realistic effects. If the application is run on a 64 bits machine, an optimized pipeline rendering system named HDR is used to improve performance.
Physics within the engine are provided by Nvidia's PhsyX, one of the most advanced implementations in the market. It also features illumination and shadowing by the UnrealLightmass application. Kinematics are offered in cinematographic like style by the UnrealSDK, presenting a very realistic game experience. Network layer provides measurement of the gamers game-playing and statistics, providing the necessary information for cooperative, multi-player online deployment.
Finally, UDK features a behavior editor, UnrealEd, for rapid development of complex environments. User can modify terrains, trees and other elements, including sounds and complete scenes of the game.

4.5 Cryengine 3

Web page: http://crytek.com
Platforms: Windows, Xbox 360, PlayStation 3
Licensing: Free for non commercial use.
Free: YES

Open source: NO
Programming language: YES
Additional tools: YES
Alternative programming languages (ports): YES

The CryEngine is property of Crytek Inc, and is one of the most complete and awarded engines available, providing top-notch functionality for the games developed with this technology. It's considered a next generation solution for game development, able to use scalable computing technology. Is expected for version 3 to be free for outside development, but the licensing and business model has not been disclosure yet.

CryEngine technology is build on top of a "sandbox" that permits real time adjustment of parameters and error correction, and optimization as well. The sandbox allows creation and control of the application in real time across multiple platforms in a principle called WYSIWYP (what-you-see is what-you-play). Applications can be deployed directly to the Xbox360 and PlayStation 3 consoles, and to the Windows powered PC's.

CryEngine has a very intuitive interface, which allows closely observing and controlling the event flow in a visual manner, largely avoiding the need for ground zero coding of the application. Generation of terrain and vegetation is carried on by a set of tools that focus in realism and quality.

A real time particle system permits the creation of complex explosions in a few steps, alongside the FX editor, it allows for quick creation of high quality and graphically demanding content. Terrains and rivers can be created with dedicated tools within the engine, and they can be smoothed to any degree, depending on the desired realism level.

The engine offers support for modern multicore computer architectures, largely improving performance through the balanced distribution of the graphics, physics, artificial intelligence, and network tasks across process an threads as well.

Shaders used by CryEngine have been optimized in low level languages, and are compiled and assembled for every specific platform supported. Among the additional tools offered by the engine, we can found EyeAdaptation, that provides a much more realistic feel by the lighting of the scene according to the human eye movement; and High DynamicRange (HDR) Lighting, which permits realistic rendering of high contrast scenes.

Stands out in the CryEngine the generation of very dynamic renders of game characters, the animation is based on a simple skeleton scheme, which permits the creation of very realistic individual moves.

Artificial Intelligence used by the engine response adapting to the given scenario. Dynamic programming is held inside the sandbox, and permits to see changes made to code while the game is running.

Another signature feature of the engine is the extreme realism of water based environments (oceans, rivers, lakes) and high performance of lighting on different environments and natural habitats.

The engine provides a set of tools to measure and evaluate the performance of the applications powered by the engine in a detailed and consistent manner.

The engine has an integrated professional sound edition solution, which permits the edition of ambient soundtrack, event driven sounds and other media, in a very interactive and powerful manner, with a time line familiar to professional sound editors.

Finally, CryEngine offers a number of plug-in and tools that runs on 64 bits architectures, further improving performance. This Engine was released in August 2011 for free use for non-commercial purposes. Source code can be downloaded under specific agreements with Crytek.

4.6 Our recommendation: JMonkey

Selection process between Open Source Game Engines is quite complicated at this time tan some years ago when there were not many options. At this time (2011), the process is quite complicated, but we will try to explain some considerations that we use for our personal selection.

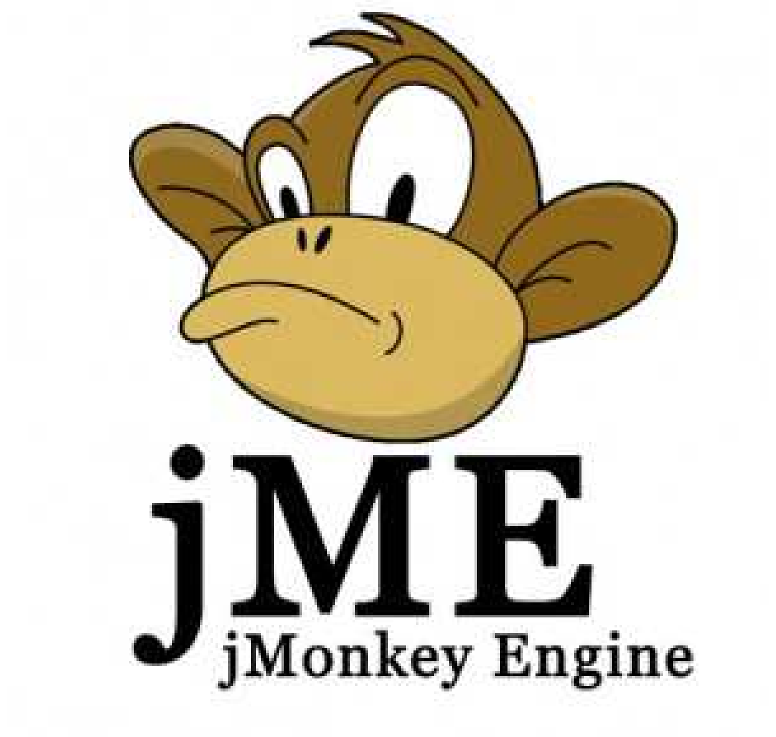


Fig. 1. The JMonkey logo

The choice between an Engine and another requires an exhaustive weighting of factors like: licensing, possibility of code modification, commercial expectation for the development, client computers specifications, platform (Operating System, Console, etc), skills of the development team, Engine support tools, development time, stability of the Engine and target public. In the previous section we have described some Game Engines and its characteristics that can be useful for decision making.
Considering that at the moment of write this lines CryEngine3 has not been released to open source and UDK have only few months under free delivery, we consider some useful aspects for serious games development using JMonkey.
JMonkey is written in Java, giving it a big span in the desktop world, allowing the development for different platforms. The con is that JMonkey does not support consoles or mobile devices. On the other side, is an open source and free game engine with excellent possibilities for Serious Games development.

