AN IMPLEMENTATION OF DAMAGEABLE BUILDINGS IN A VIRTUAL ENVIRONMENT

Joe Ortiz, David Russell, Dwayne Nelson, Curtis Lisle
Institute for Simulation and Training
Orlando, Florida

ABSTRACT

As training in computer-generated environments becomes more prevalent and begins to overtake other forms as the primary method of training, more emphasis will be placed on interaction. Most simulators in use today allow limited types of interaction between participants, but virtually no interaction with the environment, which is usually a static database. In many instances interaction with the environment is not crucial to training effectiveness. A flight simulator, for example, does not require much interaction with the terrain or cultural features (buildings, roads, etc.) when the aircraft is at altitude. Other applications, however, require a dynamically changing environment since modification of the environment is a key component of the simulation. One such application is the Team Target Engagement Simulator (TTES).

The TTES is designed to train individual members of a reinforced rifle squad in the tactics of military operations in an urban environment. In the TTES, the trainee has a variety of weapons at his disposal and computing and visualizing the effects of the munitions on the buildings of the simulated environment is crucial to the training effectiveness of the TTES. We have developed and implemented a method of providing dynamic (damageable) buildings within the TTES. This paper discusses some of the requirements for simulating damageable building in the TTES, presents our system architecture and modeling procedure, and describes some of the strengths and limitations of our approach.

ABOUT THE AUTHORS

Joe Ortiz is a Visual Systems Scientist at the Institute for Simulation and Training and is the Principal Investigator of dynamic structures for the Team Target Engagement Simulator project. He received a B.S. degree in Aeronautical Engineering and a M.S. degree in Mechanical Engineering from Purdue University in 1983 and 1993, respectively.

David Russell is a Graduate Research Assistant at the Institute for Simulation and Training. He is currently working with dynamic structures for the Team Target Engagement Simulator project and was involved in the development of the Mirage display. He received his B.S. degree in Electrical Engineering from Vanderbilt University in 1994 and is currently working towards his M.S. degree in Electrical Engineering at the University of Central Florida.

Dwayne Nelson is a Graduate Research Assistant at the Institute for Simulation and Training. He received a B.S. degree in Computer Science from Oakwood College (Huntsville, AL) in 1994 and is currently in pursuit of a Ph.D. in Computer Science from the University of Central Florida.

Curtis Lisle is currently on leave from the Institute for Simulation and Training to pursue a Ph.D. in Computer Science at the University of Central Florida. He led the Dynamic Terrain Research and Development Testbed project and was also involved in Visual System Database Research and Development. He received a B.S. degree in Electrical Engineering from the Georgia Institute of Technology and a M.S. degree in Computer Science from the University of Central Florida in 1986 and 1991, respectively.

AN IMPLEMENTATION OF DAMAGEABLE BUILDINGS IN A VIRTUAL ENVIRONMENT

Joe Ortiz, David Russell, Dwayne Nelson, Curtis Lisle
Institute for Simulation and Training
Orlando, Florida

INTRODUCTION

The Team Target Engagement Simulator (TTES) being developed by the Naval Air Warfare Center Training Systems Division (NAWC-TSD) for the United States Marine Corps is designed to train members of a deployed rifle squad in the tactics of military operations on urbanized terrain [Horey, Fowlkes and Reif, 1995]. In the TTES, one trainee can be opposed by multiple computer-controlled hostiles, or by networking several simulators, an entire squad of trainees can participate in force-on-force engagements. While in the simulator, each trainee is able to move about freely within the synthetic environment and practice threat identification, cover, communication, and engagement skills.

Currently the only weapon available to the trainee in the TTES is the M16-A2, but future modifications will give the trainee access to explosive munitions such as grenade launchers and satchel charges. In many instances, these munitions will be directed against the structures within the simulated environment in order to breach a wall and gain entry into a room or building. This makes the ability to compute and visualize the effects of the munitions on the buildings of the simulated environment crucial to the training effectiveness of the TTES.

The Institute for Simulation and Training (IST) is currently developing dynamic structures which will be integrated with the TTES. These structures will permit a trainee to use explosive munitions in a completely arbitrary manner and the appropriate effects on the buildings will be computed and displayed. For instance, if a satchel charge is used, a hole will appear in the building exactly where the munition was detonated, and the hole will be appropriately sized for the weight of the munition and the wall properties. In addition, it will be a true hole, that is, the trainee will be able to see through it, walk through it, and fire a weapon through it. If the charge is not sufficient to breach the wall, a crater will appear instead, and the wall will be "weakened" in the area of the crater.

