Open system for synthetic terrain visualization

Horacio Abbate[1,2] and Leandro Linardos[2]

[1] Instituto de Investigaciones Científicas y Técnicas para la Defensa CITEDEF (MINDEF)
habbate@citedef.gob.ar
[2] Universidad de Buenos Aires, Facultad de Ingeniería, Buenos Aires, Argentina
{habbate, llinardos}@fi.uba.ar

Abstract. This paper is concerned about the production of open source tools for development of training simulators in particular and any kind of applications in general that need real-time visualization of large geographical environments. The purpose is to use these tools in projects of the Ministry of Defense at first, to finally open them to the rest of the Government and the community. The proposed solution is based on OpenSceneGraph (OSG), which is an open source visualization engine, and Common Database (CDB), which is an open standard for systems oriented geographic databases with visualization of large geographic environments. CDB is a synthetic environment database specification and OSG is an open source high performance 3D graphic toolkit widely used by the simulation, spatial, scientific visualization, games and virtual reality industries and it supports many of the same formats used by CDB. This paper presents the design and implementation of a CDB visualization technique that uses OSG. The solution composes OSG compatible SG's from CDB nodal data so that datasets can be shared, reused and made extensible. The elevation imagery example described in this paper includes an evaluation of loading times and frame rates. The results indicate this is a real-time solution that can be applied across large synthetic environments to visualize data stored in CDB databases using OSG as the image generator.

Keywords: open source, geographical database, synthetic environment, OSG, CDB.

1 Introduction

Computer applications that require data management and visualization of geographic scenarios, must solve both the specific functionalities and those of the representation of the scenarios.

This characteristic implies high additional costs in specialized human resources and economic resources corresponding to the necessary hardware tools and software licenses. Likewise, they will require longer development times.

This work is part of the effort being carried out by Instituto de Investigaciones Científicos y Técnicos para la Defensa (CITEDEF) to produce open and economical tools for the development and production of training simulators that produce real-time visual representations of wide geographic scenarios.

A geographic information system (GIS) is a computer system for storing and displaying data related to positions on the Earth's surface. GIS's are designed to manipulate, manage, and present many types of spatial or geographical data enabling people to more easily see, analyze, and understand patterns and relationships. A Synthetic Environment (SE) is used by the GIS to represent physical world activities – for example, theaters of war where military systems can interact with a variety of simulated entities such as autonomous vehicles, weapons, terrain, buildings, vegetation, climate, and weather conditions. Due to arrange of temporal and space dependencies, SE's use relational databases designed specifically for GIS's known as Synthetic Environment Databases (SEDB). Visualization of the SEDB is achieved using an Image Generator (IG). An IG is software that generates images or frames to be visualized on screen. This is done by traversing SE data structures called Scene Graphs (SG).

The Open Geospatial Consortium (OGC) [3] is an international organization committed to making open standards for the geospatial community, and it consists of hundreds of commercial, government, and research organizations worldwide that collaborate in the development of open standards for geospatial content and services, GIS data processing and data sharing. The OGC has established the Common Database (CDB) specification as a best practice for the storage, access and update of SEDB's.

The CDB Specification is an open synthetic environment database specification that defines the data representation, organization and storage structure of worldwide synthetic representations of the earth as well as the conventions necessary to support full-mission simulators. CDB defines the database structure and content using open formats e.g. elevation data - GeoTIFF format, satellite imagery - JP2K format, 3D models - OpenFlight format, and vector data - ESRI Shapefile format. CDB uses established simulation industry formats and is specifically tailored for real-time applications where inter-connected simulators can share a common view of the simulated environment. CDB was developed by CAE [2], and is now managed by Presagis Inc. [4].