5. An example of serious game with JMonkey

COMCITY is a simulation and strategy game based on a future chaotic situation where the player has the role of a radio planning agent from an specialized international agency. The player has the task of solve some communications problems occasioned by natural disasters. By means of this interaction, the player learns about the behavior of propagation models and the fulfillment of some basic KPI. The game has been developed especially for technicians, undergraduate students and some decision level personnel with no knowledge of radio planning.

COMCITY could be considered a serious game, because it has been conceived as an educational game; it is supported in real environments and has a mix between reality and entertainment. In COMCITY, game engines capabilities are fully exploited in order to obtain a dynamic environment, with a rich experience, and, at the same time, introduce the student to complex concepts related with wireless planning. In Figure 1, some 3D models used in COMCITY are shown.

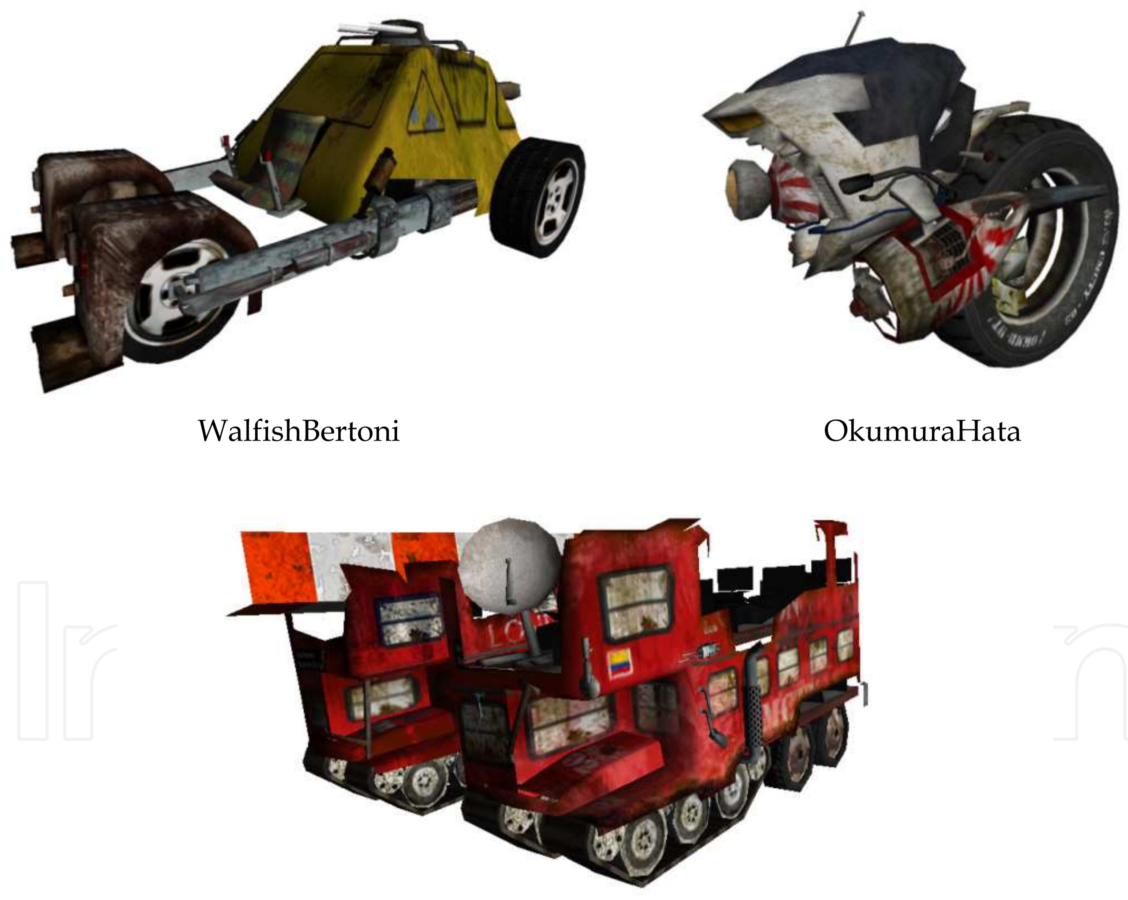


Fig. 2. 3D models for Comcity

TEST is a series of "virtual learning objects" running on a learning platform that combines serious games with training software. The goal of the system is to train people for a specific job and evaluating the most qualified people for the field job. TEST is in part an application of Comcity for specialized training.

In this serious game, we try to reproduce real situations that the apprentice will face in their real activities, preparing them to recognize potentially dangerous situations. Each activity that the player executes is associated with typical activities performed in real situations in remote areas where the trainee will work in a future, if the training is successful.
In this way, the player (trainee) makes associations between real objects that will find in a real situation, but through simulated situations in a 3D environment. In Figure 2, a 3D/2D interface for TEST is shown implemented in the Game Engine.