Although the development of our system is not complete, the architecture and object hierarchy presented herein have been implemented to an extent that a working prototype has been integrated with the TTES. A modeling scheme (not discussed in this paper) has also been developed that allows databases to be constructed using MultiGen modeling software (albeit using specialized modeling rules).

MODELING REQUIREMENTS

Three main requirements drove the design of our system. First, it must be capable of responding quickly enough for an interactive, man-in-the-loop simulation. Sluggish response in updating the visual database after a munition detonation can result in a loss of confidence in the simulator and adverse training. When explosive munitions are used in live fire situations, the target is usually obscured by dust and smoke for a few seconds after detonation. Assuming that special visual effects within the TTES will similarly obscure the synthetic target, a time of approximately two seconds was used as a goal for our system to update the visual display after receipt of a detonation signal.

Secondly, the system should be organized and structured in such a way as to facilitate the use of relatively simple munition algorithms for calculating blast effects. With the constraints on response time mentioned above, the ability to quickly compute the damage from a munition denotation becomes extremely important. Early in the development process, discussions were held with representatives from the U.S. Army Corps of Engineers Waterways Experiment Station (WES) who are responsible for developing the munition algorithms. As a result of these discussions it became clear that theoretical computations on arbitrarily shaped objects could not be performed within the time constraints. Analysis of relatively simple objects using algorithms based on empirical results and table look-up routines were most appropriate for this application.

Lastly, the system must be capable of modeling a wide variety of buildings and of displaying arbitrary damage on any of the buildings. The urban environment that is used in the development of the TTES is the Quantico Combat Training Village (QCTV) consisting of 16 buildings. Many of the buildings are multi-storied and contain structural components such as stairwells, elevator shafts, balconies, fire escapes, and

crawl spaces. Our system should be flexible enough to model each of the buildings and to simulate damage due to an explosive charge detonated anywhere within the village.

GENERAL SYSTEM DESCRIPTION

The general layout for implementing damageable buildings within the TTES is shown in Figure 1. Our system, called PSIM for physical simulator, runs as a separate process and receives detonation signals and munition data from the TTES. PSIM contains models for each of the buildings within the TTES and a library of munition algorithms which compute blast damage. PSIM also has access to those portions of the TTES visual database which are used to render the buildings. Upon receipt of a detonation signal, PSIM calculates the blast damage for each building in the simulation, updates its internal models to reflect the new damage, and then modifies the visual database.

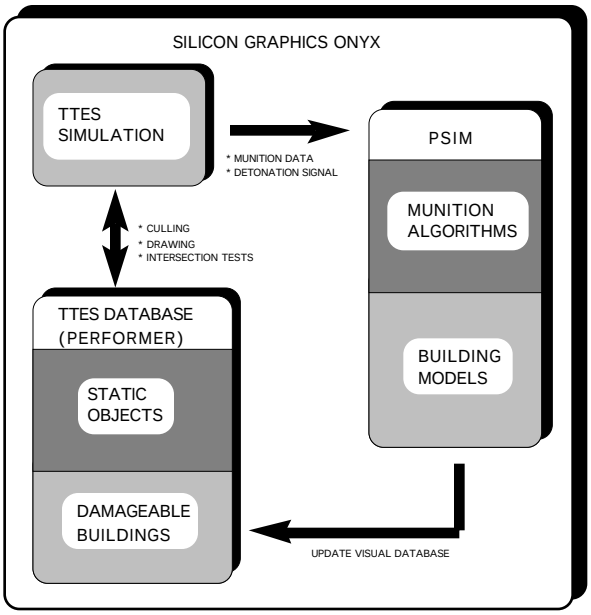


Figure 1 : TTES/PSIM System Architecture

Within PSIM, each building is a separate software object composed of an analytical and a visual model. The analytical model implements an objected-oriented, hierarchical description of building geometry and materials and uses software components that represent real-world concepts such as rooms and walls. The hierarchical representation provides a generic solution for modeling any building and breaks up buildings into smaller components which can be analyzed efficiently by the munition algorithms.

Each analytical model computes its own blast damage by accessing the appropriate munition algorithms. The algorithms are being developed by WES specifically for the TTES and are designed for real-time simulations. They receive data describing both the munition and the geometry/material properties of the building (or building component) and return information detailing the amount of damage incurred. The analytical model then uses the damage information and modifies its geometry accordingly.

The visual model is a collection of texture mapped polygons that can be rendered by an image generator. It is contained within the TTES database, but is created and modified by the building object located in PSIM. Whenever a building is damaged, the newly damaged components are updated in the visual model to reflect the current state of the analytical model.

