

AUTOMATING THE GENERATION OF LARGE SCALE ENVIRONMENTS

Daniel J. Hershey
TerraSim, Inc.
Pennsylvania, United States

Joshua A. Klein
TerraSim, Inc.
Pennsylvania, United States

David M. McKeown
TerraSim, Inc.
Pennsylvania, United States

William J. Starmer
TerraSim, Inc.
Pennsylvania, United States

ABSTRACT

The automated generation of large scale environments requires the ability to address "scale" in a number of interesting ways. Typically "large scale" implies a large area of extent. Increasingly it can mean relatively small areas which are modeled at an extremely high level of detail. In this paper we discuss automation techniques for both types of environments. Naturally each situation requires a different approach, but the common theme is reduction of complexity in generation of large scale environments. This is achieved by the application of algorithmic principles working on source data to reduce the complexity of the environmental database problem.

INTRODUCTION

Modeling and simulation requirements include the ability to generate large area operating environments as well as the representation of high resolution content within smaller operating areas. The need for visual and constructive simulations, highly correlated to real-world environments to enable live participation, has become the cornerstone for future LVC simulations. Large scale environments can thus be found in terms of area of coverage as well as high density of detail within relatively small but complex areas. Careful algorithmic design and system engineering show that computational efficiency combined with data flow organization can result in high throughput database generation on standard technical workstations.

In this paper we describe techniques used to support both situations simultaneously. For large scale areas we have developed a user configurable process flow that supports the use of "Batch Mode" scripting to iterate over large numbers

of geospatial data, organized in fixed units, usually one degree cells (80km x 100km approx.), which can be composed into larger areas of coverage. The advantage of a scripting interface over a visual interface is that user developed applications can be used to control these batch scripts, can automatically monitor database generation progress, and can produce human readable processing logs which can be used to support validation and verification of the database process.

In terms of high density databases, particularly for urban modeling with hundreds of thousands of unique buildings, there are a number of techniques that have been shown to be effective for visual simulation. Small numbers of carefully modeled buildings can often be cleverly placed and oriented to give the appearance of urban variability, especially for flight and nap of the earth applications. However this approach breaks down for ground based visual applications. Increasingly, the quality of geospatial source data gives unique building footprints, accurate height information, and roof types and orientations.

We describe an approach to the more accurate representation for city-scale simulation. It relies on the observation that representative buildings can be derived from the source data itself, and by developing best-fit matching criteria, one can automatically reduce, by several orders of magnitude, to a set of unique buildings that closely match the original source data. Since this process is purely data dependent, it is completely adaptive to the underlying data. In this paper we will provide an example of this technique and process for visual and constructive simulation runtimes as well as implications for a significant increase of high fidelity content in serious game runtimes.

AUTOMATING LARGE SCALE ENVIRONMENTS

The economic production of large scale simulation environments requires special considerations beyond the traditional software solutions that are developed to support a database engineer working with geospatial source data. In particular, while user-centric interfaces support iterative development of a database methodology as well as the exploration of source dataset content, there are advantages to larger scale automated engineering systems to support scalable production of environmental databases. Once a large area source dataset is organized, in terms of area of coverage and data content (layers of geospatial data), and validated with respect to attribution, the user focused model can become a bottleneck in repetitive database production.

The goal, then, is to transform the individual workstation environment into an autonomous production environment based on user configurable extensions to the underlying database generation solution. The key observation is that by organizing source data into manageable work units, one can develop a simple processing language that supports large area production by iterating processing over each work unit. In the remainder of this section we describe how we have augmented our database production system, TerraTools, with a command language based upon simple process control primitives. This scripting language, combined with the process oriented architecture in TerraTools, supports user customization to achieve large scale database generation without requiring a significant engineering effort.

