

PARAMETRIC GENERATION OF STREET LEVEL DETAILS FOR URBAN VISUALIZATION

Joseph L. Giuliani
Principal Visualizations Systems Architect
&
Juliet LaDieu
Visual Database Engineer
&
David M. McKeown
President
TerraSim, Inc.
Pittsburgh, PA

Abstract

One of the most challenging problems in the rapid construction of complex urban environments is the intricacy of the geospatial source data and the need for fully automated generation of urban details in order to meet critical database construction timelines. In this paper, we present a set of new techniques for the parametric generation of “urban details” that both improves the visual appearance of the resulting environmental database and supports correlation with constructive simulation databases.

The addition of “urban details” addresses the mismatch between the availability of detailed GIS source data and the desire for more realistic and less pristine-looking visual environments. Using existing geo-specific source data as anchor content, we describe techniques to intensify the urban environment with user defined and culturally appropriate models and appearances.

Several automatic generation primitives have been developed to allow for parametric generation of model placement points using a rapid generate-and-test paradigm that supports priority lists, automatic model orientation, and model integration into an integrated triangulated irregular network (ITIN). Other constraints support limits on model placement, density, and application of model vanishing ranges as well as level-of-detail.

Introduction

Expectations for modeling and simulation database visual content continue to increase, somewhat driven by experiences in virtual world and computer game developments where highly realistic visual environments can be cost effectively manually modeled with the goal is to provide an immersive visual experience that supports the game storyline. This cross-over between military modeling and

computer gaming has been dubbed “serious games.” Further, online visualizations of real world urban environments including Google “StreetView” displays and Microsoft Virtual Earth have shown that terrestrial photography and high resolution aerial imagery can be used to provide a compelling sense of place when delivered via interactive viewers. In the latter cases, the generation of urban visualizations is limited to those areas where one can drive and record terrestrial video and still photography. And perhaps, situations where the budget and timeline would make a military modeling and simulation program manager envious.

The use of high resolution satellite imagery for broad area coverage coupled with ultra high resolution aerial imagery for high resolution insets is now commonplace for aircraft simulation and provides geospatial realism by virtue of being geo-specific imagery. Nap-of-the-earth, low altitude helicopter simulation environments also rely on high resolution geo-specific imagery, but also need to address the generation of 3D cultural data in limited areas of interest as high resolution insets. Depending on the locale, this can also include a large number of individual tree and foliage models, as well as vertical obstructions such as utility lines and telephone poles.

Ground operations, particularly in urban environments, require the highest fidelity in terms of environmental modeling, and unfortunately do not succumb to simply increasing the resolution of remotely sensed imagery. While environmental appearance is very important in urban visualizations, it is typically captured by terrestrial textures and, more importantly, the generation of geo-specific environmental models derived from vector GIS and/or CAD design data. Advanced visual database generation systems are able to convert vector data to 3D geometry in a variety of ways.

Urban Details

The literal conversion of GIS and geo-referenced CAD data into correlated visual and constructive simulation databases is well understood. Typical urban cartographic features such as buildings, roads, bridges, natural features such as parks, forests, lakes and linear drainage can be portrayed by generating 3D polygonal geometry (terrain skin) and applying a mix of geo-typical and geo-specific photo textures for appearance. When models are referenced in the source data, automating their placement and orientation is generally straightforward in advanced database construction toolsets. Additional processing can include the automatic scatter of trees within a polygonal park area, the automatic placement and orientation of point reference models such as street signs, light poles, trash cans, etc. and the construction of spaced power and telephone lines. This process is controlled and defined by specific references in various GIS source data layers.

Some of these conversions involve a mix of geometry generation as well as spatial reasoning. For example, one would expect that trees would not be scattered in roads or on top of buildings, linear pole models would be controlled to be on sidewalks and not form road navigation obstructions, and point models would need to be oriented with respect to centerlines of roads, and so on. Automation must be highly reliable since the practical cost of having to detect and manually edit and correct misplacements can exceed the value of automation. Nevertheless, there is a direct relationship between geometry generation and model placement and feature specification in the source data.

In this paper, we describe solutions to the problem of creating interesting urban environments when only the most basic source data is available and specific details of the environment are lacking. We call the autogeneration of missing urban content “urban details” processing. Two examples of the application of urban details processing are described. One supports the addition of culturally appropriate models and appearances in a dense “North American” urban environment. Similar processing is highlighted in a “sample desert village” example. In both cases we start with the most basic set of building footprints and road centerlines which are typically available for basic geo-specific data processing. Two general classes of urban features are added: integrated models and placed models as well as modifications to generated geometry and the use of textures to improve the visual appearance.

Table I organizes the creation of urban details based on placement methods, geometric constraints and

types of geometric processing. Generally, either areals or linears can be used to provide constraints for the placement, and in some cases linears can be derived from areals using medial axis transformation. Ultimately point model (or multi-model) references are created, with orientation information. These model references can be exported, on a layer by layer basis, to form a persistent representation of the urban details process. This is particularly important when non-visual constructive simulation databases are constructed since these data layers provide complete correlation with the visual representation.

Some urban clutter features such as sewer grates and pavement anomalies are directly integrated into the ITIN representation while maintaining spatial relationships with existing curbs and sidewalks. Constraint processing determines suitable placement areas that do not conflict with existing features, based upon user defined parameters. Likewise, using feature topology derived from the actual geo-specific source data, features such as light poles, power lines, fire hydrants, and parking meters can be automatically placed and displaced relative to preexisting urban features acting as anchor points. Collision detection is automatically performed so that urban details models will not overlap or be placed unnaturally close to one another. In the case of the desert village example, a set of appearance modifications, also based upon building footprints and road centerlines, are procedurally generated.

Once model position and orientation constraint processing is completed, all of the generated urban details models can be exported along with appropriate attribution as derived source data for correlation with the generation of computer generated forces (CGF) databases. This representation can also be archived and edited as necessary to support downstream geospatial processing.