ANALYTICAL MODEL

The analytical model is used to store building geometry, material properties, and connectivity information in an easily accessible, logical manner so that damage computations can be performed quickly and efficiently. Damage computations consist of determining the effect of a munition on a particular component (e.g. whether or not it was breached, the size and depth of the resulting hole), modifying the component geometry to accurately reflect the damage, and identifying and removing any unsupported (floating) components from the simulation.

Each building has its own analytical model composed of three types of components - rooms, slabs, and chunks. The components are stored in an hierarchical organization (see Figure 2) so that all buildings are structured and analyzed in the same manner regardless of the differences in geometry and/or material properties. Munition effects for an entire building are calculated using algorithms that analyze only one or two of the component types.

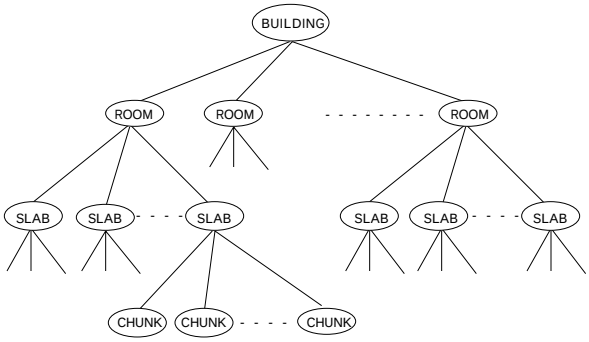


Figure 2 : Analytical Model Object Hierarchy

Chunk Class

At the lowest level of the hierarchy is the chunk class. In the simplest terms, a chunk is a single, contiguous piece of the building that has homogenous properties throughout. The shape of a chunk is defined in two dimensions by a shape polygon located on a centerline plane (see Figure 3). The shape polygon is, in general, a concave polygon and can contain an unlimited number of holes. The overall three-dimensional geometry of the chunk is determined by extruding the shape polygon along the line of its normal vector. Functions within the chunk class allow it to modify its geometry as a result of damage and to communicate to other objects the amount of damage it received.

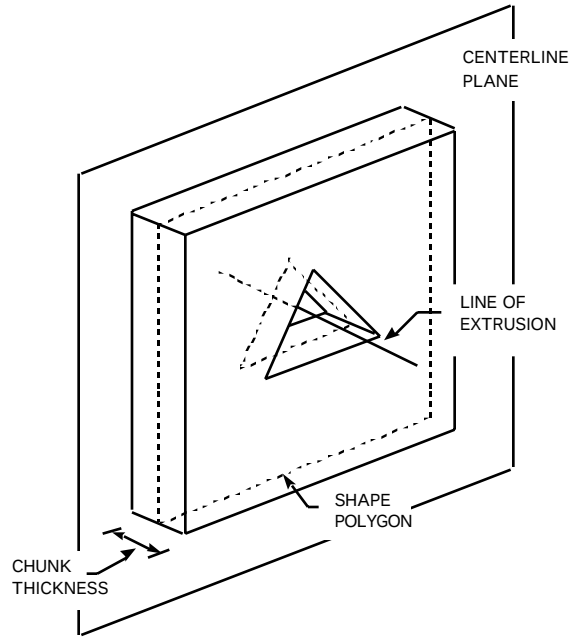


Figure 3 : Chunk Definition

Slab Class

Slabs are the heart of the modeling system and are used to represent almost all of the basic components of a building (i.e. walls, floors/ceilings, railings, stairs). A slab is essentially a collection of one or more chunks that are composed of the same material and are defined relative to the same centerline plane (see Figure 4). Chunks can touch along edges, but cannot overlap since this could result in interpenetrating chunks of the same slab (interpenetrating chunks from different slabs are allowed).

Each slab instance is defined in its own coordinate system. The shape polygons of each of the chunks are defined in the x-y plane of the slab and are extruded along the z-axis. An offset distance (relative to the slab centerline plane) allows chunks to be staggered in the z-direction. Generally, each slab is instantiated using a single chunk and additional chunks are generated as required as a result of damage. Each slab interfaces directly with the munition algorithms so that it can compute its own blast damage

Room Class

A room is a collection of slabs that encloses a volume within a building. Rooms provide an intuitive way to partition a building so that internal blast calculations can be performed.

Each room contains a list of the interior and exterior slabs that compose it. An interior slab is one that is completely contained within the room boundaries, while an exterior slab is one that forms part of the room boundary. Exterior slabs can be associated with more than one room since adjacent rooms may share one or more slabs. A room can communicate directly with the building and also with each of its slabs.