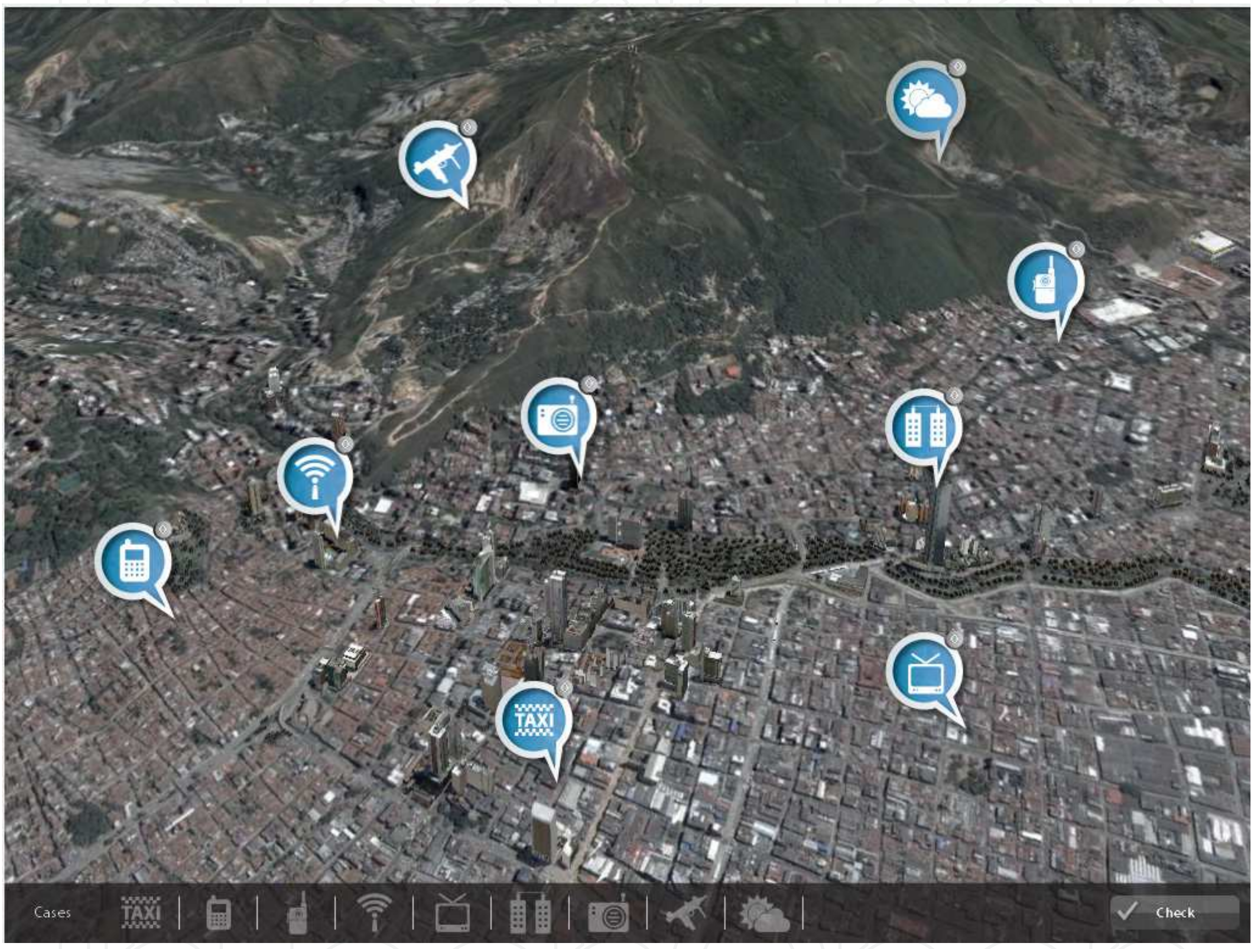


Fig. 3. An interface developed for TEST

6. Considerations for 3D modeling of real scenarios in game engines

The use of Game engines imposes some restrictions to the scenarios and the “3D world” typically used in a training system. Besides the current generation of graphics cards have a high processing capacity and video memory, there exist important restrictions that we have to consider when model different scenarios.
In order to illustrate this procedures and considerations, we will use the example of a Communications training game that uses real information from terrain maps and 3D models of real elements like towers and radio antennas.
In the next figure, we show a view of a real city, in the Game Engine. This view is based on an ASCII Raster map with a 1m resolution, which is a typical resolution used in Digital

Terrain Maps for Radio Communications (Mobile) Planning. An initial translation from the raster to 3D format, can have more than one million polygons, which is unusable in a Game Engine. Then, it is necessary to apply some techniques that reduce the number of polygons to around 200.000, that is an acceptable number, considering that the 3D scenario will contain other elements like towers, antennas, etc.