We will illustrate the visual impact of urban details processing using multiple examples from the two sample databases. We maintain that this approach to feature intensification can be used to greatly improve the sense of realism in otherwise sterile urban visualizations without resorting to manual modeling. The processing time for the generation of urban details adds a minimal additional cost to basic source data processing since application of placement and search constraints can be locally computed.

To illustrate this concept, Figs 1-6 show three separate views of the North American urban environment, before and after urban details processing. Figs 1 and 2 detail placement of a sewer grate and bus shelter, 3 and 4 show sidewalk multi-textures, and 5 and 6 depict lights and road textures.

Table 1: Urban Details Feature Types

Source Data	Placement methods	Constraints	Geometric Processing	Model Examples
Areals	Scatter	Density / proximity	Model point reference	Trees, shrubs, AC units
Linears	Alignment	Spacing / alignment	Integrated TIN	Sewer and vent grates
Linears	Orientation	Spacing / orientation	Model point reference	Traffic signs, shelters

The Urban Details Process

The simplest form of automated placement of urban objects is a stochastic scatter of point reference models within an areal feature. For example, areals designated as parks or vegetative regions can be used to scatter a randomly selected set of models from a palette of vegetation types. In the sample urban database, park areals were used to scatter tree models with mulch pile models at the base. In order to provide some variability, each park areal was assigned one of three mulch textures for placement with the trees models and three sizes of mulch circles were randomly used within each areal.

The creation of more sophisticated urban details involves several steps where the user first defines the set of model primitives and then defines how the placement of those primitives will be controlled. For example, in a North American context, it is not atypical for urban source data to contain areal polygons delimiting land use such as roads and road medians, sidewalks, “urban ground,” and park areas. In our design, these constraints are embedded as placement rules that are implemented as Tcl scripts. These scripts have access to intermediate geometry derived from the original source data that includes boundaries, centerlines, derived intersections (topology) and all of the original feature attribution.

Model Placement Using Constraints

In order to establish precedence of model placement, a single model is selected that has the property that it will span a linear swath with regular spacing. The particular model is not important; the key is establishing a regular spacing interval within which other model placement can occur. This is the sub-spacing interval.

In the sample urban database, street light models were used as anchor placements at a user-defined 25 meter spacing along both sides of the road areals including median areals. Once placed, a second Tcl script was used on the urban ground areals to assign a MODEL attribute with one of three possible values: 1) *None*: Areal marked with “none” had no models placed on their edges; 2) *Tree*: Areal marked with “tree” used a mega-model composed of trees with a metal grate centered at their base placed every 6

meters along the urban ground areal edges; 3) *parking_meter*: Areal marked with “parking_meter” placed one spaced every 5 meters.

This has the effect of placing tree and parking meters with respect to the street lights, without regard to collisions or proximity testing, which were performed next based on these proposed positions. This reduces the combinatorics of placement, and allows a precedent to be established by how proposed placement points are pruned from the environment.

First, proposed tree model points that fell close enough to a street light that the tree grate and the street light would overlap were deleted. Using the tree grate model and tree model together as a mega-model ensures that a single proximity test could be performed. This avoids street light models being placed on top of tree grates. Points that fell too close to the corners of the urban ground areals were deleted. Trees are not normally located on corners since these areas need to clear for pedestrians to cross the streets at intersections. Figs 7 and 8 show the same tree and grate with and without wireframe view to show the grate integrated into the TIN.

The tree grate models were integrated into the urban ground areal surfaces to avoid Z-buffering issues and having the grate model cutting into the urban surfaces at odd angles. This feature integration is not computationally expensive and also supports future work in automated underground structures. Figs 9 and 10 show two types of sewer grates integrated into the TIN.

The remaining areals were marked with “parking_meter” and parking meters were initially placed every 5 meters along the edges of the urban ground areals. These models are automatically aligned so that they are perpendicular to the nearest road centerline edge, which allow for appropriate orientation on curved road segments. Next, parking meter model points that fall within one meter of street light models were pruned as well as those that fell too close to the corners of the urban ground areals. Like the removal of trees from the intersections, cars cannot park within a close distance of road intersection. All of these spatial constraints described are variables within the Tcl scripts and can be easily changed using a text editor.