Upon instantiation, a room obtains position and geometric information from each slab and calculates its volume and vent areas (A vent area is an opening in the boundary of the room, such as an open window or doorway, that allows air to flow in or out of the room). These properties will be used in future versions of PSIM to calculate internal blast effects. If as a result of a munition detonation any of the exterior slabs of a room are damaged, the vent areas are updated to reflect the changes.

Connectivity Information

Connectivity information for each component is computed at the beginning of the simulation and is updated as needed as a result of munition damage. Each component stores its own connectivity information which consists of a list of similar components with which it is in physical contact. For instance, each chunk object has a list of all the other chunks that it touches - likewise for each slab and room object. At the present time, connectivity information is used only to determine which components are unsupported and should be removed from the simulation. Future modifications will also use this information to determine the propagation of internal blast effects.

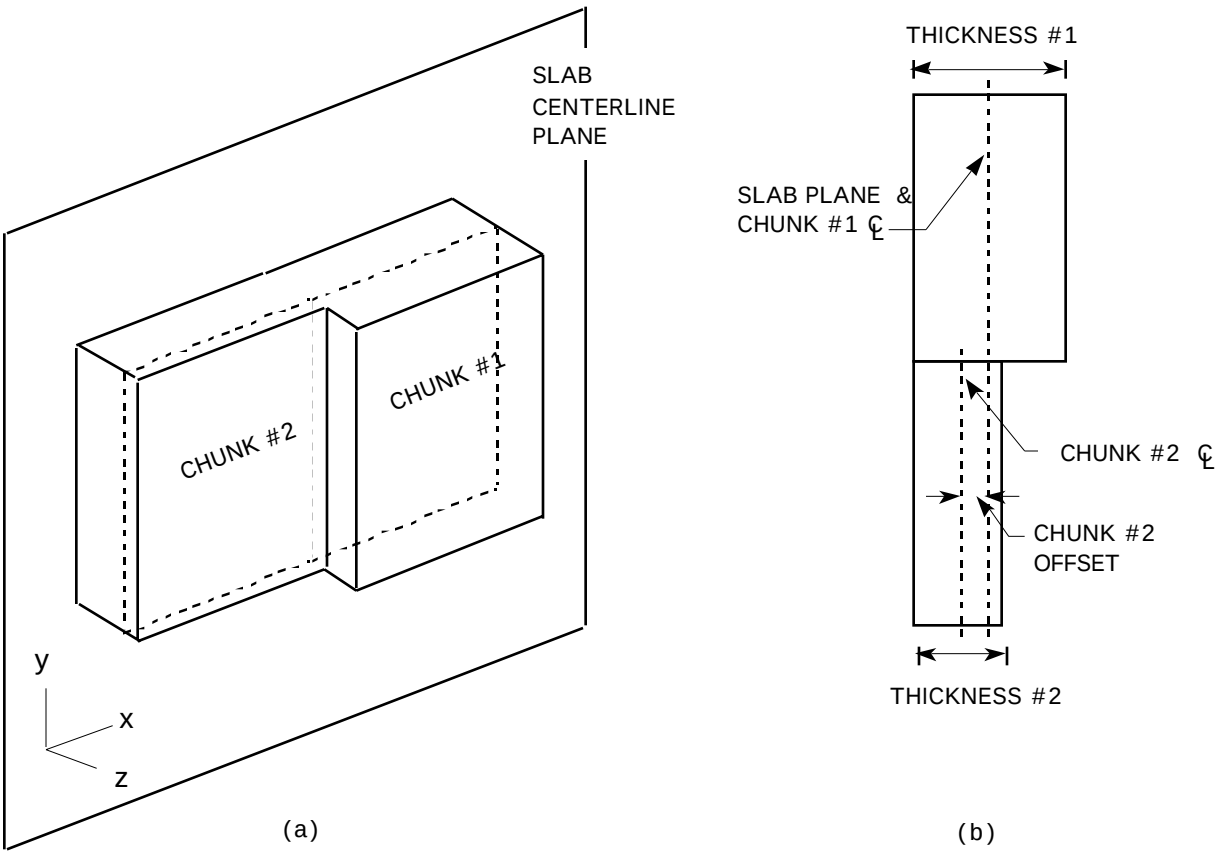


Figure 4 : Slab Definition (a) Orthographic View, (b) Overhead View

VISUAL MODEL

The visual model is the portion of our system with which the trainee interacts and is also the only part that is directly accessible by the TTES. Since it is created by accessing the geometric data of the analytical model and creating texture mapped polygons that represent each of the building components, the visual model also has the same hierarchical structure as the analytical model.