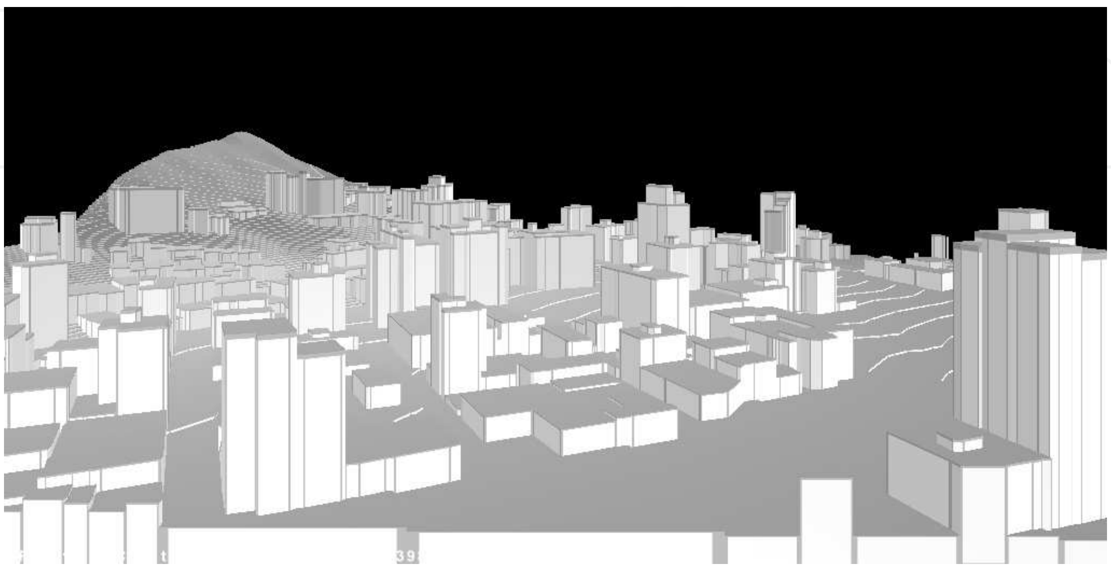
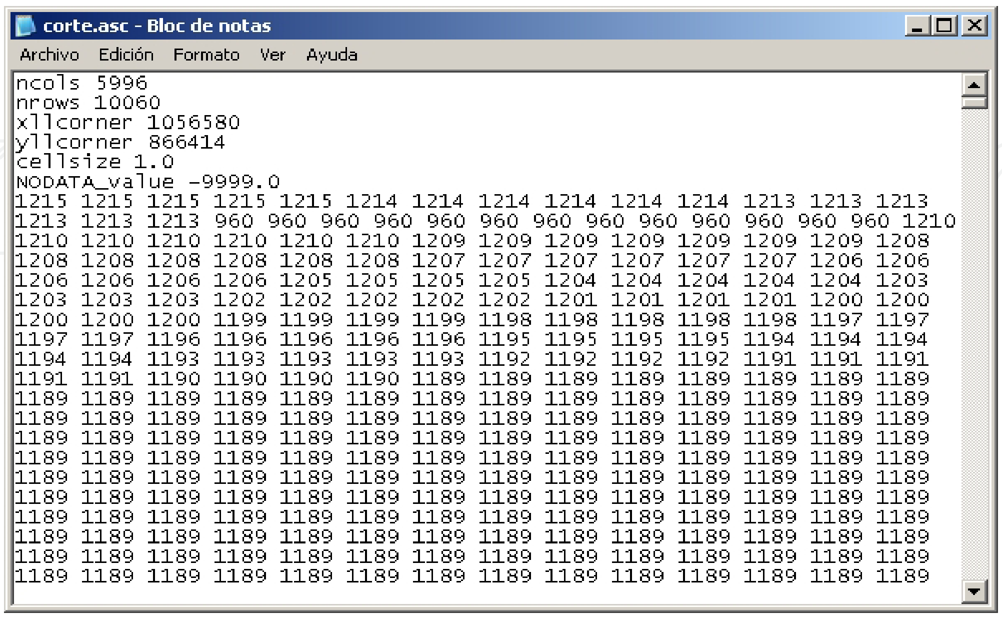


Fig. 4. A 3D map obtained from digital terrain file

There exist many applications for 3D digital city generation. Typically this tools are of the procedural type, meaning that 3D models are created from an automated algorithm instead of been modeled by a designer. This is quite useful where there are not specific requirements for the model, which is common in entertainment games.
One of the most popular open source tools for 3D modeling is Blender. Blender is an open source 3D modeling tool with plug-ins and scripts programmed in Python. Is very useful for 3D modeling and animation and is licensed using the GNU GPL model. For city generation, blender have some scripts like Suicidator City Generator, Suicidator City Engine (SCE) and Blended Cities, which is an evolution of SCE. Suicidator City Engine can use polygons information from an image file (JPEG) in order to generate streets, sidewalks or buildings. Buildings can be modified using the options available in the script. As mentioned before, this tool can be useful for entertainment games, bit for Serious Games this kind of unreal scenarios is not so useful and it is necessary (or desirable) to use real scenarios. In order to use such real scenarios, the most common source is the Digital Elevation Models used in Geographic Information Systems (GIS). This DEM information is represented in formats developed for the GIS requirements and for Cartography requirements, not for 3D game engines. In the next session we explain a procedure for the conversion of a map to a 3D model useful for its implementation on a Game Engine.