BATCH MODE SCRIPTING

We call the integration of the TerraTools database generation engine with a simple scripting command language (Tcl) "Batch Mode processing". This integration allows the existing TerraTools' flowgraph graphical user interface to be controlled directly from the batch mode script. Users can generate processing templates with the TerraTools GUI and apply them with batch mode in automated fashion. Figure 1 gives a schematic view of the control structure provided by batch mode processing.

There are several uses for the TerraTools batch mode process. For complex database construction, a prototype project can be designed and debugged using the TerraTools GUI and then applied, repeatedly, with batch mode on multiple subareas of an overall database project. This can allow for the automated generation of multiple high resolution insets to be integrated into a larger area project.

Batch mode processing can also be used to support the

automatic generation of multiple geotiles, or one degree geographic cells, for large area simulation. By standardizing a single batch mode script and organizing geospatial data into an array of source data directories, TerraTools can be invoked to build each geotile, leaving the process results, including intermediate derived data and node execution messages, in the associated file structure for subsequent analysis and validation. Simple geospatial processes, such as reprojection of cartographic data, generation of building models from GIS footprint data, and generalization of complex vector data, can all be accomplished from batch mode scripts without building a full visualization output. Finally, the batch mode processing model is also applicable for the development of user defined database generation servers that utilize Tcl scripting to define their functionality.

Batch Mode Commands

One of the key insights of Batch Mode processing is that batch mode commands do not directly process geospatial data. Instead they provide the ability to link and control processing nodes that transform the geospatial data. Table 1, Table 2, and Table 3 show current batch mode commands organized into three operational classes: Control Commands, Graph Building Commands, and Management Commands.

Control commands are used to load, manipulate, and control the execution of a TerraTools project graph within batch mode. Unlike Batch Mode scripts, TerraTools project graphs are collections of independent processing nodes that can be independently executed using a linked dependency graph that defines node data dependencies. As long as it has a valid input, the TerraTools processing node can execute, providing for the potential for highly parallel execution.

Figure 1 shows a representative TerraTools project graph with processing nodes as named boxes and source data dependencies represented as directed arcs between the nodes. Colors of the nodes in the graph indicate the node processing state. Green nodes have completed processing and their outputs are available as inputs for processing nodes further down in the graph. Yellow nodes have completed processing with a non-fatal error. These errors are logged and are available to the batch mode processing control process as well as human readable text files. Grey nodes are unexecuted nodes.

Purple nodes are actively running. In this execution snapshot example, five processing nodes are running in parallel. Finally, light blue nodes indicate that they are waiting for one or more of their source data inputs. They will be scheduled for execution once their data dependencies are satisfied.

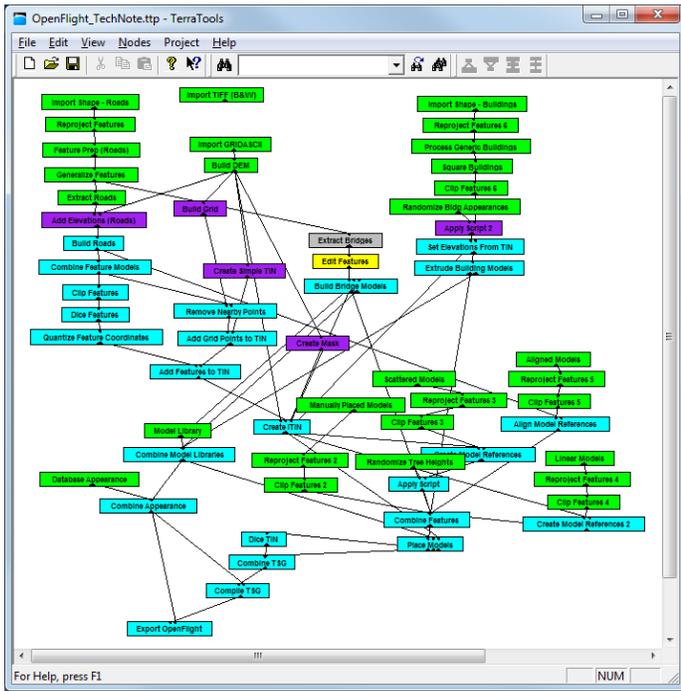


Figure 1: Sample TerraTools Project Flow Graph

The most commonly used control commands are tsdload, tsdgetvar, and tsdsetvar. These set project variables to define terrain database bounds and coordinate system settings. To iterate through and apply changes to specific nodes in the process flow graph, the tsdlistnodes command can be used. Nodes are identified by their unique TerraTools node name.