The visual model is de-coupled from the analytical model so that it can be tailored to the specific graphics requirements of the simulation. For instance, in the TTES the visual models are created for inclusion in a Silicon Graphics Performer database. A different simulator may require the use of a different graphics application programming interface (API), but could still use the same underlying analytical model and munition algorithms since they are not tied to a specific graphics system. The object-oriented approach makes the separation of models transparent to PSIM; it sees only one object per building which incorporates both the analytical and visual models.

The visual model is composed of polygons (in most cases, triangular) that can be rendered by an image generator. Each chunk can be thought of as being composed of face and edge polygons (see Figure 5). Face polygons are those that lie on either of the two planes perpendicular to the line of extrusion and are created by triangulating the shape polygon [Brassel and Fegeas, 1979; Lee, 1981; Fournier and Montuno, 1984]. Edge polygons form the sides of the chunk and the interior edges of any holes and are created as the edges of the shape polygon are extruded from the front to the back face of the chunk. Once all of the polygons are created, a texture pattern that is appropriate for the particular material (e.g.. brick, cinder block) is applied to them.

The visual model for each building is attached to the TTES database at the beginning of the simulation. Once a model is attached to the database, it is accessed by the TTES in the same manner as any other element for culling, drawing, intersection testing, or any other function. The only difference between the visual model of a building and any other element of the database is that a building may be modified during the simulation as a result of a munition detonation.

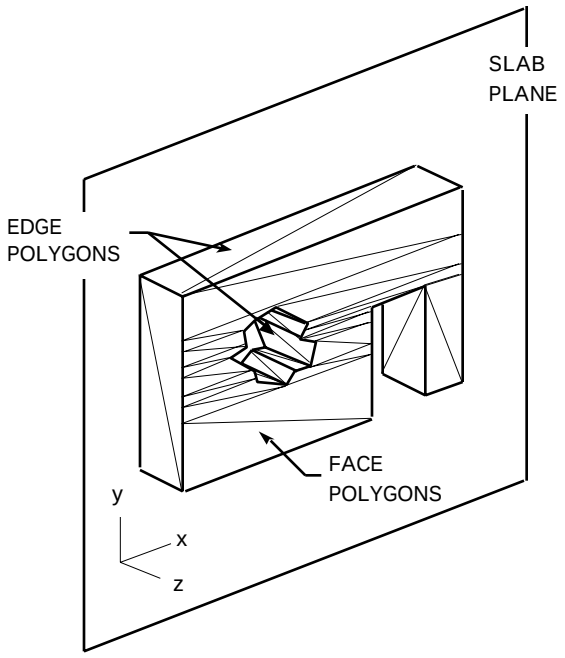


Figure 5 : Creating Renderable Polygons for a Chunk With a Breach

Modification of a visual model is done automatically after a munition detonation (if necessary) and does not require any special action by the TTES. After the analytical model has been updated, the building determines which components have been recently damaged and modifies their portions of the visual database to accurately reflect the current state of the analytical model. The new damage will be displayed by the TTES following the next traversal of the database.

CALCULATION OF MUNITION EFFECTS

Calculation of munition effects begins with the receipt of a detonation signal from the TTES. Munition data (charge weight, velocity, location, etc.) is distributed to each building which determines the effect of the munition on itself by accessing the appropriate munition algorithm(s). Each newly damaged building component is identified and the geometry and connectivity information in the analysis model is modified to reflect the damage. The visual model (and consequently, the TTES database) is then updated by rebuilding the affected components.

At present, only one munition algorithm has been integrated in PSIM. It is a single contact breaching algorithm and is used to determine the effect of a munition detonated on the surface of a wall. The algorithm, which was developed by WES, assumes a spherical charge and uses a table look-up routine to calculate blast effects. Damage is based on the weight of the explosive (specified as an equivalent weight of C4) and the geometric/material properties of the wall. Additional algorithms under development will calculate effects due to small arms fire, air blasts, pattern charges, and fragmentation devices.

Damage is determined by descending through the building hierarchy and analyzing each component separately. Since only the single contact breach algorithm is currently available, the room class is bypassed and the damage calculations are performed only at the slab level (When additional munition algorithms become available, room partitions will be used to determine internal blast pressures, propagation of blast effects throughout the building, and shielding due to internal walls). Each slab accesses the munition algorithm and provides its own unique geometry data and material properties. The munition algorithm determines whether the charge was sufficient to breach the slab, and if so, returns the radius of the resulting hole. If the charge was insufficient to breach the slab, but capable of producing a crater, the radius and depth of the resulting crater are returned.