6.1 The Source map

It is quite common to have cartographic information in text files in ASCII format, known as ASCII Raster, because well know GIS applications, like ArcInfo, Idrisi or Ilwis use this format. This data included in text file have information about coordinates and height and must be displayed in graphic manner. In Figure 4 we show an example of an ASCII raster file and in Figure 5 the same map is showed using a GIS tool like Google Earth.



```
ncols 5996
nrows 10060
xllcorner 1056580
yllcorner 866414
cellsize 1.0
NODATA_value -9999.0
1215 1215 1215 1215 1215 1214 1214 1214 1214 1214 1214 1213 1213 1213
1213 1213 1213 960 960 960 960 960 960 960 960 960 960 960 960 960 1210
1210 1210 1210 1210 1210 1210 1209 1209 1209 1209 1209 1209 1209 1208
1208 1208 1208 1208 1208 1208 1207 1207 1207 1207 1207 1207 1206 1206
1206 1206 1206 1206 1205 1205 1205 1205 1204 1204 1204 1204 1204 1203
1203 1203 1203 1202 1202 1202 1202 1202 1201 1201 1201 1201 1200 1200
1200 1200 1200 1199 1199 1199 1199 1198 1198 1198 1198 1198 1197 1197
1197 1197 1196 1196 1196 1196 1196 1195 1195 1195 1195 1194 1194 1194
1194 1194 1193 1193 1193 1193 1193 1192 1192 1192 1192 1191 1191 1191
1191 1191 1190 1190 1190 1190 1189 1189 1189 1189 1189 1189 1189 1189
1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189
1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189
1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189
1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189
1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189
1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189
1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189
1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189
1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189
1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189 1189
```

Fig. 5. ASCII Raster file example

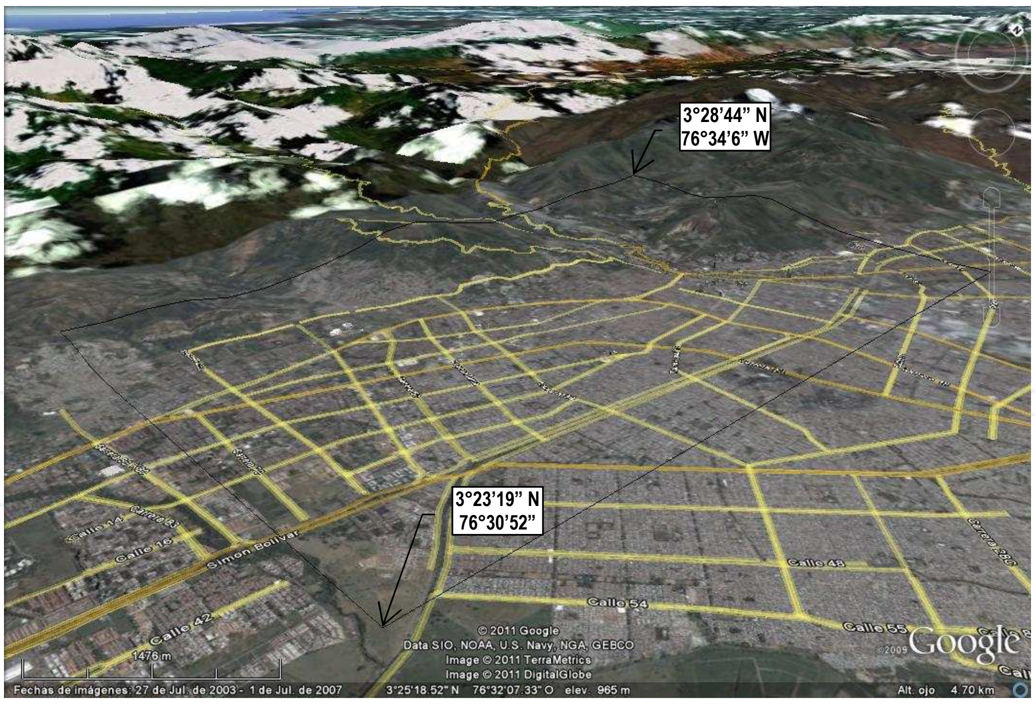

3°28'44" N
76°34'6" W
3°23'19" N
76°30'52"
Calle 16
Simon Bolivar
Calle 42
Calle 48
Calle 54
Calle 55
1476 m
© 2011 Google
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Image © 2011 TerraMetrics
Image © 2011 DigitalGlobe
Fechas de imágenes: 27 de Jul. de 2003 - 1 de Jul. de 2007
3°25'18.52" N 76°32'07.33" O elev. 965 m
Alt. ojo 4.70 km
Google

Fig. 6. Same map viewed in Google Earth

There are many open source or free tools for GIS data visualization, as well as different commercial tools for the same purpose. Some Open tools for GIS visualization are MapWindow, GRASS or SAGA. A quite well free tool is OpenEV, which can be sued for 3D visualization and data conversion.

If the map file is available in shape 3D, this kind of file can be exported to Blender almost directly.

6.2 Conversion process

The first step to convert a raster file in to a 3D model is to convert the ASCII file (.asc extension) into a shape file (.shp extension), which is vector file format, more close to most of the 3D format files. For such conversion, tools like MapWindow are adequate, but results do not allow a direct conversion from the shape to 3D tool like Blender. The development of scripts for tools like SAGA or GRASS allows better results.
Other possibility is to use a JPEG file obtained from a GIS visualization tool like OpenEV and use an open source tool like Blender to generate 3D from the gray scale obtained initially. In the Figure 6 is shown a section of a real map in OpenEV.