Command	Purpose
tsdcancelbuild	cancel pending node updates
tsdcheckfilebounds	determines data coverage
tsdcheckfiletype	run file recognition algorithms
tsdlistnodes	lists names of all nodes
tsdwait	waits for pending node updates
tsdnupdate	updates specific node
tsdload	loads a project file
tsdsetvar	sets project variable
tsdgetvar	gets project variable value

Table 1: Batch Mode Control Commands

Graph Building Commands allow users to construct a TerraTools processing graph within a scripting or programming environment. All of the primitives required to procedurally build an executable TerraTools project, by calling out the names of processing nodes and connecting inputs and outputs, are available within the Graph Building Commands.

The most commonly used graph building commands are tsdnetoption and tsdnetoption. Node options are specific to each node, but usually contain specific functions that change how the geospatial source data is modified or specific attributes and settings are handled. Adding and deleting nodes from the graph can be accomplished with the tsdnadd and tsndeldelete commands. To get the processing status of a node the command tsdngetstatus is run. Node status gives information about whether the node can be updated or if it has finished processing successfully or with warnings or errors.

Command	Purpose
tsdcleanup	deletes intermediate files
tsdconnectnodes	connects node outputs to node inputs
tsddescribetype	return TerraSim filetype description
tsdedelete	deletes arc between nodes
tsdlistnodetypes	lists valid node types
tsdnadd	creates new node
tsndeldelete	deletes node
tsdnexpand	replaces \$(TERRASIM_HOME)
tsdngetoption	gets current node option value
tsdngetstatus	gets node state
tsdninfo	gets node info
tsdnname	gets node name by ID
tsdnode	multi-purpose node command
tsdnsetoption	sets node options
tsdnsetname	renames a node

Table 2: Batch Mode Graph Building Commands

Batch Mode Management Commands support access to runtime actions and information during Batch Mode execution. This includes the creation of a process log file and

IMAGE 2011 Conference

the accumulation of all individual node status, warning, and error messages. Finally, the TerraTools project graph that is the result of Graph Building Commands can be saved as a project (.ttp) file.

The most common management commands are tsdlogfile, tsdsave, and tsdngeterrors. With tsdlogfile, the logfile is generated to store a human readable list of node completion times as well as input files, node warning and error messages. The script can also use the error and warning messages directly with the tsdngeterrors command when that command is run on a node directly. Saving of multiple copies of a project is also supported via the tsdsave command.

Command	Purpose
tsdlogfile	creates log file
tsdngeterrors	gets warnings and errors
tsdngetoutput	displays nodes standard output
tsdsave	saves project (.ttp) file
tsdstatus	prints table of nodes

Table 3: Batch Mode Management Commands

Multithreading and Scripting

TerraTools is multithreaded, meaning that it can simultaneously run multiple nodes in the project graph. As a consequence of multithreading, TerraTools Batch Mode scripts must use a special command to ensure that projects are properly run to completion. To understand why multithreading has implications for scripting, let's look at an simple example. Most TerraTools project graphs culminate with a Compile TSG node. Tiled Scene Graph (TSG) is the common internal representation that is constructed prior to TerraTools export to correlated runtimes. Thus, the simplest invocation of Batch Mode is to load a project, instruct it to update and then exit Batch Mode processing.

```
tsdload project.ttp
tsdnupdate {Compile TSG}
exit
```

Unfortunately, this script will not work as intended. tsdnupdate works just like its "Update" counterpart in the TerraTools graphical interface: it requests that a node be updated, then returns immediately without waiting for the update to complete. The update is handled by a separate processing thread. The script above will load the project file, request an update, and immediately exit before the processing thread has had time to execute the update. To solve this problem, we use the "tsdwait" command, which

waits for all pending node updates to complete before allowing the script to continue.