OpenSceneGraph (OSG) is an open source high performance 3D graphic toolkit [5] used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modeling. It runs on a variety of operating systems including Microsoft Windows, Mac OS X, Linux, IRIX, Solaris, FreeBSD and mobile platforms, namely iOS and Android. OpenSceneGraph uses a SG approach with libraries for construction and manipulation of the scene graph, shadowing, physical representations, and user interactions. It contains classes representing various types of nodes, scene geometry, state abstraction, geometric transformations, and it uses OpenGL which is a cross-language, cross-platform application programming interface (API) for hardware-accelerated rendering 2D and 3D vector graphics. Visualization is achieved using a deferred rendering technique where the graphics processing unit (GPU) breaks geometries down into vertices, and then transforms and splits them into fragments before the final rendering with shading. The osgTerrain library is particularly of interest for this work because it provides terrain rendering support to OSG. An introduction to its use was done by Wang and Qian [5]. This work was made possible because of OSG's maturity and its format support versatility for real-time visualization of CDB synthetic environment (SE) terrains. This presented solution makes it

possible to dynamically visualize CDB synthetic environment terrains, particularly its elevation and satellite imagery datasets at varying levels of detail.

This work was supported by projects: “Universidad y Transporte” 32-64-007 Secretaría de Políticas Universitarias (SPU) Ministerio de Educación, y por PIDDEF 03/2014 “Simuladores Móviles” Ministerio de Defensa.

1.1 Other Works

CDB content can be displayed utilizing osgEarth [8] (a geospatial SDK for OSG applications to visualize terrain models) by using an open source driver to read CDB data made available by GAJ Geospatial Enterprises LLC [8].

2 CDB and OSG

CDB specifies the way to organize the data to achieve a synthetic representation. To guarantee the efficient localization and access to the database, its data is organized in Tiles, Layers and Level of Details (LODs).

CDB divides the Earth surface into cells of 1° of latitude and variable longitude in function of latitude named Geocells. A geocell logically organizes the SE features data into inter-independent Layers. A layer is a set of datasets grouping related SE features and it is identified by a number and a name. A layer is organized into components and each component groups the datasets for a single SE feature (e.g.: the daily and nightly imagery datasets are components of the imagery layer). A layer component is identified by a pair of component selectors. Finally, each layer component is divided into LOD-tiles, following a quad-tree structure. A LOD-tile is the subset of datasets of a layer component which have the same level of resolution. The LOD-tile data is organized for each geocell and layer component into rows and columns resulting in a grid, identifying a LOD-tile with an up index reference (UREF) and a right index reference (RREF). The UREF is the row position in the grid, being zero the bottom row corresponding to south. The RREF is the column position in the grid, being zero the leftmost column corresponding to west. This of level of detail organization is applied to a big subset of CDB layers named tiled datasets that vary with space, i.e.: terrain elevation, satellite imagery and vector data.

OSG represents terrains by objects of classes of the osgTerrain library, i.e.: Terrain, TerrainTile and Layer. A Terrain node groups a set of TerrainTile nodes and builds the terrain geometry using the elevation layer (instance osgTerrain::HeightFieldLayer) associated to each TerrainTile instance. The TerrainTile geometry is textured using one or many image layers (instances of osgTerrain::ImageLayer) associated to each TerrainTile instance. Each terrain tile is located on the 3D world by an object of class osgTerrain::Locator. A locator converts geographic coordinate (i.e.: latitude, longitude and altitude) to 3D scene coordinates (i.e.: x, y and z).

Terrain representation and rendering techniques belong to a field of interest in computer graphics. OSG includes functionality related to this field. The OSG LOD

node is a group node which allows switching among different children nodes; each child node is a different resolution representation for the same model. OSG also support database paging with osg::PagedLOD nodes and the osgDB::DatabasePager entity, with a similar LOD node behavior but for the former does not need all children nodes loaded at all times. It allows referencing a child (usually with a file name) to be loaded and added to the scene graph when it would be necessary. Regarding the level of detail switching mechanisms, OSG implements two methods: distance to the observer and pixel screen size. In both cases, each LOD child has an active range and the active child node is the one in which the active range is valid. A descriptive analysis of this kind of nodes and mechanisms can be found in AlphaPixel [12].

