DATABASE DESIGN FACTORS FOR A SERVER-BASED, WHOLE-EARTH RENDERING ENGINE

W. A. Harvey, Jr.
TerraSim, Inc.
Pittsburgh, PA USA

S. Rao
TerraSim, Inc.
Pittsburgh, PA USA

ABSTRACT

This paper describes recent work to develop a worldwide GIS data manager that simplifies storage of very large-scale GIS data while providing efficient retrieval for rendering the data. We have based our solution on open data sources, such as OpenStreetMap and available government sources, in order to keep data up to date and to keep costs low. The database must also store GIS data that is minimally processed in order to ease the storing of new data, and to support interoperability by making the data useful to the widest range of runtimes. The internal database is versioned and supports reliable transaction system properties, i.e., a "real" database. All data layers necessary for simulation — elevation, imagery, vector features, model features, land use classification — must be representable in the database in default open source formats, e.g., GeoTIFF for raster data and well-known binary (WKB) for vector data. The format is extensible so that new data sources, such as point cloud data, weather data, and proprietary model formats, can be stored. Extensibility should also support the storage of internal, runtime-specific data so that a runtime can create intermediate forms of the data that may be more efficiently rendered. Finally, the database provides efficient direct access, or it can be streamed to remote hosts from a server. Sharing the database, or a subset of it, to disconnected clients is simplified by sharing a database file.

INTRODUCTION

Simulation-based training continues to be a fast-growing market as entry costs are lowered and the capabilities and fidelity of simulation systems increase. However, even with many commercial sources of data available, the task of obtaining high quality GIS data for relevant locations continues to be expensive and time consuming, limiting the uses of simulation. VBS Blue, a whole-earth simulation technology produced by Bohemia Interactive Simulations, addresses this problem by providing geo-specific global terrain that is augmented with geotypical procedural detail. Moderate- and high-resolution baseline data is necessary to be able to derive higher resolution procedurally enhanced details. Thus, it is necessary to efficiently serve the rendering system from a planetary-scale, manageable GIS data storage solution.

Defense modeling, simulation, and training runtimes present significant challenges when developers are collecting and preparing GIS data for use. This is because the runtime requirements vary significantly based on the mission. Constructive simulation runtimes like OneSAF require basic 2.5D representations for rendering map-like visuals, while visual runtimes like OpenFlight require large-scale terrains with basic to moderate detail supporting landmark identification. Serious game runtimes for ground combat training, like VBS, require detailed ground-level content to support individual and group entity movements while providing a convincing immersive experience to the trainees.

Several format standards have been published to support most of the requirements for defense modeling and simulation runtimes. MDB (SECore/Army), NPSI (Navy), AFCD (Air Force), and recently OGC CDB (SOCOM), are all geospatial data format standards developed for a segment of the defense simulation community. While these existing simulation format standards are useful to their subset of defense simulation users, the formats contain requirements specific to the runtime(s) for which they were originally developed, and they are represented as collections of files and folders. We believe that most efficiency-related data requirements should be on the runtime. Even so, these formats can be used as containers for GIS data specific to a particular training exercise.