With this command, it is easy to fix the script to get the intended behavior:

```
tsdload project.ttp
tsdnupdate {Compile TSG}
tsdwait
exit
```

Batch Mode Runtime Manipulation

As mentioned previously the Batch Mode Building commands allow for dynamic modification of a standard TerraTools project.ttp. These prototype project files can contain a large number of similar database construction tasks. However, when certain data elements are not present, rather than build and maintain a family of related but slightly different prototype project, users can specialize a single prototype project by manipulating the underlying graph.

For example, consider a particular project that does not involve the use of road data. The tsdndelete command in TerraTools Batch Mode can be used to easily prune away the section of the process flow graph that handles roads. The script for this example would look like the following.

```
tsdload prototype.ttp
tsdndelete {Build Roads}
tsdnupdate {Compile TSG}
tsdwait
exit
```

This script loads the prototype project graph, deletes the Build Roads node, and then updates the Compile TSG node, which will update all of the nodes upon which Compile TSG depends. Since the Build Roads node has been deleted, its output no longer feeds into the main graph and the processing chain that feeds Build Roads will not be executed.

This strategy allows users to maintain one prototype project graph capable of handling a wide variety of cartographic source data processing, while simple scripts tailor this processing to specific cases. This is particularly relevant to large database construction.

One method for constructing large databases using batch mode works in four stages. First, the feature data for the entire database is prepared in a separate data prep project. Tile boundary elevation profiles are defined here as feature data, to guarantee that tile boundaries will match. Second, a prototype project is created in the TerraTools GUI for building a single tile of the database. Third, batch mode is

used to build each tile with the prototype project file. Finally, the tiles are assembled and exported in the desired format.

Many variations on this theme are possible. For example, the feature data could be prepared in a GIS or in another database generation system. In another example, each tile project could export the corresponding piece of the database, rather than combining the tiles and then exporting the combined database.

Another application for Batch Mode processing is in the construction of a database generation server. Batch mode can be used to implement a server which automatically produces terrain databases. In this processing model, the GUI is used to create a prototype project capable of generating all databases that could be built from a cartographic source dataset. A server receiving a terrain generation request would execute the following sequence of steps:

- load the prototype project and set the database bounds
- identify the input cartographic data for the requested area
- set properties on the appropriate import nodes to load it
- request an export node update and wait for completion
- deliver the finished database to the client process.

Such a process, while bounded by the quality of the cartographic input data, is nonetheless useful for meeting requirements of low cost and/or rapid turnaround for database construction.

An Example of Large Area Batch Mode Processing

Figure 2 depicts the graphical process flow of Batch Mode script execution previously described. Many constructive simulations, e.g. OneSAF, JointSAF, JCATS, and others, require that large area databases be "tiled" in one degree cells of the earth defined in latitude and longitude. These "geo-cells" or "geo-tiles" are used to compose the overall simulation area. Developers, and their software tools, must ensure that the continuity of elevation and vector data is preserved across the cell boundaries. Likewise visual databases generally require some spatial organization, usually smaller than a 80km x 100km geocell, organized as quadtrees in a multiple resolution hierarchy.

Source data, organized by Geotile Reference System (GTRS – ISO IEC 18025) cells and stored on a common file system, is pre-processed within TerraTools. This generally involves reading in large data coverages and clipping datasets into geocell boundaries, taking into account the coordinate system being used for the cells. A second pre-processing step creates a shared data structure with all of the necessary boundary condition data for each cell.

As the Batch Mode script iterates across all of the GTRS data cells, the processing results are left in the folder associated with the source data. This includes all process log files, error files, and other results of the Batch Mode execution. Thus process verification and validation is supported. Some users have developed simple scripts to analyze or to collect together the log files.