Completely Breached Slabs

Once the damage to the slab has been determined, the analytical model is modified to reflect the results. In the case of a complete breach, this means that a hole must be "cut out" of the slab. Since our modeling process extrudes a two-dimensional shape to create a three-dimensional object, creating a breach simply involves modifying the shape polygons. A hole polygon that represents the damaged area is created and is used by each chunk as a pattern to modify its own geometry. The damaged area is removed by performing Boolean operations [Weiler and Atherton, 1977; Weiler, 1980] between the hole and shape polygons.

Three possibilities can result from the Boolean operations: 1) the chunk is undamaged, 2) the chunk is partially damaged, or 3) the chunk is completely destroyed,. The first case occurs when the hole and shape polygons do not intersect. This can occur frequently with slabs composed of multiple chunks and signifies that the chunk was too far from the explosion to have been affected. No further processing of this particular chunk is required. The third possibility occurs when the shape polygon is completely contained within the hole polygon. This signifies that the chunk has been destroyed by the explosion and is usually the result of a munition being detonated directly on top of

a small chunk. In this situation, the chunk is removed from the object hierarchy and is no longer a part of the simulation.

The second case is the most general and the most commonly occurring. The shape and hole polygons intersect, signifying that the munition damaged the chunk, but was not powerful enough to completely destroy it. The area of intersection between the shape and hole polygons represents the damaged area. The Boolean operation removes the damaged area from the chunk and the modified shape polygon represents the new chunk geometry. It is possible that as a result of the Boolean operation, the shape polygon is actually split into two or more non-intersecting polygons. In this situation the original chunk is assigned one of the resultant polygons as its new shape polygon and a new chunk is created for each of the remaining polygons. The new chunks are then incorporated into the object hierarchy.

Craters

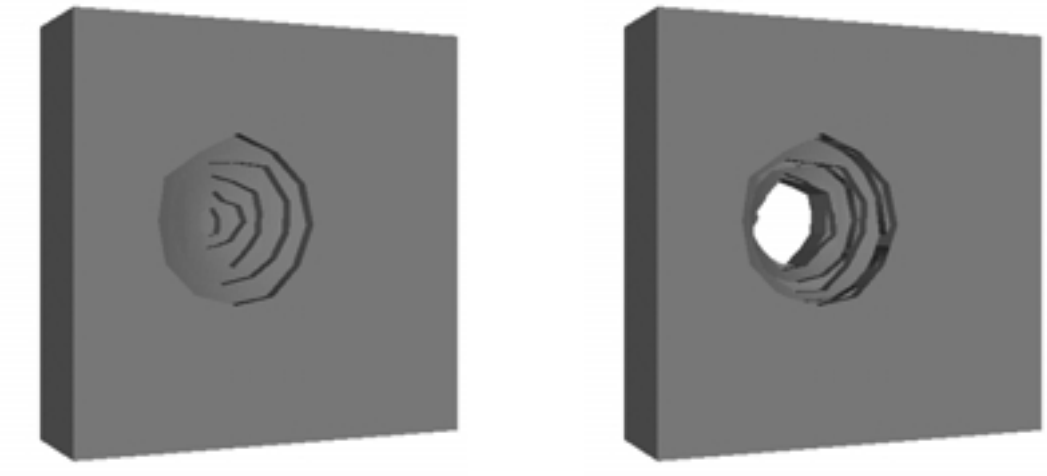
There are instances when a munition, although not powerful enough to completely breach a wall, is still capable of causing damage. In these cases, a crater is generated and the slab geometry modified to account for the removal of material. Since our modeling process extrudes a two-dimensional shape, it is not possible for a slab composed of a single chunk to contain a feature such as a tapered hole or crater.

Formation of a crater requires that additional chunks be generated to simulate the tapering effect. The process begins by removing the area that will contain the crater. A hole polygon whose shape matches the maximum extents of the crater boundaries is used in the Boolean process to produce a hole in the slab that can be filled by new chunks. Each of the new chunks, except the last, has an annular shape and together they form concentric rings which grow progressively smaller and thinner as they approach the center of the crater. The last (innermost) chunk has a circular shape and forms the bottom of the crater.

The resulting crater (see Figure 6a) has a stair-stepped appearance where the thickness of each chunk decreases the closer it is to the crater center. The new chunks have the same material properties as the original, but since their thicknesses are reduced, they are less able to withstand additional blasts. If another munition of the same weight as the original were detonated in the center of a crater, it may be possible for it to breach the slab (see Figure 6b). This simulates a "weakening" of the slab in the area of the crater.

Connectivity and Removal of Floating Components

As a result of a munition detonation, components of the building that were previously in contact with each other, may now be separated. Since each component keeps a list of similar components with which it is in physical contact, these lists must be updated to reflect the current building configuration. This is done by identifying each recently damaged component and then confirming the accuracy of the lists of all components near these localized areas of damage.