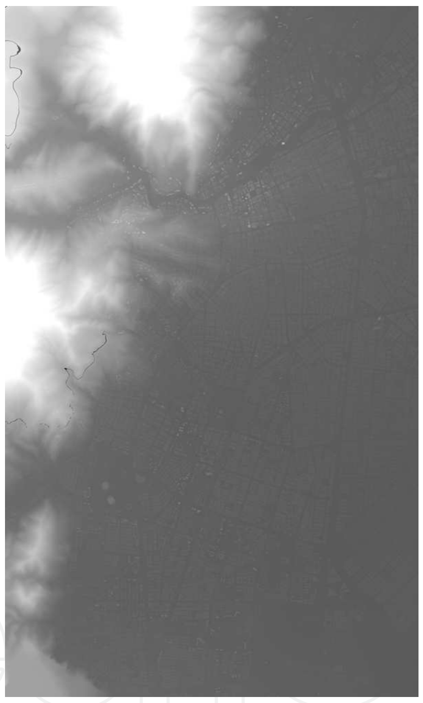


Fig. 7. JPEG image obtained from OpenEV

From this image, it is possible to import the image to Blender, and obtain a 3D model like the one shown in Figure 7. This model can be used in a game engine, just exporting it to an adequate format. However, this model could be improved substantially using additional tools or simple tricks like separating the image of the Figure 3 in two different images that allows separating the terrain from the buildings.

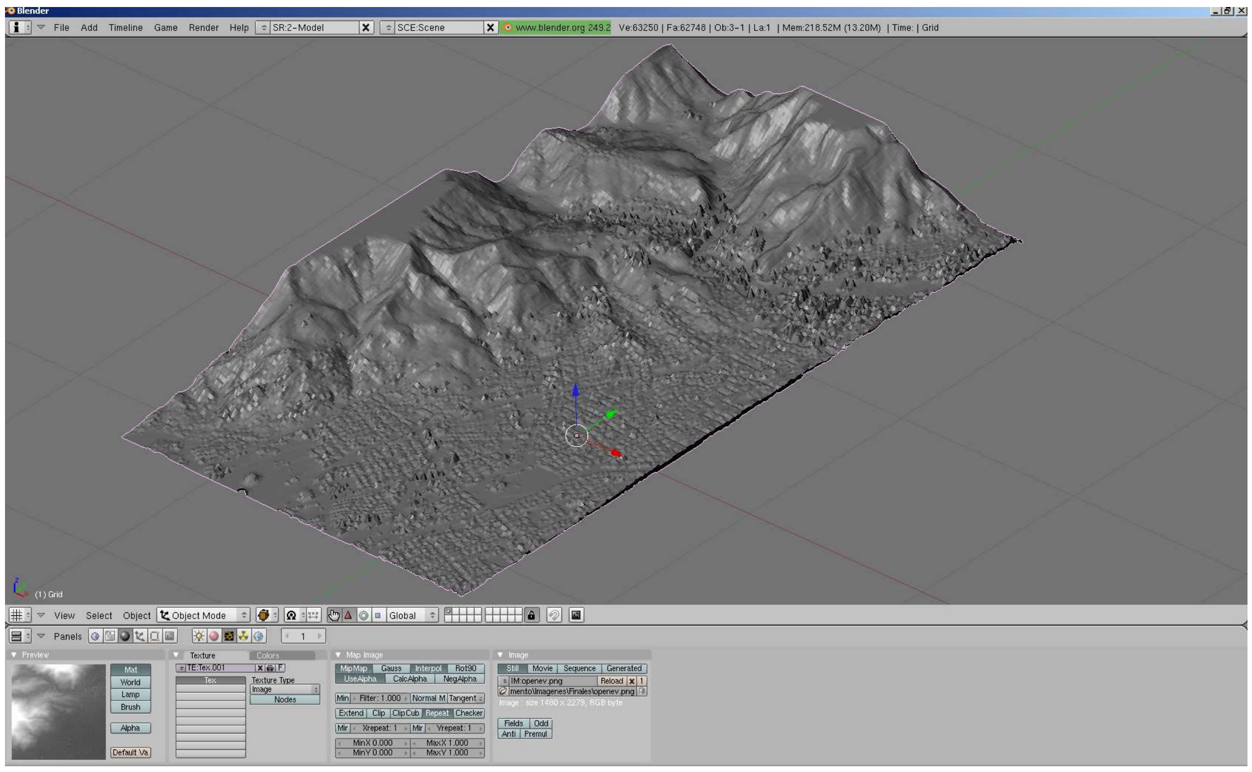


Fig. 8. 3D model obtained in Blender

In Figure 9, two images, one for terrain and the other for buildings are shown. Both images were obtained from the same image of Figure 6, but processed using an image processing tool like Gimp to modify the gray scale and separating buildings from terrain.