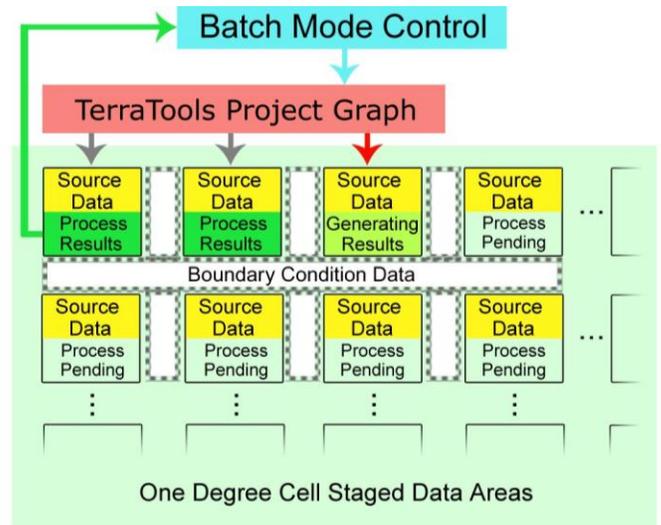


Figure 2: Batch Process Flow

Figure 2 illustrates that the Batch Mode Control is separate from the underlying sequential execution of the TerraTools project flowgraph. While in most cases the same flowgraph is applied to a standardized dataset, there is no reason why data specific project graphs could not be controlled or executed. Since project (node level) parallelism is employed using multithreading within the execution of each of the geo-cells, these results were achieved using a single technical workstation. This also minimizes communication costs (latency) compared to solutions that have to distribute data on a local network in order to achieve parallelism.

Batch Mode processing has been successfully used to create a number of large scale databases from assemblies of geocells. Figure 3 shows the GTRS bounding box for a 130 cell Batch Mode process for a source data set over the United Kingdom. Figure 4 and Figure 5 shows the resulting OneSAF 5.0 (OTF 8.0) multi-cell database and a correlated visual database in an OpenSceneGraph viewer.

Generation time for both products averaged 15 minutes per cell using a 64-bit Windows workstation with a single 2.8ghz quadcore processor and with 24 GB of system memory.