The application can control the image generator OSG directly through an API (Application Programming Interface), or by an UDP (User Datagram Protocol) link by mean of the open-source library CIGI (Common Image Generator Interface)[1][2], as shown in Fig 1. The solution proposed in this work is based on the API mechanism (Fig. 1 right).

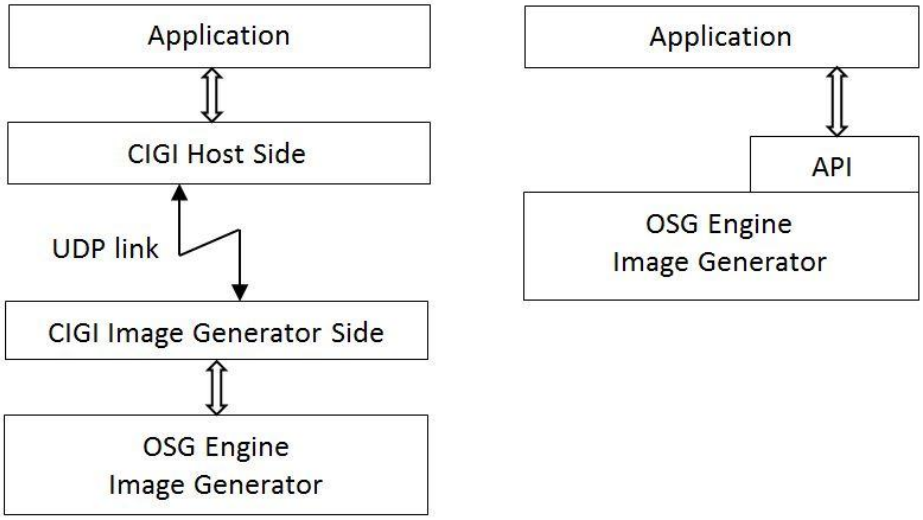

Fig. 1. control by UDP link (left), control by API (right)

3 Solution

3.1 Locating and Accessing CDB

A CDB database can be seen as a set of terrain areas and features associated to each terrain area. A CDB dataset is the description of a feature for a terrain area. We can identify a terrain area through a geocell, a level of detail and a pair of UREF and RREF indexes. We group this data in a tuple named TileID. Moreover, we can identify a feature through a CDB layer and one of its components.

CDB database is a set of folders and files in the OS file system. Each CDB dataset is usually one or more files in the CDB database. To locate a CDB dataset we build the path to the file representing this dataset based on the TileID and the layer and layer component identifiers. In the case of tiled datasets (such as elevation and satellite imagery), the full path depends on the geocell latitude and longitude, the dataset layer and its components, the dataset level of detail and the dataset reference indexes (UREF and RREF) [11].

Once located the file, we need to access it, represent it with OSG entities and finally render it. We will access two CDB layers components: primary elevation and yearly satellite imagery (there are also monthly and quarterly satellite imagery).
OSG read the elevation datasets, which are GeoTIFF files, by using the GDAL library plugin [10]. This plugin reads the .tiff file as an instance of osg::HeightField, to represents a height field used to build an instance of osgTerrain::ElevationLayer associated to an instance of osg::TerrainTile as its elevation layer.

To locate this osgTerrain::TerrainTile on screen space we associate an instance of osgTerrain::Locator to it. We can set the locator parameters by using the TileID data or the GeoTIFF geospatial data. This osgTerrain::TerrainTile is ready to be contained in an instance of osgTerrain::Terrain and to be rendered.

OSG reads the imagery datasets, which are JPEG 2000 files, by using JasPer library plugin [13]. This plugin reads the .jp2 file and returns an instance of osg::Image to represents an image to build an instance of osgTerrain::ImageLayer associated to an instance of osg::TerrainTile as a color layer.