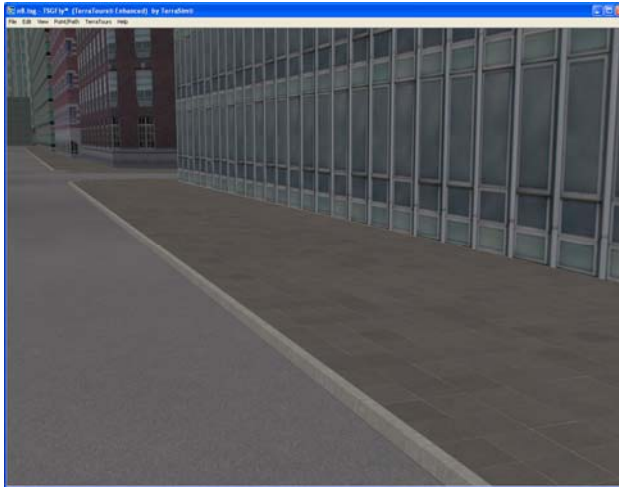


Fig 1: Street view prior to Urban Details Processing



Fig 2: Street view with bus shelter mega-model



Fig 3 Street view prior to Urban Details Processing

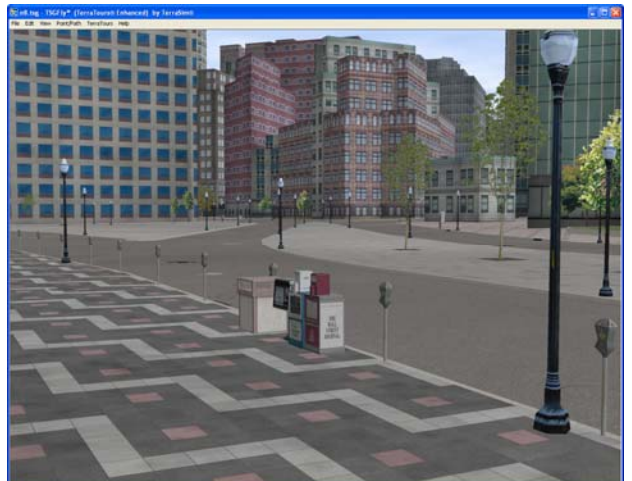


Fig 4 Sidewalk textures, and wedding cake buildings

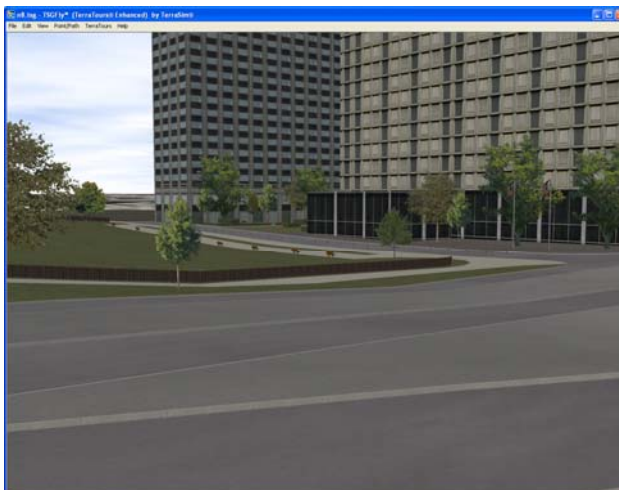


Fig 5 Street view prior to Urban Details Processing

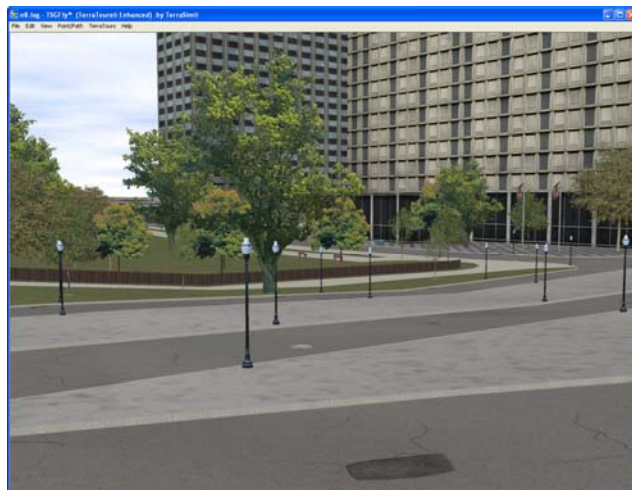


Fig 6 Improved road textures, lighting, and foliage



Fig. 7: Tree with grate model placement

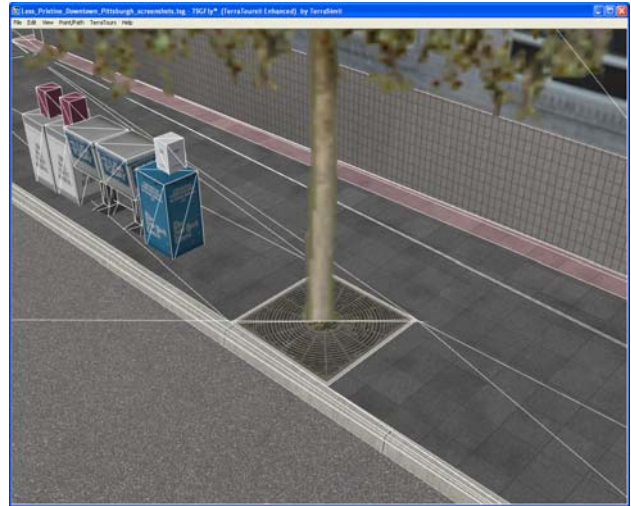


Fig. 8: Tree with grate showing terrain integration



Fig. 9: Integrated sewer grate aligned to curb



Fig. 10: Integrated sewer grate with hole in curb

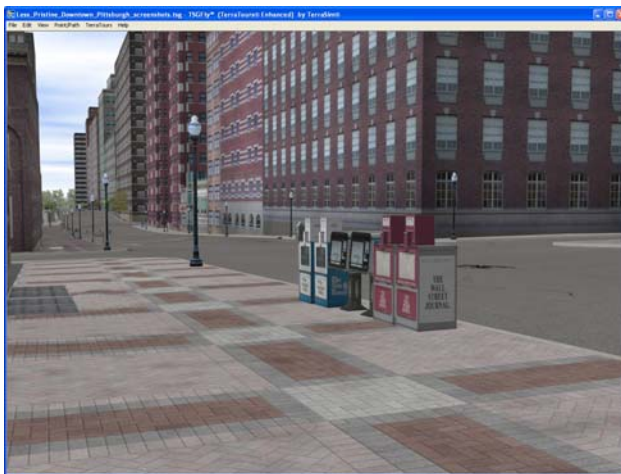


Fig. 11: Newspaper box mega-models

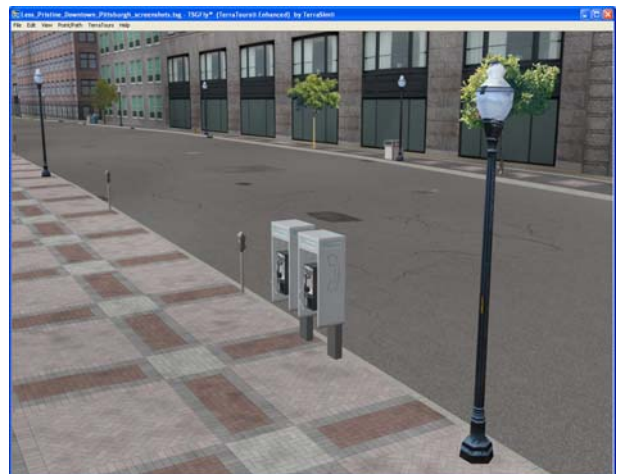


Fig. 12: Phone, parking meter, light placement

Using Mega-Models

In the previous examples we used an anchor model (lighting) to determine the sub-spacing areas for positioning other point reference models, trees and parking meters. However, there are many cases where models appear as variable collection of objects. For example, newspaper boxes are rarely positioned independently along a street. They tend to be grouped together, with close spacing closer to the intersection than in the center of the block.