Presented at the **IMAGE 2017** Conference
Dayton, Ohio – 27-28 June 2017

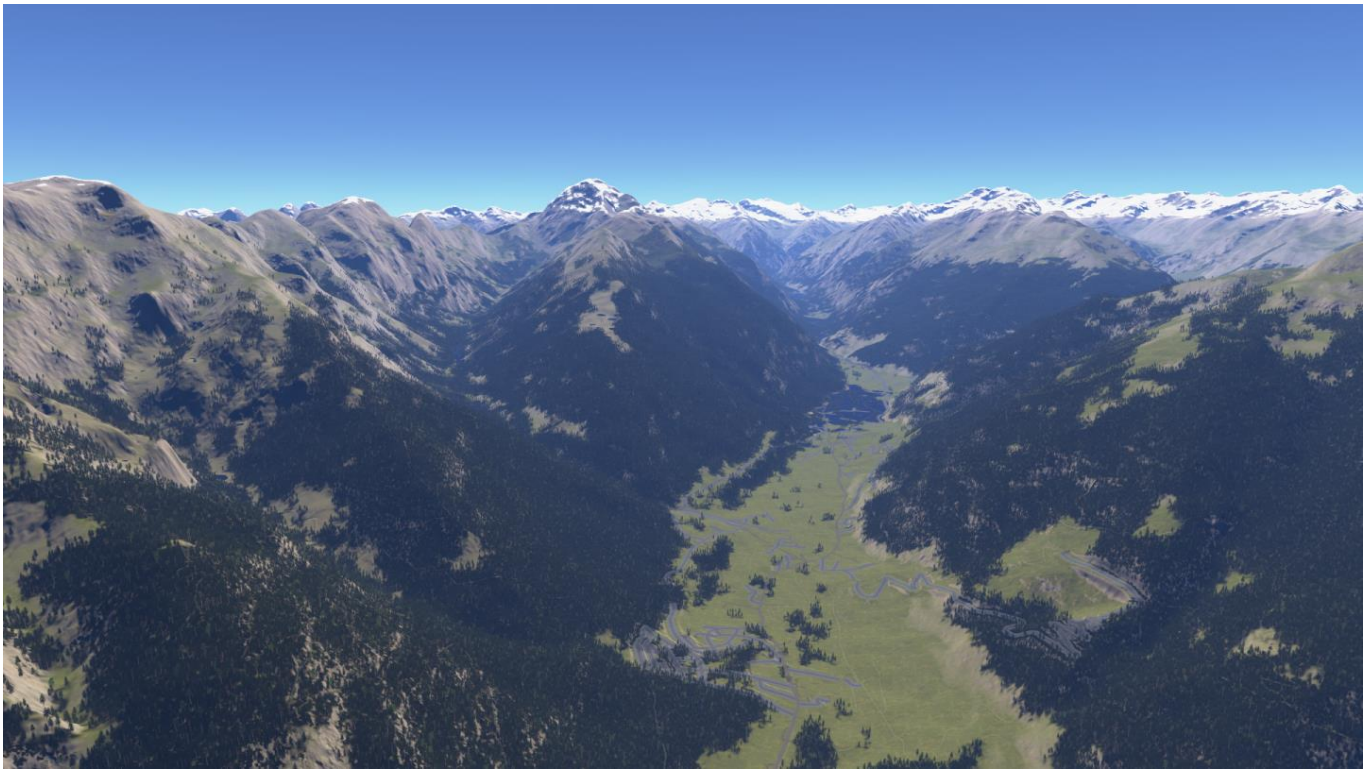

Figure 1: VBS Blue Rendering of the Alps Using Baseline Data Only

We will begin by briefly describing the VBS Blue renderer and its data architecture and requirements. We will then discuss our current work on organizing the necessary geospatial data and model asset sources into a database that can be maintained and efficiently served to VBS Blue. Based on the GeoPackage (OGC GeoPackage WG, 2017) format, this database uses modern, open GIS data formats that have existing open source API implementations, and it supports server-based (cloud) data access via the GeoServer (OGC GeoServer WG, 2017) open source GIS server. We anticipate that utilizing open formats and technologies will make it much easier to assemble and maintain geospatial data to augment gaming area terrains.

VBS BLUE: A WHOLE-EARTH RENDERING ENGINE

Traditional training exercises begin with selecting a "gaming area" — the virtual terrain location and requirements for the exercise — after which the terrain development effort begins. This includes identifying and obtaining GIS and model source data, building a process to create the terrain, then refining both the source data and the resulting virtual terrain to meet the requirements. This development process can require weeks or months of effort.

VBS Blue is a new rendering engine from Bohemia Interactive Simulations that combines procedurally-generated content with the ability to merge in geo-specific data, such as satellite imagery or building models. A global GIS database provides the baseline data from which procedural details are generated. Users can also supply their own data which is integrated and rendered with the procedural content. VBS Blue is a round earth representation of the world, allowing for larger gaming areas while still rendering very high game-quality details at the ground level. Figure 1 is a VBS Blue view of the Alps using only baseline data, i.e., without additional development to refine the scene. Figure 2 is an example of a VBS Blue rendering of an area in Poland where additional model and geospatial data content were used to improve the virtual terrain.

The global data required for the VBS Blue baseline database consists of layers of elevation and bathymetry

Figure 2: VBS Blue Rendering of Polish Landscape Using Baseline and Additional Content

data, coastline vectors, road and river centerlines, building footprints, surface material data, and some geotypical and geo-specific model assets. The baseline data size is growing rapidly as developers build out other areas of the whole-earth database from which VBS Blue derives its visuals. This growth provides additional motivation for our effort to build a database solution that can supply data to VBS Blue via a cloud- or server-based source.

The VBS Blue data architecture is intentionally designed to be open and extensible. It is built around a plug-in architecture, where each plug-in is a provider that supplies data to the renderer. The plug-in reads from a data source when a request with a location and resolution is made from the renderer. The plug-in compiles the data into an internal "binarized" format and then supplies it to the renderer. VBS Blue creates the compiled form of the data to make it more efficient to render, and it caches that form to improve performance on repeated requests. Plug-ins exist for each of the database layer types and dependencies between plug-ins are permitted.

GLOBAL BASELINE DATABASE REWORKED

In order to reorganize the baseline data into manageable pieces, we tessellate the globe into cells. This allows for database updates to occur on individual cells without invalidating the entire database. Each cell is a database that holds all the layers of data required by VBS Blue. Model data is stored separately since it can be placed/referenced globally. For the present, we set the cell size to be a single geocell (1 degree x 1 degree), and we are treating the poles (area above 80N and below 80S) as special case cells.

The cell data is stored in a GeoPackage database. It is a modern, flexible Open Geospatial Consortium standard format that stores phototexture and vector data in an SQLite database, thus supporting the basic data types (raster and vector) necessary to represent almost all of the geospatial data required by VBS Blue. It has an existing mechanism to support extensions, and there are implemented extensions, under review for inclusion in the standard, for storing elevation and point cloud data. GeoPackage has several additional useful properties:

- it stores GIS data in a minimally processed form, i.e., georeferenced, tiled, with no runtime dependencies, which simplifies storing new data and supports interoperability by making the data useful to the widest range of runtimes;
- it is implemented as a database which helps to support consistency and versioning;
- it provides single-file access so that baseline updates, or

sharing subsets of data, is simple and efficient; and,
- the format can be served by GeoServer, the open source geospatial data server application, using standard protocols, e.g., WFS (vectors), WMS (rasters), and WMTS (rasters).