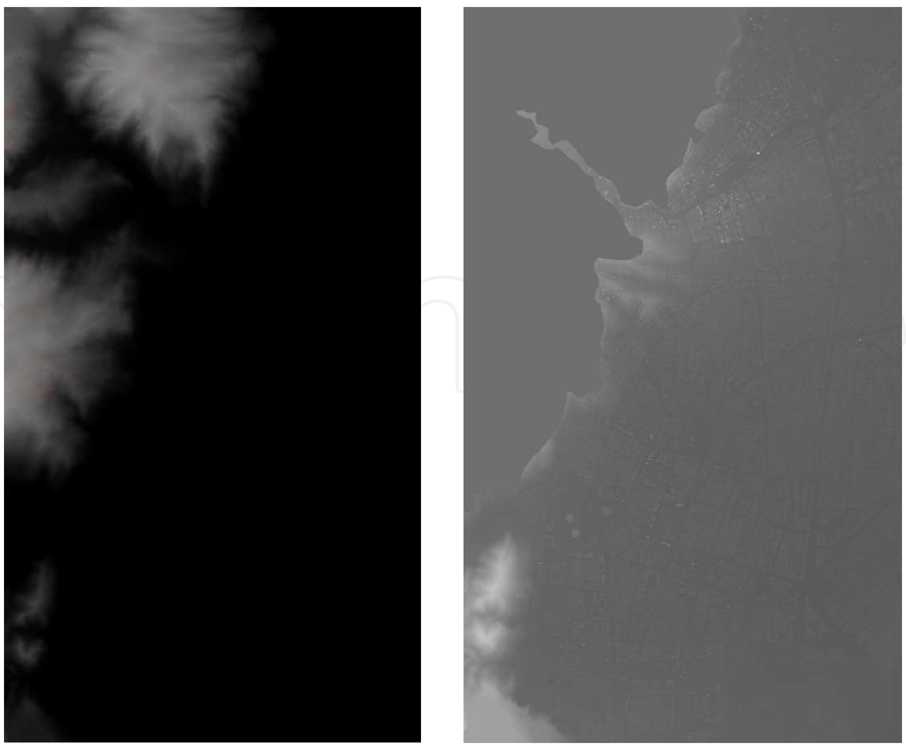


Fig. 9. (a) Image from terrain (b) image for buildings

Additionally and in order to improve the visualization on the game, the 3D map obtained in Blender can be enriched with 3D models obtained from tools like Google SketchUp, and combined with different light sources and renders, finally obtaining a 3D model like the one shown in Figure 10.

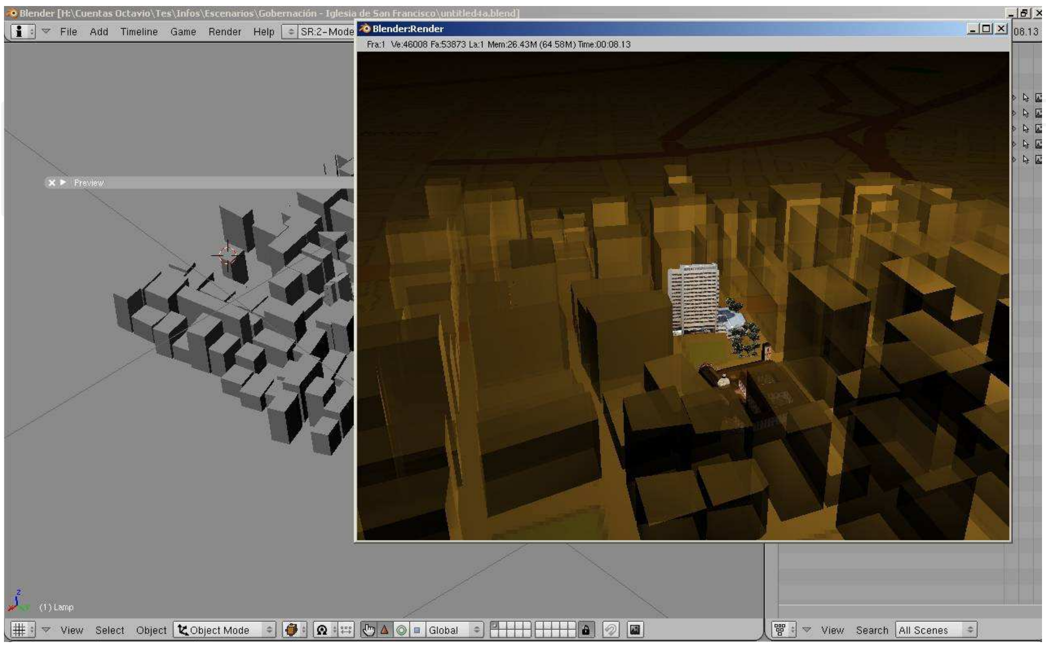

Blender [H:\Cuentas Octavio\Tes\Infos\Escenarios\Gobernación - Iglesia de San Francisco\untitled4a.blend]
File Add Timeline Game Render Help
Blender:Render
Fra:1 Ve:46008 Fa:53873 La:1 Mem:26.43M (64.58M) Time:00:08.13
Preview
(1) Lamp
View Select Object Object Mode Global
View Search All Scenes

Fig. 10. Buildings 3D model in Blender

6.3 3D modeling tools and its interaction with game engines

In the market exists many 3D modeling tools, like 3D Studio in the commercial world, or Blender in the open source world. However, these tools do not consider the specific needs of a 3D model that will be used in a Game Engine, neither in default formats nor in number of polygons or visualization characteristics. In this section, we will explain how to use such tools in order to model 3D objects for a 3D game with Open Source Game Engines.

In previous section, we explain how to obtain a 3D model from a real map and how to manipulate it in a 3D tool like Blender. The next step is to "optimize" the 3D model in order to load it in the Game Engine and allow a "fluid" navigation in the game. Basically, this process depends on the limitations of the Engine respect to the number of polygons accepted and hardware capacity (RAM, Graphic Card, etc).

The first consideration is the 3D format. There are many 3D formats, but two commonly used are 3ds (originally from 3D Studio Max) and OBJ (is a recent format commonly used in the open source community). For Engines like JMonkey, the better format is OBJ, because it allows preserving the uv coordinates, texture loading and interoperability with alpha channels used in 3D modeling tool. The 3DS format has some problems with alpha channels and iv coordinates in some Engines. Another advantage for OBJ is the preservation of object size. An important consideration when the file is exported from Blender is to export the different 3D layers in different OBJ files, according to the game requirements.