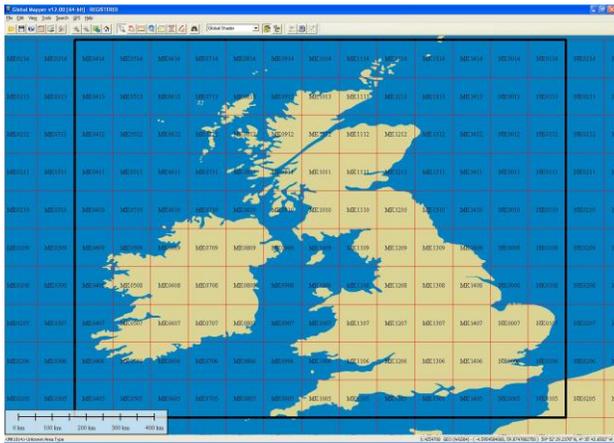


Figure 3: GTRS Bounding Box for 130-cell Process

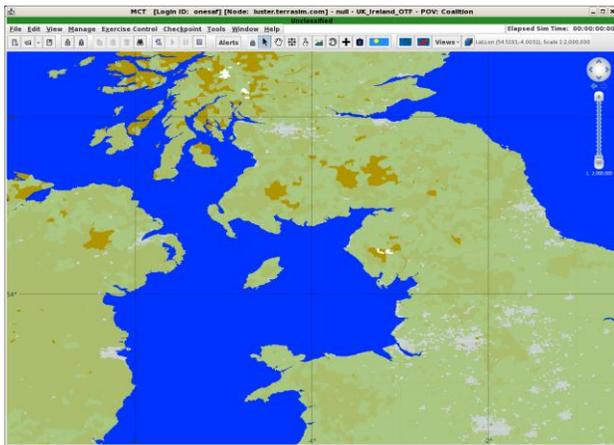


Figure 4: 130-cell OneSAF 5.0 (OTF 8.0) Database

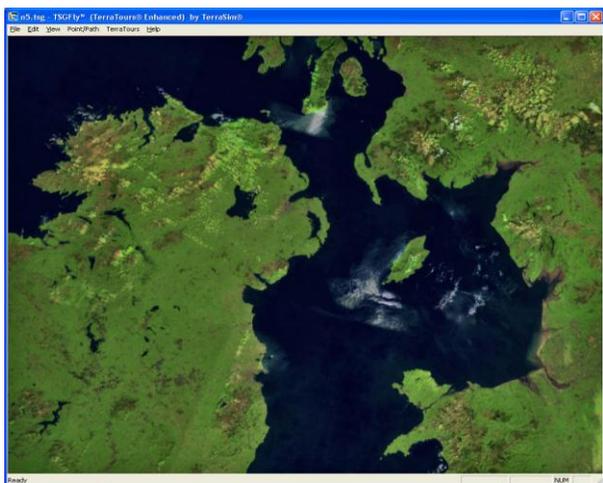


Figure 5: 130-Cell OpenSceneGraph Visual Database

ADAPTIVE APPROXIMATION FOR LARGE SCALE DATABASES.

Many commercial database generation systems support some degree of parametric building model generation from simple GIS building footprints. This support, depending upon the implementation, can greatly simplify the time from source data ingest to 3D model generation. As a result of this, and other commercial technologies, high density urban databases with hundreds of thousands of unique buildings are currently possible for visual and constructive simulation. However in many cases, the use of very large numbers of parametrically generated building models, geo-specifically located, presents a significant cost driver for applications. Correlating visual and constructive runtimes with serious games applications poses other, often runtime dependant, challenges.

For example, typical serious game runtimes exact a significant performance penalty when a large number of unique models are present. These game engines are optimized to render a large number of instances of a relatively small number of models, usually organized as external model references. As a result, small numbers of carefully manually modeled buildings can be cleverly placed and oriented to give the appearance of urban variability, especially for high flight and nap-of-the-earth applications. However this solution breaks down for ground based visual applications. Increasingly the quality of geospatial source data with unique building footprints, height information, roof types and orientations need to be preserved in the resulting simulations, particularly those used for mission rehearsal. Further, when networking disparate simulations, correlation is reduced when one system requires simplified models.

We describe an approach to the generation of more accurate representation for city-scale simulation particularly for runtime systems based on game engine technology. This work relies on the observation that a reduced representative set of buildings can be derived from the source data itself. By developing best-fit matching criteria, one can automatically reduce the source data, by several orders of magnitude, to a set of unique buildings that closely match the original source data. This set of building models can then be used in the simulation as a de facto model library, allowing for efficient model referencing. Since this process is purely data dependent it is completely adaptive to the underlying data.

Building Footprint Equivalence Determination

We have developed an automated method for taking a very large collection of building footprint data and determining a subset which is considered representative of the dataset. This

IMAGE 2011 Conference

representative set can then be used in place of the remaining buildings. This supports complex building simulation without exceeding limitations on unique model counts.

The process fundamentally revolves around taking two building footprints, computing a similarity metric, and obtaining a quality score. The most interesting question is which similarity metric to use to preserve salient building shapes. A second consideration is whether to use a set of pre-defined exemplar buildings or to attempt to derive that set from the data itself.

We decided on a metric based on the comparison of geometric histograms, which builds a histogram of all the relative angles and lengths within a building footprint, and then does a histogram comparison and produces a similarity score. While this provides geometric similarity, other tests based upon existing building attribution were added to take into account non-geometric properties.

Our initial implementation performed a pairwise comparison of all of the building footprints, which provided initial results from which we could fine tune matching metric and scores.