The same terrain region only with elevation dataset and with elevation and imagery dataset is shown in Fig 2.

The osgDB plugins mechanism uses the file extension to determine the plugin to be used. In order to differentiate GeoTIFF files with TIFF files is necessary to add ".gdal" extension to the ".tiff" GeoTIFF file. OSG uses the GDAL plugin when a file with extension ".tiff.gdal" is going to be loaded and then, the GDAL plugin removes the ".gdal" extension to access the existing ".tiff" file.

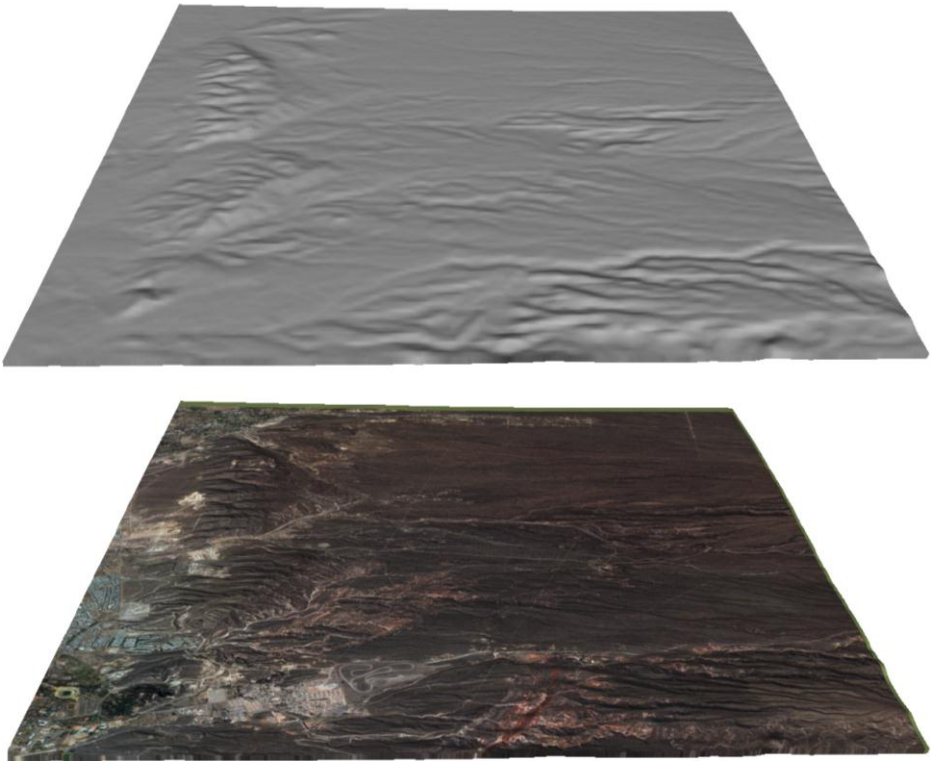

Fig. 2. Rendered CDB terrain tile: elevation dataset (up), elevation and imagery datasets (down).

3.2 Composing the Terrain Scene Graph

Taking advantages of deferred and asynchronous OSG load mechanisms, the node representing a terrain section will be composed by an OSG terrain tile at a certain level of detail and the necessary data to build a new node representing the same terrain section at a more detailed level of detail.

We want to visualize a terrain section at different resolutions. To do this, we first represent the terrain section at an arbitrary level of detail L. We access to the corresponding elevation and imagery datasets in the CDB database using a CDB TileID (representing the terrain section and level of detail), build an OSG terrain tile and locate it in the scene. Second, we create a new sub-graph that contains the representation at a level of detail L (the previously created OSG terrain tile) and supports the representation of the same terrain section at the next level of detail, L+1. We achieve this by creating a PagedLOD node with two children nodes. One of the children nodes is the previously created OSG terrain tile and the other is the data to build the next level of detail sub-graph. This data is the CDB TileIDs of each one of the at most four next more detailed level of detail CDB tiles. In the case that there is no more detailed

datasets, the OSG terrain tile with a level of detail L will be the only node in the terrain area sub-graph. The sub-graphs structure is shown in Fig 3.