Such organizations require a flexible placement strategy that scales with the number of models. The techniques used in the previous examples do not provide sufficient control to be used when grouping and proximity to other non-grouped models (newspaper boxes near light poles) must be considered. As a result we use the concept of “mega-models” can be dynamically generated from a collection of individual models, but can be placed using a single model reference point and a group bounding volume. In the sample urban database six mega-models were created: 1) A bus stop containing a bus shelter, trash can, and bus stop sign; 2) A bus stop containing a bus shelter, trash can, ash can, and bus stop sign; 3) Three newspaper boxes, side-by-side, where each newspaper box is randomly selected from a group of three different newspaper box models; 4) Three newspaper boxes and a trash can, side-by-side, where there is one each of three types of newspaper box; 5) Six newspaper boxes, side-by-side, where there are two each of three types of newspaper boxes; 6) Two payphones, side-by-side.

Model points, representing both single models and mega-models, were placed along road edges so that the points were within 25-45 meters of each other. Model points that fell within various distances of street lights, parking meters, or tree grates were deleted. Mega-models were resolved into individual model points for each component of the mega-model. Again, models points that fell within 0.3 meters of street lights, parking meters, or tree grates were deleted. All of the models were aligned to the road edges and rotations were added to models, where necessary, so the models would face the pedestrian walkways rather than the roads. Figs 11 and 12 show mega-models placed in the database. Fig 13 shows single models and mega-models placed along curved road edges.

Automated Placement of Urban Objects Within Road Surfaces

In the previous examples, with the exception of the tree grate models, all of the urban details features were placed and aligned relative to the urban terrain surface. However, there is an important class

of urban details models that require integration into the terrain surface. These include storm drains that become part of the road surface and extend into the road curb, and changes to the road surface, such as potholes and asphalt road patches, the latter being quite important for realistic modeling of southwestern Pennsylvania road surfaces.

In the urban details database, two types of storm drain models (drains with and without curb-face openings) were placed every 20 meters along road edges that were not adjacent to median areas. Models that fell too close to the road corners were removed because the integrated models need to have a curb surface at least as long as the model for proper integration. Without that constraint, model integration can often pull the curb geometry out of alignment. These models were integrated into the terrain surface to ensure the road surface and curb faces would match with the storm drain models.

Three types of 3D pothole models and one 3.5x3.5-meter 2D asphalt patch model were scattered within the road areas. Placement constraints ensured that models were not placed close to the road edges, to each other, or to overlap either the road edge or other models of this type. Models that overlapped the storm drain models, or were located very close to the storm drain models, were removed. All of these models were integrated into the terrain surface. Figs 14 and 15 show a 3D pothole integrated into the TIN.

A variety of 2D models of asphalt patches, broken asphalt, manhole covers, and smaller utility covers were scattered within the road areas. Models that fell within 1.3 meters of the pothole or the 3.5-meter asphalt patch models were deleted. Figs 16 and 17 show a large 2D asphalt patch model integrated into the TIN.

Automated Placement of Urban Objects Around Building Edges

All of the previous examples primarily used road areal geometry to control placement or integration of urban details models. Building footprints provide another focus point for the application of urban details models. In most North American cities, buildings are significantly offset from road boundaries, and these offsets usually provide pedestrian walking areas, sidewalks, mall areas, or park areas.

In the sample urban database, four types of sidewalk vents were placed individually and in groups of two, three, or four around the edges of building footprints. The models were offset from the building edges so that there was a space between the building and the vent model while models were placed 20-30 meters

apart. After initial placement, model points that fell too close to the building corners were removed to avoid creating models which do not align with the building edge. As before, mega-models were used to reduce the combinations of the placement calculation. Six mega-models were used to place two types each of two-vent, three-vent, and four-vent groups. The mega-model points were resolved into individual model points. Any individual model points that fell within 0.84 meters of road or building areals were removed. Each sidewalk vent model was automatically integrated into the terrain surface. Figs 18-20 show three types of sidewalk vent grates integrated into the sidewalk surface.

In addition to integrated models, various manhole covers and smaller sewer, water, and natural gas utility cover models were scattered within the urban ground areals as placed models. Any model points that fell within 0.3 meters of roads, buildings, models placed along the roads, and the integrated models placed around the buildings were removed. Fig 21 shows the distressed road multi-textures and placed models.

Urban Ground Visual Improvement

Up until this point we have discussed the automatic integration or placement of models within the urban environment as a way to improve urban details. However, we have also developed techniques to improve the visual appearance of urban ground, in cases where there is no specific information. As a part of the North American processing, we developed eighteen different appearances that were assigned to the urban ground polygons to give the database visual variety. Each of the textures composed of square blocks had a multi-texture applied that varied the intensity of the blocks as well as a second multi-texture that was a random pattern of blotches to give the appearance of stains, wet spots, and other dark irregular blotches on the urban surface. A multi-texture was applied to the urban ground surfaces having a cloud-like large scale intensity map which gives a general light and dark shading across the urban surfaces. Finally, the source data building footprints were used to create a buffer on the ground that was textured with a single-color block texture around the circumference of each building. Many figures show the impact of these urban ground multi-textures, some of the most striking examples are in Figs 4, 11-13, 18, and 19.

In addition to the urban ground processing we performed similar appearance improvements limited to the road surfaces. We updated the road surface to use a multi-texture feature with three textures. The smallest scale texture is the asphalt surface, the mid-range texture looks like cracks in the pavement, the

largest scale texture has light and dark patches to break up the uniformity of the road at a distance.

Automated Building Processing

We developed two urban details processing techniques to improve the appearance of building structures and roofs. In the case where source data for buildings is limited to building extent (footprints at ground level) and height, only basic rectangular solids can be generated, with some variability provided using peaked, pitched roofs, in place of a generic flat roof. A quick scan of an aerial view of a North American city would indicate a variety of architectural geometry, where buildings frequently resemble a “wedding cake” structure with the dimensions of the cross section decreasing in steps on the upper floors. Independent of this, building roofs generally have some form of clutter, including machinery rooms, access rooms, HVAC machinery, and vents.