However the computational complexity of that naive process did not scale to large urban datasets. In order to get around this problem without compromising representative power of equivalence checking, we sort building footprints into groups based on their area and perimeter. This divide and conquer approach worked because the equivalence metric would not have buildings in different groups to be more similar than those within their own group. Within each equivalence class we retained the pairwise matching algorithm.

Any equivalence metric still requires a threshold value to determine when to declare footprints to be the same. This threshold allow users to control the number of building equivalence classes and therefore the number of unique models that will be used to represent all of the buildings in the original source data set. The process to generate the equivalence classes is fast enough that in practice a set of thresholds is used, of increasing tolerance to small differences in shape between the reference group model and the test model. This allows users to select a cut point threshold to determine the number of unique equivalence classes that will be present in the resulting simulation based upon pragmatic system performance considerations.

Processing Results for Equivalence Class Generation

In this example, 287,383 unique building footprints from GIS source data were divided into 260 equivalence classes based on histogram matching equivalence of 20% (same for all

threshold tests below).

Similarity threshold setting	Unique building at this threshold	Percent model reduction
0.075	25,056	91.28%
0.10	13,395	95.34%
0.125	7,395	97.26%
0.15	4,919	98.29%
0.20	2,357	99.18%
0.25	1,388	99.52%
0.30	949	99.67%

Table 4: Processing Results for 287,000 Input Footprints

Figure 6 shows a small area from the original source dataset. A quick visual inspection shows that there is a diversity of building shapes in the source data, with blocks of buildings being composed of collections of individual structures sharing a common internal wall. Figure 7 – Figure 10 show the result of applying the equivalence class to each of the buildings in Figure 6. Matching cut points at .075, .10, .15, and .30. are shown in these figures with Table 4 providing the quantitative reduction in the number of unique models. Even with a fairly conservative threshold we can achieve an order of magnitude reduction in the number of unique models required to represent the entire dataset.

Of course, as the number of unique models decreases, the deviation from the original dataset increases. We have added white circle areas within each of the figures to highlight areas when large geometric changes become more apparent as the adaptive approximation becomes more extreme.

As a further refinement to the basic process, we added an option to remove the largest buildings in the source dataset prior to computing equivalence. This was based on the observation that as buildings get larger and have more complex shapes, smaller differences have greater impacts on the usability within the final environment. This was accomplished by a simple histogram of building area, setting an appropriate cut point to select out those buildings from the dataset above, and adding them to the unique model count. The use of the building area histogram preserves the ability for the overall process to be purely data dependant.

The building equivalence class process is being integrated as an enhancement into TerraTools' current Urban Details™ processing chain. This will allow users access to advanced fully automated parametric model generation from simple GIS building footprints. Since the results can be previewed at the source data level, users can choose appropriate cut point based upon number of resulting buildings, system load requirements, or even visual inspection.



Figure 6: Original data



Figure 7: 0.075 threshold setting



Figure 8: 0.10 threshold setting

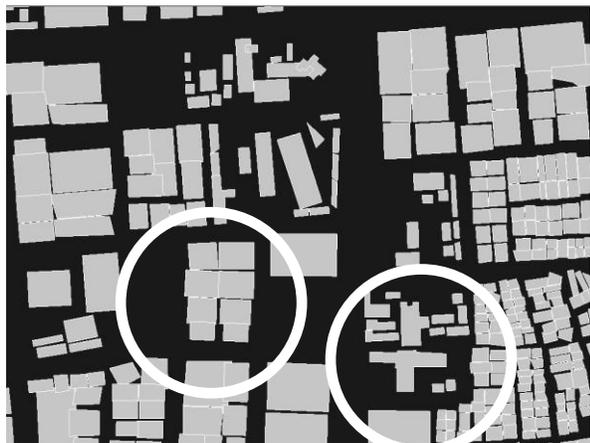


Figure 9: 0.15 threshold setting



Figure 10: 0.30 threshold setting

IMAGE 2011 Conference

CONCLUSIONS

The rapid generation of large scale simulation databases can be facilitated by the application of a number of innovative technologies. This reflects the breadth and diversity of the problem of building sophisticated visual and constructive simulation environments to support users with different training, simulation, and mission rehearsal objectives.