There are several implementations of the GeoPackage specification available, though we have chosen to use the GDAL library for our implementation. We are implementing format extensions to support the additional layers that VBS Blue requires: surface material maps (raster with metadata), coastline data (vector), hydrography (vector).

We are using open and public sources to populate the baseline database cells. DTED is used as the elevation data source, MODIS and the Global Forest Change Map provides surface materials, World Data Bank II provides coastline data and river centerlines, and OpenStreetMap (OSM) provides road centerlines and building locations and footprints. The data layers will be clipped to the bounds of each cell, reprojected to geodetic coordinates (WGS84) and stored in the appropriate GeoPackage file for binarization and later use, e.g., we need to recompile a portion of the baseline database.

Other geospatial data sources can easily be used, either by republishing the data to a format supported by an existing VBS Blue plug-in, e.g., most GDAL-supported formats are included here, as well as some common streaming protocols like HTTP, or by implementing a VBS Blue plug-in tailored to the new data format. This is one of the methods used internally to import additional data required to improve a virtual terrain location. Once the improved region of the terrain is evaluated and accepted, it can be published as part of the baseline.

More complicated formats can also be handled this way. For example, CDB could be served as a network geospatial data source to VBS Blue or to another web-enabled geospatial data consumer application by converting the data layers to GeoPackage, then publishing the resulting GeoPackage database file to a GeoServer. VBS Blue could then render the data as it is requested from the server.

MANAGING DATABASE UPDATES

There are three scenarios where it would be necessary to import new geospatial data into VBS Blue. The first is to update a data layer, e.g., the road vectors, in one or more baseline data cells because it is out of date. A second scenario is to import custom data over a local region of the globe, e.g., change the surface map, to supplement the virtual terrain in that region. A third scenario is to support dynamic terrain updates and edits.

For baseline data updates, we need to identify all of the cells in the baseline database that overlap with the bounds of the data being imported/updated, as those are the cells in our managed database that will need to be changed. We would then lock the targeted data layer in those cells and do the update. Since the cells are independent from one another, this process can be done in parallel. Each cell update will also need to trigger a compilation task to update the compiled version of the updated cell data.

We can handle custom data import in the same manner as baseline data updates, however we intend to store those changes in a separate custom data GeoPackage file associated with the cell so that we can ensure that any user-provided changes take precedence over the baseline data.

Finally, to support terrain editing and dynamic updates, we need to permit changes to the geospatial source data, which will in turn trigger a recompilation and cache update. The delta between the current state stored in the database and the state stored in the editor can be saved to provide the basis for the next change and delta calculation. This history of terrain changes allows us to support undo/redo operations.

NETWORK-BASED RENDERING USING COMPILED DATA

The compiled or binarized form of the geospatial data is cached by VBS Blue's plug-in system to improve rendering efficiency. Currently, the compiled data is stored on the local disk of the machine where VBS Blue is running. However, this data can be stored alongside the geospatial data source from which it is derived, then streamed back to the renderer when requested. Such a system allows multiple users to benefit from the improved performance of the cached data.

CONCLUSIONS

VBS Blue is a new whole-earth rendering technology Bohemia Interactive Simulations derives realistic procedural details from moderate- and high-resolution geospatial data. In this paper, we have described our on-going efforts to build on these innovations to design and implement a database that stores the source data, making it easier to maintain and deliver efficiently via a server. This effort is intentionally built on open formats and open source technologies like Geopackage and GeoServer so that other runtimes might be able to make use of the design and extensions, making it easier to interoperate and share data.

ACKNOWLEDGMENTS

The authors happily acknowledge the staff of TerraSim and Bohemia Interactive Simulations for their congenial interactions and continued technical support of this effort. We also acknowledge the patience of our session head and the comments of the IMAGE Society reviewers.

PRIMARY AUTHOR BIO

Wilson Harvey is Director of Software Engineering at TerraSim, Inc. At TerraSim since 2006, Mr. Harvey supervises the development of new technology and source data preparation products which complement and inter-operate with TerraTools, TerraSim's geospatial database construction product. Prior to joining TerraSim, Mr. Harvey held a succession of research scientist positions within the Department of Computer Science at Carnegie Mellon (1985-2006) working in the areas of computer vision, cartographic feature extraction, knowledge-based systems, and parallel processing. Mr. Harvey is the author of over 25 journal and conference publications in these areas.

REFERENCES

[1] OGC GeoPackage WG. (2017). *GeoPackage Encoding Standard*. Retrieved from GeoPackage, an open format for geospatial information: http://www.geopackage.org/

[2] OGC GeoServer WG. (2017). *GeoServer*. Retrieved from GeoServer, an OGC-compliant geospatial data server: http://geoserver.org/