(a) (b)

Figure 6 : (a) Crater Geometry Has Stair-Stepped Appearance, (b) Second Detonation Inside Crater May Breach Slab

Once the connectivity lists have been updated, any unsupported components are identified and removed from the simulation. Components are assumed to be supported if they are either directly or indirectly in contact with the ground. That is, if a component touches the ground or if there is some unbroken path consisting of other components that leads to the ground, the component is supported. If the component is not directly or indirectly in contact with the ground, then the component must be floating in mid-air and is removed from the object hierarchy.

Modification of Visual Model

Once the analytical model has been updated (i.e. shape polygons modified, connectivity lists updated, and floating components removed from the hierarchy), the building modifies each section of the visual model that corresponds to a recently damaged component Each chunk has a specific location where its renderable polygons reside. If a chunk has been recently damaged, rather than trying to determine which polygons should be kept and which discarded, everything within its section of the visual model is rebuilt using up-to-date information from the analytical model. Those portions of the visual model that correspond to a component that was removed from the analytical model hierarchy are deleted, while additional sections are added for any newly created components.

CONCLUSIONS

Our modeling approach has been to use a few basic components and organize them into a hierarchical structure to create relatively complex buildings (see Figure 7). This is beneficial in several ways. First, relatively simple algorithms can be used in the analysis of munition damage. Since damage computations are conducted on individual components at either the slab or the room level, the munition algorithms can be tailored to analyzing only these types of components.

Secondly, the hierarchical organization assists in developing a general methodology for analyzing all buildings. Any building, no matter how complex the layout (multi-storied, internal stairwells, hallways, rooms, etc.), is analyzed by the same methodical and logical manner of descending through the object hierarchy and examining individual components. In addition, the organization allows quick rejection tests to be utilized at each level of the hierarchy so that time is not wasted examining unaffected components.

The Boolean operations show great flexibility in representing damage. It is unnecessary to know in advance how a munition will affect a slab (e.g. will it create a hole in the middle of the slab or only damage the edges). The procedure is also indifferent as to whether the slab has damage from a previous munition. Previous damage is embedded in the geometry of the shape polygon so two or more distinctly different

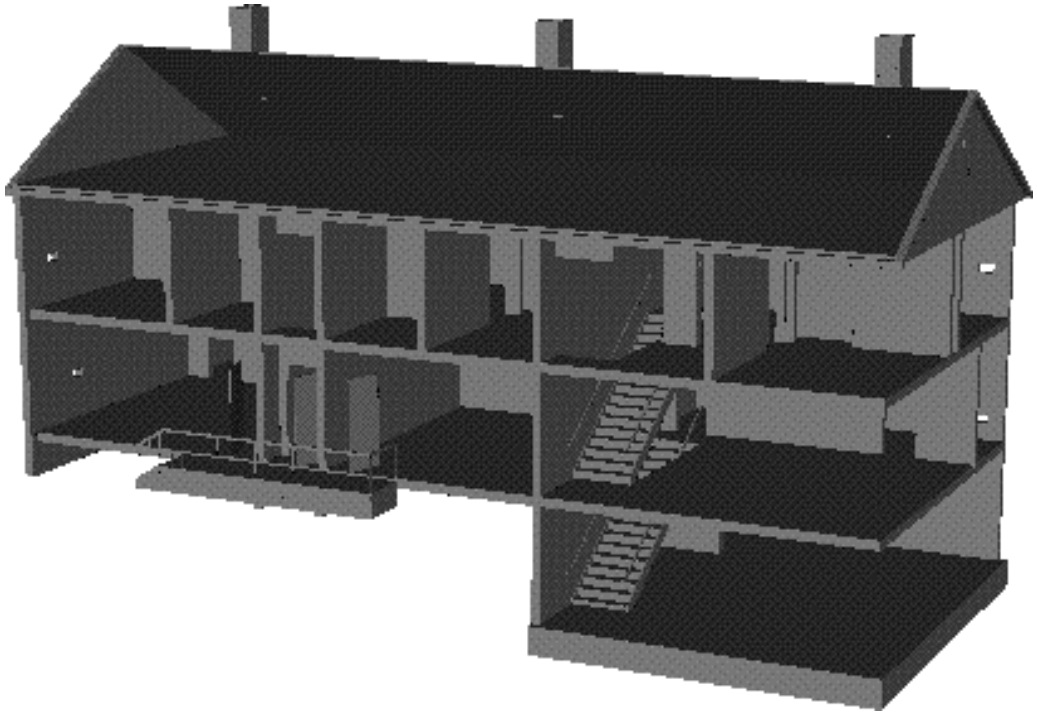
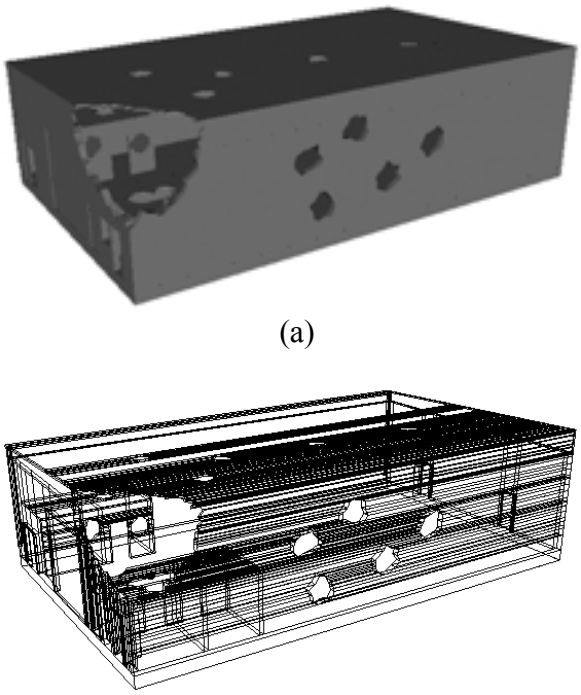