In the case the sub-graph contains a PagedLOD node, the more detailed CDB TileIDs are processed and a new sub-graph is built at switching time. The process is to access the CDB database, generate the OSG terrain tiles, build a sub-graph for each OSG terrain tile following the same mechanism explained above (a PagedLOD with information to build the sub-graph with level of detail L+2 or a single OSG terrain tile) and finally group these sub-graphs (in a OSG Group node). This sub-graph will be merged into the terrain scene graph by OSG as a PagedLOD child, replacing the data used to generate it. The group node structure is shown in Fig 4.

This process is the same for each child of the PagedLOD at switching time (note that switching process is not necessarily triggered at the same time).
Fig 5 shows the same area of terrain represented with one tile of a level of detail L and with four tiles of the more detailed level of detail L+1.

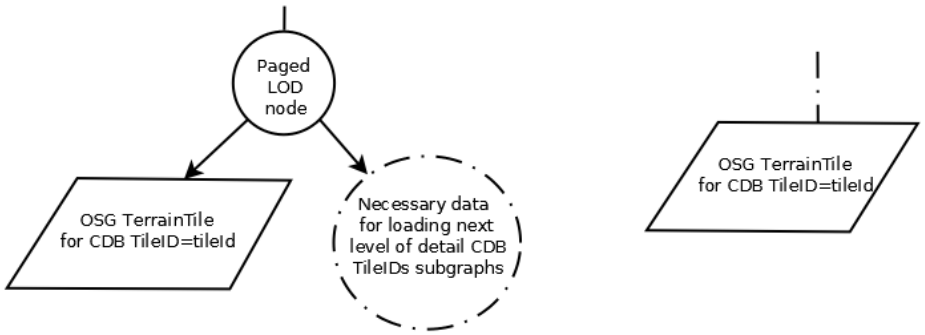

Fig. 3. Sub-graphs if there are (left) and there are not (right) more detailed CDB TileIDs.

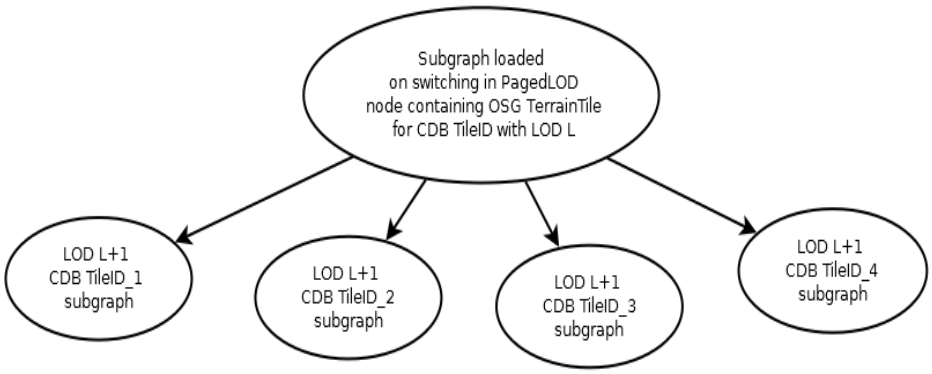

Fig. 4. Sub-graph loaded when a PagedLOD level of detail switch is triggered.

4 Solution Design

The solution was designed and implemented in three software components or layers: CDB layer, Scene sub-graph builder layer and Scene builder layer.

The CDB layer manages access to the CDB database. It represents CDB model, resources, queries and responses. It solves queries accessing the CDB database and gives responses for them. Fig 5 shows this layers design.