Wedding cake buildings

For the North American database, a new technique was developed that offered some variety in the geometry of extruded footprint buildings. Normally, buildings are just the footprints extruded up to a specific height. This technique would generate buildings that are stacked like a wedding cake: the footprint is extruded up to a certain height, the outer regions of the footprint are buffered away effectively shrinking the footprint, and then the extrusion continues to the top of the building. By randomly applying this to buildings, in an urban database, a greater variety in building design is added, and a higher level of realism is achieved.

In order for this process to generate reasonable appearances, the vertical height of all the building textures must be taken into account so that windows in the textures will not be cut off. The width of the inset is randomized. After the inset is made, the geometry is checked to see if it would be realistic to extrude the top piece of the building. If the buffered shape is too long and thin, it will be rejected and the building will be extruded normally. To get a metric for the “long and thin”-ness of a building, the area is compared against the perimeter, a simple compactness measure. Figs 22 and 23 show the final building geometry on wedding cake buildings after TerraTools processing.

3D parapet generation

For middle-eastern style buildings, most roofs are living spaces with walls or parapets extending above the roofline, but generally this information is not present in the source data.

Previously, simple railing textures with no thickness were added to extruded buildings and the buildings themselves appeared as simple prismatic extrusions. This lack of a full 3D appearance was an issue for visual simulation where dismounted infantry were positioned on rooftops. As a part of our urban details processing, Tcl scripts were used to buffer the building boundary and identify those areas that would form parapets and then using a standard building extrusion process construct the parapets for that their walls and rooftop appearances are made to match the walls of the original building. This provides a fully automated way to generate parapets, even when complex building footprints with multiple levels are present.

Automatic Placement of Roof Objects on Building Rooftops

The generation of roof top clutter was accomplished using a Tcl script detailing a variety of roof object models. A TIN surface was constructed that included the building rooftops as flattened, integrated polygons. This TIN surface was used to automatically assign the correct elevations to each roof object model. A density and frequency metric in the Tcl script was used to control the number and type of models. The following process only requires the rooftop areals, which can be the original footprints or the area inside the buffered parapets or wedding cake buildings described above.

This same technique was used to place clotheslines, satellite dish, and AC units and fan models on rooftops for middle-eastern style dwellings. In the case of clotheslines the boundary of the rooftops was used to control their positions so that they ran adjacent to building edges and appeared at random from building to building. However, in the case of satellite disks, some additional consistency was required to the scattered models. Many middle-eastern rooftops have satellite dishes but simple random scattering could produce several dishes on a single, small roof. This would not be very realistic, so the Tcl script assures that per rooftop, there is at most one dish per 200 square meters of area. Further, all the dishes in the database are aligned in the same general (satellite) direction, with some slight random offset, again in an attempt to duplicate reality.

Processing AC units and fan models also requires some geometric constraint processing so that they are aligned, randomly, with one of their four sides pointing at the closest roof edge segment. Also, their placement has been implemented in such a way that whenever it is possible to place another satellite dish, it is more likely to happen than one of the other models. Figs 24 and 25 show the effects creation of roof parapets, application of multi-textured roofs with

shadowing, and the placement of roof clutter using edge alignment. Fig 24 shows a multi-level building complex extrusion with simple external wall extension. Fig 25 shows the generation of full 3D parapets which greatly improves the visual realism of the building models. The application of multi-textures to improve the appearance of generic buildings will be discussed in the following section.

Improving the Appearance of Generic Buildings

When the actual appearance of building structures is not available, one can resort to several techniques to automatically apply different external textures to break up the monotony of the environment. These textures can be selected using source data attribution, such as "building type" or "use codes" or can be derived from building size and height. For our urban details processing we provide two different techniques, rooftop texture generation and wall macro texture generation.

Previously, the rooftops of buildings (normal extrusions or extrusions with interiors) were mostly single textures. We developed a macro-texture that would be applied to the rooftops at a different scale so that the rooftops would look good whether the camera was up close or far away. The macro textures also added a "dirty" feeling to the roofs which made for a more realistic feeling. Fig 26 shows the roof textures applied for a "dirty" appearance.

The appearance of building exterior walls, both on regularly extruded buildings and buildings with interiors stood to benefit from macro-textures just as the building rooftops. However, this is not a problem that could simply be solved by adding multi-textures to preexisting features. The problem had to do with the texture coordinate generation for the multi-texture. With the rooftop multi-textures, all the texture coordinates could be generated with a projection along the world's Z axis. However, each wall of a building would require a different texture projection orientation as their normals are in different directions. Another approach would be to scale the texture coordinates used for the existing 0 channel texture. Unfortunately, that would not work when two textures are combined on one section of wall, such as with a storefront or a top-story. In those instances, the scaled macro texture would not repeat smoothly across the texture boundaries and would make for some obvious visual discontinuities.

Our solution was to make a new tool that would generate multi-texture coordinates for the buildings by evaluating the building geometry, find all the sets of contiguous, coplanar polygons, and use each of them as a single group. The building exterior process is similar to the roof macro-texture in that a Tcl script

that changes the exterior walls so that they reference the new macro-texture versions of the appearance. This results in buildings with walls that look weathered and non-repeating.

As a final improvement to building wall appearance, we added a processing step that applied a multi-texture to give the appearance of external lighting. This technique was applied to both interior and exterior building walls. The texture itself is designed to cause the corners and sides of the wall to be darkened, while there was a centered bright spot in rest of the texture. By applying this across entire planar walls, dark areas form where walls meet the ground or other walls. Although this is an extremely rough approximation of illumination, the visual effect is impressive.

It is interesting to note that the wall lighting technique is almost exactly the same as the wall macro-texture technique previously described. The only differences are that a different multi-texture is used and that the planar texture coordinate generator node is set to normalize the texture across the surface. This means that the lighting texture will be repeated exactly once across every wall, rather than repeating a number of times based on the wall's dimensions, as was done with the macro-texture. This normalization ensures that the corners and edges of the walls will have the dark areas. Figs 27 and 28 show a portion of a complete sample desert database where all of the middle-eastern building and appearance details effects have been applied. We believe that this provides a significant visual improvement over what could be generated from the simple vector source data.