In this paper we have described two very different approaches, one which addresses the physical size of large area databases using a scripting language to control the database production process, and the other which seeks to reduce the complexity of urban environments while preserving the geo-specific properties of the source data.

Both techniques represent implemented and tested engineering solutions which are available to the modeling and simulation community. There are, of course, many other places in the simulation environmental generation design space where innovation is required to drive down costs and improve overall quality of the runtime database product.

ACKNOWLEDGMENTS

We acknowledge Julius Yako (TerraSim) who designed and implemented much of the underlying Batch Mode scripting language. Several customers have used Batch Mode scripting to support large area database construction projects, most notably, the OneSAF ERC team in Orlando, and the Alion Science and Technology, JETS laboratory, at U.S. Army Geospatial Center (AGC). Both groups have contributed to the refinement of the process during its development and deployment. Finally, the authors acknowledge the IMAGE Society referees who provided specific and substantive comments on the original draft of this paper.

AUTHOR BIOGRAPHIES

Daniel Hershey, Senior Software Engineer.

Mr. Hershey received a BS in Computer Science and minor in Physics with University Honors from Carnegie Mellon University in May 2006. Since joining TerraSim, he has contributed to our automated building interior generator, parametric roof generation, and conversion of TerraTools blueprints to OneSAF UHRB building representations. He has had extensive experience in developing our CTDB to OTF conversion process, and is experienced with the automated generation of urban geometry for serious games and constructive simulation systems.

Joshua Klein, GIS Analyst.

Mr. Klein received a BS in Geography with a specialization in Geographic Information Science from the Schreyer Honors College program at Penn State University in May 2009. He is a member of Gamma Theta Upsilon, The International Geographic Honor Society. Since joining TerraSim, he has contributed to the automated generation of large area terrain databases in both visual and constructive formats. He designed the system used to generate multi-cell OTF terrain databases with TerraTools Batch Mode.

David M. McKeown, President, TerraSim, Inc.

Mr. McKeown is the President and a co-founder of TerraSim, Inc. He is also an Adjunct Research Professor of Computer Science at the Computer Science Department of Carnegie Mellon University, Pittsburgh PA. Founder and head of the CMU Digital Mapping Laboratory (MAPSLab), Professor McKeown has over 35 years of experience in the areas of image understanding for the automated analysis of remotely sensed imagery, digital mapping and image/map database systems, and geospatial visualization. Currently, his work at TerraSim is focused on automated generation of complex urban environments to support next generation correlated visual and constructive simulations.

William J. Starmer, Principal Geographic Technologist.

Mr. Starmer received a BA in Geography in 1994 from the University of Maryland, Baltimore. He has over fifteen years of experience in GIS and Remote Sensing. He developed and presented a well-received three hour specialist course entitled "Issues in Urban Database Construction", presented over three successive years (2002-2004) at the IMAGE Society Conference in Scottsdale, AZ. He is responsible for TerraSim's process architecture for creating correlated visual and constructive databases using TerraTools, including the generation of correlated OneSAF OTF and VBS2 databases.

REFERENCES

- [1] TerraSim, Inc., 2011, *TerraTools 4.0 User's Manual*, Pittsburgh, PA, United States, 1-334.
- [2] J.L. Giuliani, J. LaDieu, and D.M. McKeown, *Parametric generation of street level details for urban visualization*, Image Society 2008 Conference (2008).
- [3] J.L. Giuliani, D.M. McKeown, J. de la Cruz, and T. Sotomayor, *Automating the generation of urban details*, Image 2007 Conference (2007).
- [4] B.B. Welch, K. Jones, and J. Hobbs, 2003, *Practical Programming in Tcl and TK*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1-127.