The Scene sub-graphs builder layer creates CDB queries requesting the necessary resources (images, height-fields and model nodes among others) to build the scene sub-graphs. For example, we implemented a Dynamic tile builder which creates elevation and imagery queries, send them to the CDB Layer and uses the height-fields and images responses to build the terrain sub-graphs with PagedLOD nodes, as it was described in the previous section. This layer gives extensibility to support others CDB datasets: for example, we can build a new builder that creates queries for 3D models and model location data, build the 3D model nodes and translate, scale and orient them in an existing terrain section.

The Scene layer handles scene parameters (components, geo-cells and CDB TileIDs to load, LOD switch policies, fog) and composes the scene from different subgraph builders: the tile builders for terrain tiles, the model builders for 3D models and other builders for sky dome nodes for example. The output is a scene graph ready to be rendered.

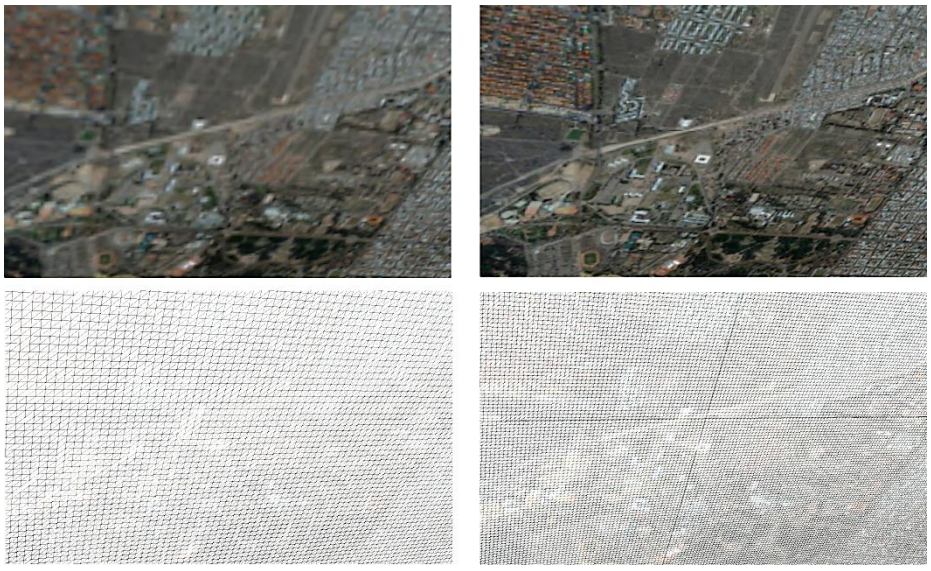

Fig. 5. The same terrain area represented with one tile with LOD 2 on the left and with four tiles with LOD 3 on the right. The terrain is rendered normally on the top and is rendered as wireframes on the bottom. Note that triangles at LOD 3 are smaller than triangles at LOD 2.