Conclusions

In this paper we describe automated solutions to the problem of creating interesting urban environments, "urban details," when only the most basic vector source data is available to the database generation system. These enhancements are controlled by combinations of TerraTools® processing nodes and collections of Tcl scripts that have access to the underlying feature and terrain geometry as well as the original feature attribution.

A variety of urban details enhancements are described including urban model placement along roads, roof top clutter generation, the modification of buildings structures based upon regional properties (e.g., wedding cake and parapet generation). In addition to model placement, model integration and modification to default generated geometry, we also describe techniques to improve the appearance of the database by processing textures to give the appearance of shadowing, lighting, and dirtiness.

Fig 29 shows the original city GIS source data for the North American city example which has been shown throughout this paper. Our urban details process results in the color coded 3D position and orientation of all placed or integrated urban models, including those placed on the tops of buildings, as shown in Fig 30. This data can be exported into a variety of attributed vector formats, and used in TerraTools to create correlated high fidelity simulations such as OneSAF, JCATS, and MÄK VR-Forces.

Clearly, human modeling of urban environments using content creation tools such as 3DStudioMax, Photoshop, and Maya, currently overshadow the capabilities of automatic toolsets that convert GIS and CAD data to 3D visualizations. But given the scale and complexity for urban database requirements, containing tens of thousands of individual buildings, the use of automated urban details processing certainly provides a cost effective alternative when detailed source data is not available.

Acknowledgements

The authors would like to thank the IMAGE Society reviewers for their helpful comments on this paper.

Author Biographies

Joseph Giuliani received a BS in Computer Science with University Honors from Carnegie Mellon University in 2001, when he joined TerraSim as a computer graphics software engineer. Mr. Giuliani has been the lead graphics design engineer in many aspects of TerraTools® advanced capabilities in urban modeling. Mr. Giuliani has presented an extended tutorial on issues in urban database construction at the IMAGE Society conference in 2003 in Scottsdale, AZ.

Juliet LaDieu received a BS in Mechanical Engineering, Cum Laude, from Cornell University in 1996. In addition, Mrs. LaDieu earned a Certificate in Computer Art and Animation from the School of Communication Arts in Raleigh, NC in 2000 when she joined TerraSim. Her primary role is as Visual Database Engineer in which she tests new product software, writes and reviews documentation, supports TerraSim customers, and develops sample databases.

Dave McKeown has worked in the area of geospatial data processing to support automated mapping, remote sensing, computer vision, and visualization of geospatial environments for most of his professional career. He has lead groups in basic research and applied development as a co-founder of TerraSim, Inc, and as a Research Professor of Computer Science at Carnegie Mellon University.