Figure 7 : Interior View of a Multi-Level Building Showing Internal Walls and Stairways

detonations can produce a single area of damage. In addition, although we currently use a circular hole to represent a breach, any two-dimensional shape may be used to represent the damaged area. Pattern charges, which are designed to cut a particular shape from a wall, can be simulated simply by supplying the appropriately shaped hole polygon.

While our extrusion methodology simplifies geometry calculations, it also limits the types of objects that can be modeled. In particular, components that are tapered or have beveled edges cannot be modeled as a single object, but must be composed of several objects to simulate the tapering effect. Depending on the particular approach taken, this may result in a blocky or stair-stepped appearance. A solid modeling package would have permitted greater flexibility in the geometric shapes of both objects and damage, but at increased developmental and computational expense.

Another area of concern is the large number of polygons that are generated in the visual database. Approximately seventy new polygons are added to the database for each hole and three hundred and fifty for each crater. If only one or two munitions are used during the simulation, the impact is trivial, but if a large number of munitions are deployed, the size of the database can increase significantly and adversely affect the performance of the TTES.



(a)

(b)

Figure 8 : Severely Damaged Building (a) Shaded View, (b) Wireframe View Showing Polygonization

Figure 8 illustrates the problem. The undamaged building was modeled using a total of 328 trapezoidal (656 triangular) polygons, while the damaged building (which does not contain any craters) requires over 1100 trapezoidal (2200 triangular) polygons. Although this amount of damage is an extreme case and would not normally occur within the TTES, it demonstrates how rapidly polygon counts can rise as a result of multiple explosions.

FUTURE WORK

The development of damageable buildings within the TTES is an ongoing, evolutionary process. The integration of additional munition algorithms is continuing, as is the methodology to support the propagation of internal blast effects. Future enhancements could also include improved spatial partitioning and automatic generation of low level-of-detail (LOD) visual models, both of which would improve simulator performance.

REFERENCES

Horey, J. D., Fowlkes, D. H., Reif, R. S., "Military Operations on Urbanized Terrain (MOUT) Training in Synthetic Environments using the Team Target Engagement Simulator (TTES)", 17th Interservice/ Industry Training Systems and Education Conference Proceedings, December 1995.

Brassel, K. E., Fegeas, R., "An Algorithm for Shading of Regions on Vector Display Devices", Computer Graphics - SIGGRAPH 79 Proceedings, Summer 1979.

Lee, D. T., "Shading of Regions on Vector Display Devises", Computer Graphics - SIGGRAPH 81 Proceedings, Summer 1981.

Fournier, A., Montuno, D. Y., "Triangulating Simple Polygons and Equivalent Problems", ACM Transactions on Graphics, Vol. 3, No. 2., April 1984.

Weiler, K., Atherton, P., "Hidden Surface Removal Using Polygon Area Sorting", Computer Graphics - SIGGRAPH 77 Proceedings, Summer 1977.

Weiler, K., "Polygon Comparison using a Graph Representation", Computer Graphics - SIGGRAPH 80 Proceedings, Summer 1980.

ACKNOWLEDGMENTS

The authors would like to thank the Naval Air Warfare Center Training Systems Division and the United States Marine Corps for sponsoring this work (contract N61339-95-K-0019).