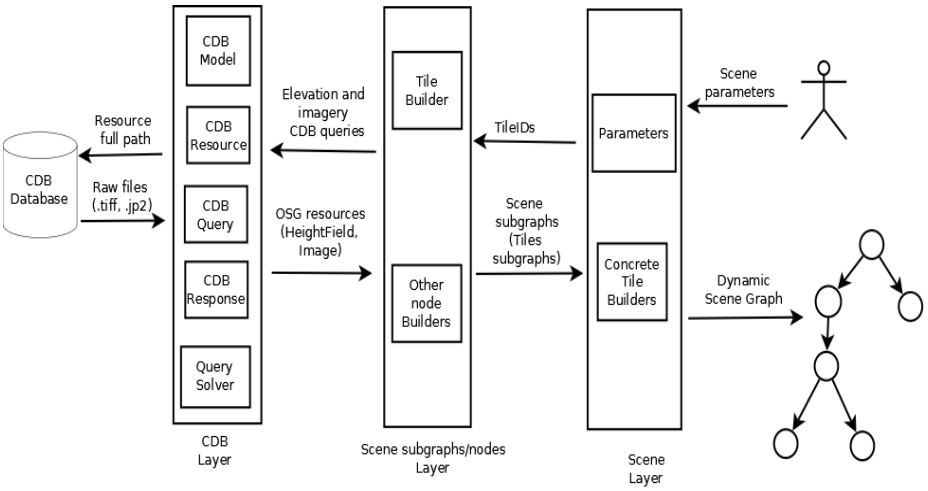

Fig. 6. Solution design.

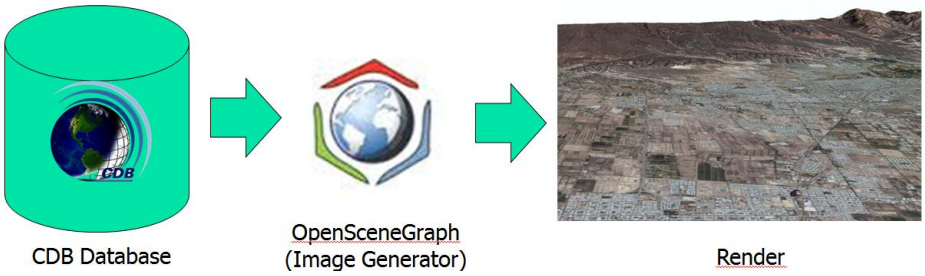

Fig. 7. Data flow from the geographic description in the database to the pixels rendered.

The application on OSG (OpenSceneGraph) localize, access and represent CDB (Common Database) content in order to render the elevation and imagery of the Synthetic Environment stored on a CDB database imagery supporting LOD (level of detail) at real-time and render the visualization of the landscape (Fig 7).

5 Results

We use a CDB database version 3.0 with elevation and imagery data for a geocell with latitude South 33 and West 69. This geocell corresponds to the area of the El Plumerillo International Airport located in Mendoza, Argentina. The maximum level of detail of the elevation dataset is 3 and it has bounds in the region with corners (U0, R0), (U0, R2), (U1, R0) and (U1, R2). A terrain scene rendering snapshot of an area of the Mendoza province is shown in Fig 8.

We run four times the implemented solution, navigating over the scene and switching level of detail by distance to the observer. We measured three metrics groups: Frame rates:

- The frames per second collected each second during the execution. It is shown in Table 1.
- CDB access time: The time from the beginning of the query execution until the image or height-field is ready to be used. It is shown in Table 2.
- Scene graph update times: The time from the LOD node switch until the scene merge. It includes the CDB access time, the node geometry building and the merge. It is shown in Table 3.

The measures indicate negligible CDB access times, where the scene graph merge is the more time consuming part of the process. This last process time increases during execution because as we have more tiles to render in the scene, we have more drawing time per frame and then less time among frames to generate and merge the new subgraph (this operation is done incrementally by OSG). Regarding frame rates, it has little decrements during the merge time where the 1024x1024 tile size has performance impact on not deployment hardware, but stays close to 60 fps as average and as median. Note that the 1024x1024 tile size is bigger than what is typically used in the industry, as Price and Mingee [15] mentioned.

The measures and snapshots were obtained using osgViewer, the OSG image generator, at a resolution of 1900x1200 running in a PC with an Intel core i7 4790 processor at 4.0 GHz, 16 GB main memory, a NVIDIA Quadro K4200 graphic card with 4 GB of video memory and a mechanic drive running Windows 7 64 bits. The OSG version is 3.4.0, the GDAL library version is 1.11.2 and the Jasper library version is 1.900.1. All source code (including OSG and libraries) was compiled with MinGW G++ (GCC) 32 bits compiler version 4.8.1.

Fig. 8. Rendered terrain (Mendoza)

Table 1. Frames rates for different runs.