Fig. 13 Complex model placement along curved road segment

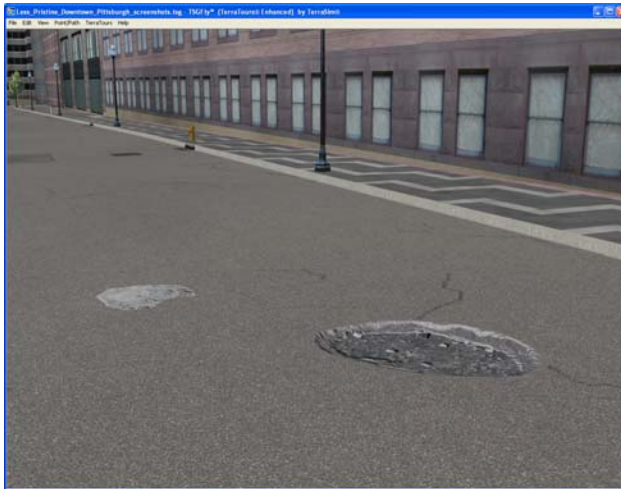


Fig. 14 Integrated 3D pothole

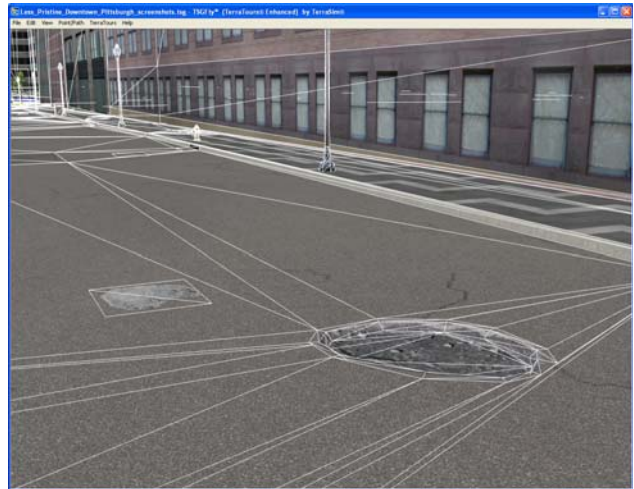


Fig. 15 Wireframe view of integrated 3D model



Fig. 16 2D model integrated into TIN

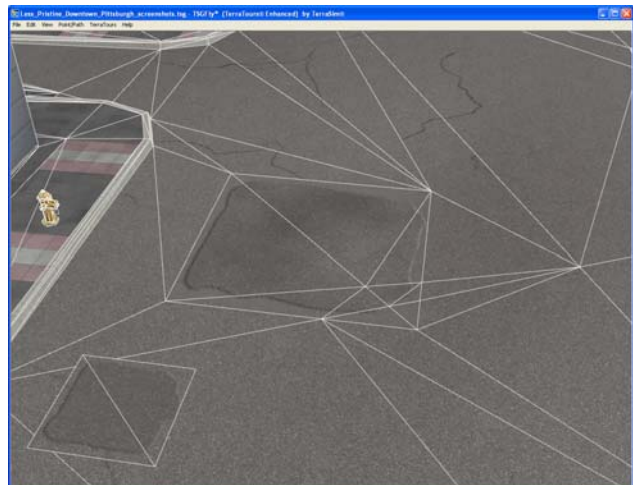


Fig. 17 Wireframe view of integrated 2D model

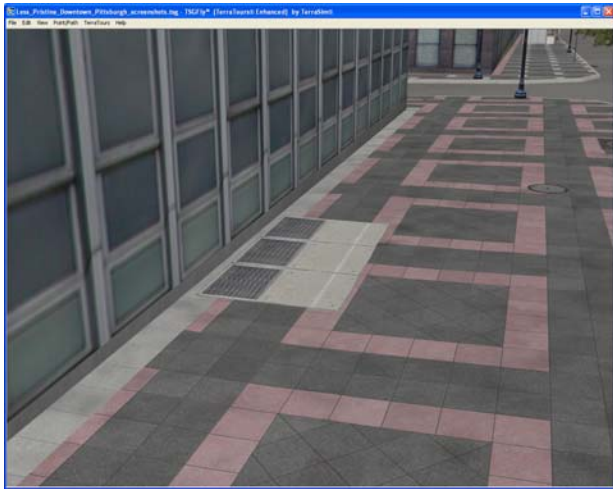


Fig. 18 Integrated sidewalk vent



Fig. 19 Integrated sidewalk vent



Fig. 20 Integrated sidewalk vents and pothole



Fig. 21 Distressed road multi-textures

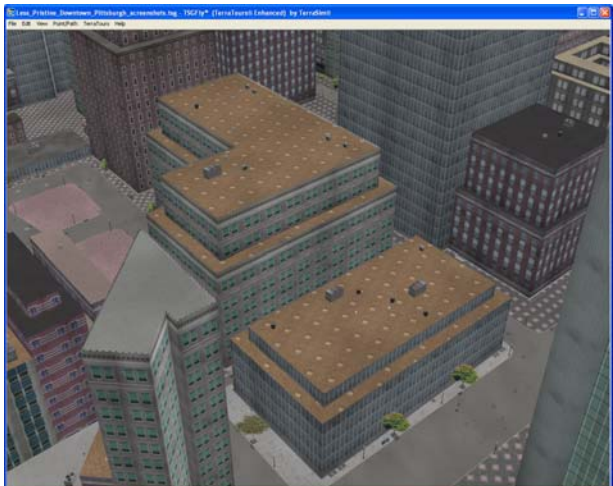


Fig. 22 North American wedding cake building