Polygon reduction is perhaps a time consuming task that requires some simplification of the 3D model in Blender. Typically a 3D model obtained from a map like mentioned earlier consist of around 1 to 2 million of polygons and must be reduced to a maximum of 700K polygons, but preferably 200K polygons, like the city shown in Figure 10.

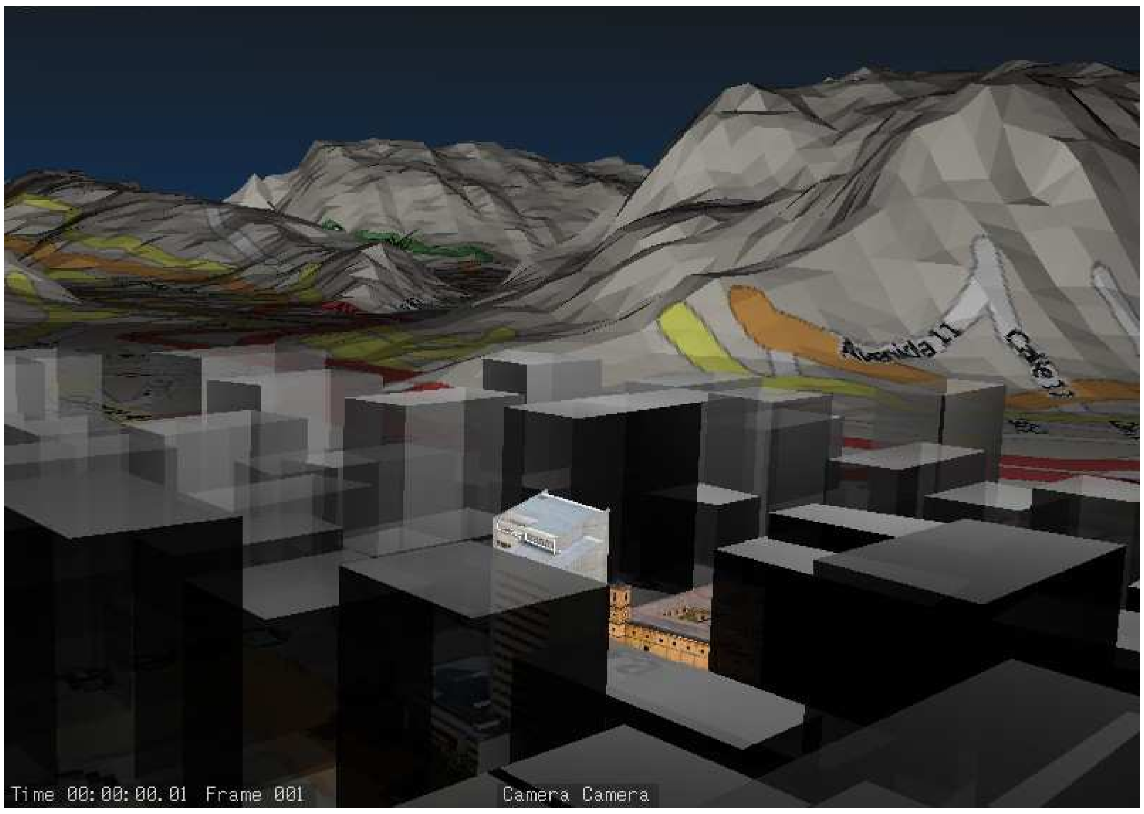


Fig. 11. Final city for Use in Game Engine

7. Conclusions

In this chapter, we have discussed our experience with Digital Terrain Maps conversion to a 3D engine compatible format, using Open Source tools and some considerations about 3D models for Serious Games. We have used Open Source Game Engines with similar performance to a commercial tool.

Special considerations have to be with the polygons number in the 3D map, because the typical conversion process from a DTM to a 3D model generates a huge number of polygons, that makes the Engine unusable.

8. References

Bruner, Jerome (1989). Acción, pensamiento y lenguaje, Madrid: Alianza Editorial.

Caillois, Roger (1986). Les Jeux et les hummens. Los juegos y los hombres : la máscara y el vértigo, México: Ed. Fondo de Cultura Económica.

Huizinga, Johan (1938). Homo Ludens. Madrid: Alianza editorial.

Navarro, Andres; Madriñan, Patricia and Pradilla, Juan Vicente (2010). A 3D Game Tool for Mobile Networks Planning 2010 Second International Conference on Mobile, Hybrid, and On-Line Learning ,Saint Maarten, Netherlands, Antilles February 10-February 16.

Navarro, Andres; Madriñan, Patricia; Londoño, Sebastian and Pradilla, Juan Vicente. (2011). Serious Games: Between Training and Entertainment, *Third International Conference on Mobile, Hybrid, and On-Line Learning*, ISBN: 9781612080031, Gosier, Guadeloupe, France February 2011.

OpenEV, Your Geospatial toolkit. Accessed: June 2011. Available from: http://openev.sourceforge.net/

Map Windows Open Source Project. Accessed: May 2011. Available from: http://www.mapwindow.org/

ESRI, ArcView. Accessed: May 2011. Available from: http://www.esri.com/software/arcgis/arcview/

Modelos Vector Raster (In Spanish). Accessed: January 2011. Available from: http://gemini.udistrital.edu.co/comunidad/profesores/rfranco/vector_raster.htm

Open Street Map. Accessed: January 2011. Available from: http://www.openstreetmap.org/

Blender. Accessed: January 2011. Available from: http://www.blender.org/

Procedural Generation. Accessed: January 2011. Available from: http://en.wikipedia.org/wiki/Procedural_generation