Run	Duration [sec]	Frames per second [fps]		
		Minimum	Average	Median
1	105,48	36,94	57,94	59,95
2	347,92	39,00	58,41	59,95
3	159,65	38,96	58,02	59,95
4	52,12	32,97	55,21	57,54

Table 2. CDB access time.

Run	Maximum		Average		Median	
	Frames count	Time [sec]	Frames count	Time [sec]	Frames count	Time [sec]
1	27	0,45	8,25	0,15	6,00	0,10
2	25	0,42	6,90	0,14	3,00	0,10
3	25	0,42	7,17	0,14	3,00	0,10
4	24	0,40	7,10	0,14	3,00	0,10

Table 3. Scene graph update metrics

Run	Maximum		Average		Median	
	Frames count	Time [sec]	Frames count	Time [sec]	Frames count	Time [sec]
1	1328	22,28	248,45	4,18	238,00	4,09
2	471	7,98	151,70	2,61	311,00	5,55
3	1608	27,24	256,79	4,39	934,00	15,72
4	614	10,39	171,95	2,95	302,00	5,34

6 Conclusions and future work

The open-source solution developed upon OSG allows the real time visualization of CDB synthetic environment, only needing a way to localize, access, represent SE data in a scene graph using OSG provided entities and grouping them into convenient sub-graphs. We obtained a fluid visualization with improvable hardware and software.

The results shown guarantee the OSG+CDB usage in interactive real time simulations, such as flight simulators, with more than acceptable performance, providing a powerful open source option.

This work provides a basis to further development into the following three areas:

- CIGI: alternate solution to control the image generator OSG by CIGI protocol.
- Content: CDB datasets for cultural and natural features can be represented by OSG using OSG formats.
- Performance: performance can be improved by the parallelization of operations, such as: the tile building process at switching time, the database management, que-

rying and image generation can be done using separate nodes, avoiding having to share hardware resources.
- Image quality: the visualization error could be implemented in order to achieve an even more realistic visualization.

References

1. L. Durham y P. Bill, «Interface Control Document for the Common Image Generator Interface (CIGI),» 2008. [En línea]. Available: http://cigi.sourceforge.net/specification.php.
2. «CIGI Class Library,» [En línea]. Available: http://cigi.sourceforge.net/product_ccl.php.
3. Open Geospatial Consortium. Web site: http://www.opengeospatial.org/
4. Canadian Aviation Electronics Inc. Web site: http://www.cae.com/
5. Presagis Inc. Web site: http://www.presagis.com
6. OpenSceneGraph Project. Web site: http://www.openscenegraph.org/
7. Wang, R., Qian, X.: OpenSceneGraph 3.0, Beginner's Guide (2010). ISBN 13: 9781849512824
8. osgEarth. Web site: http://osgearth.org/
9. Common Database Drivers for osgEarth. Web site: https://github.com/gajgeospatial/osgearth_cdb
10. AlphaPixel, LLC: LOD (Level of detail) in OpenSceneGraph (2013). Retrieved July 31, 2016 from http://alphapixel.com/wp-content/uploads/2015/04/LOD-Level-of-detail-in-OpenSceneGraph-OSG.pdf
11. Presagis Inc.: CDB Specification – Version 3.2 – Volume 1 (2014) in http://www.presagis.com/products_services/standards/cdb/more/download_the_cdb_specification/
12. Geospatial Data Abstraction Library. Web site: http://www.gdal.org/
13. JasPer Project. Sitio web: http://www.ece.uvic.ca/~frodo/jasper/
14. gdal_translate command. Web site: http://www.gdal.org/gdal_translate.html
15. Price, D., Mingee, R., Presagis Inc.: Lessons Learned Developing a CDB Run Time Publisher (2009). Retrieved July 31, 2016 from http://www.presagis.com/files/whitepapers/2009-06-WP-CDB-RTP-SimTecT09.pdf