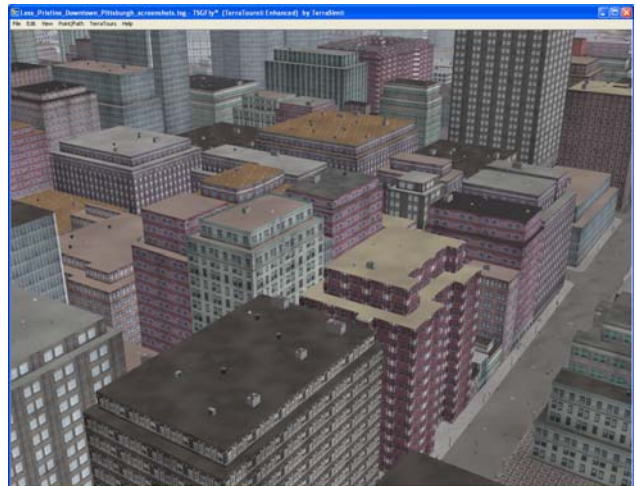


Fig. 23 More wedding cake building

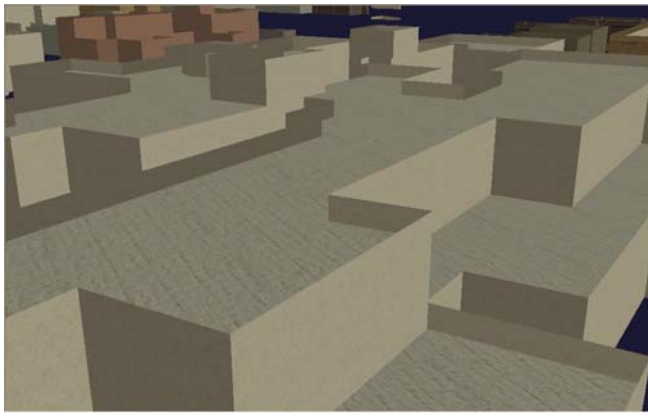


Fig. 24 Buildings before urban details



Fig. 25 After urban details, with parapets

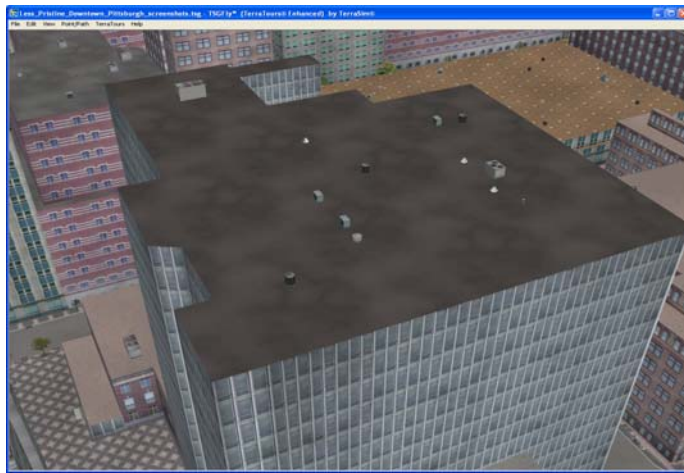


Fig. 26 "Dirty" roof appearance texture

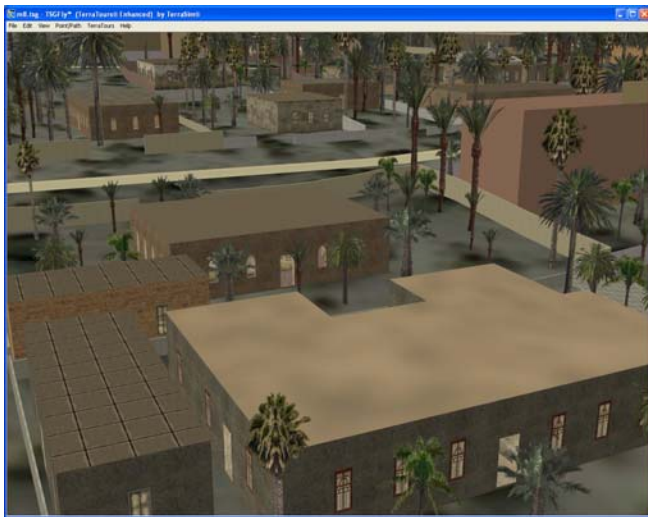


Fig. 27 Desert village before urban details



Fig. 28 After urban details, with roof clutter

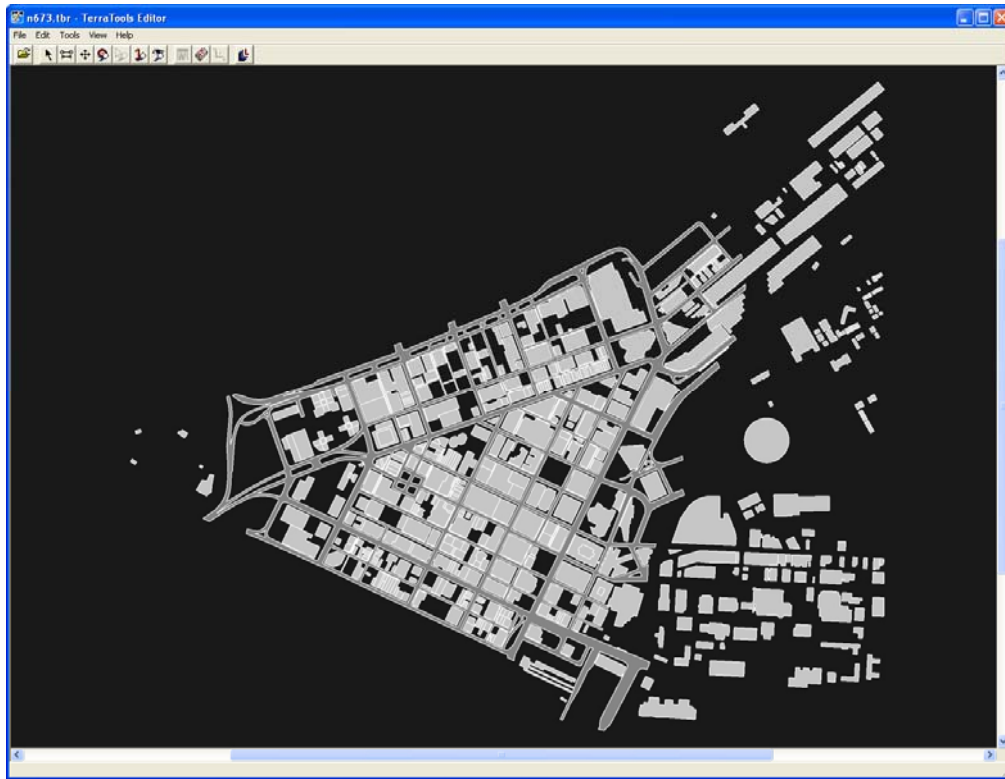


Fig. 29 GIS building and road source data

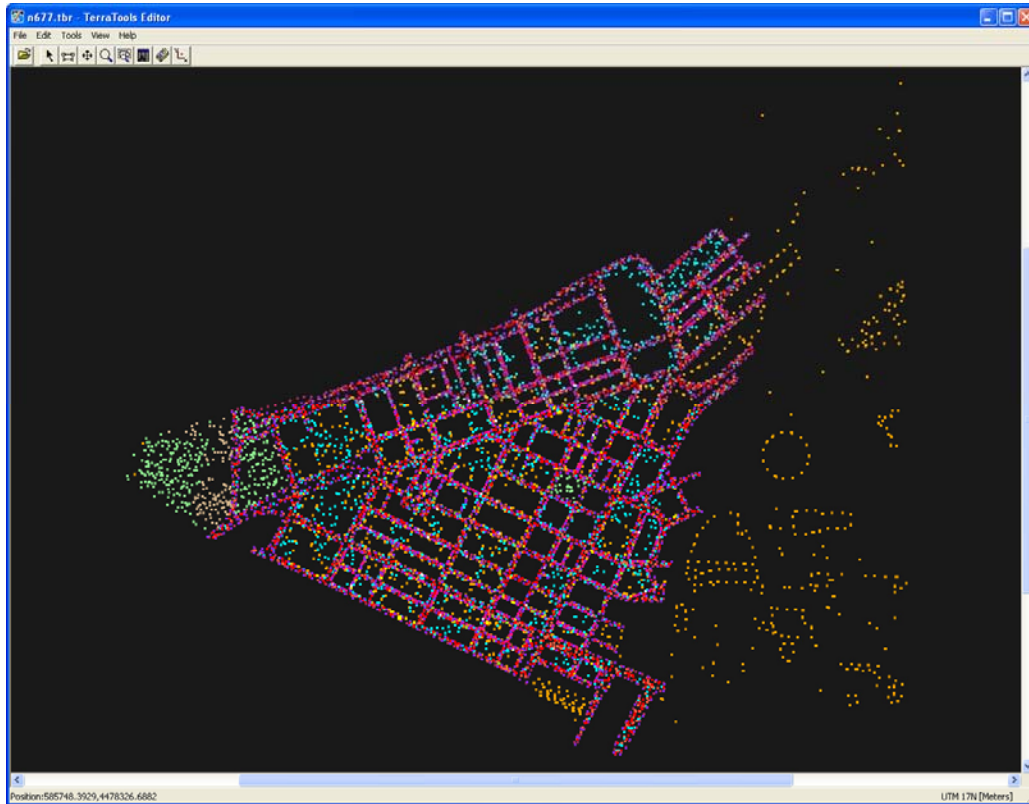


Fig. 30 Urban Details generated model placement points