Producing High-Quality Visualizations of Large-Scale Simulations

Voicu Popescu, Chris Hoffmann, CS
Sami Kilic, Mete Sozen, CE
Scott Meador, ITaP
Purdue University

Abstract

This paper describes the work of a team of researchers in computer graphics, geometric computing, and civil engineering to produce a visualization of the September 2001 attack on the Pentagon. The immediate motivation for the project was to understand the behavior of the building under the impact. The longer term motivation was to establish a path for producing high-quality visualizations of large scale simulations.

The first challenge was managing the enormous complexity of the scene to fit within the limits of state-of-the art simulation software systems and supercomputing resources. The second challenge was to integrate the simulation results into a high-quality visualization. To meet this challenge, we implemented a custom importer that simplifies and loads the massive simulation data in a commercial animation system. The surrounding scene is modeled using image-based techniques and is also imported in the animation system where the visualization is produced.

A specific issue for us was to federate the simulation and the animation systems, both commercial systems not under our control and following internally different conceptualizations of geometry and animation. This had to be done such that scalability was achieved. The reusable link created between the two systems allows communicating the results to non-specialists and the public at large, as well as facilitating communication in teams with members having diverse technical backgrounds.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Applications, Three-dimensional Graphics and Realism, Graphics Utilities. I.6. [Simulation and Modeling]: Applications, Model Validation and Analysis, Output Analysis.

1. INTRODUCTION

1.1 Problem description

Since Ken Wilson's articulation of simulation as third paradigm of science in the mid-1980s [14], co-equal with experimental and theoretical science, simulations have become essential tools in many fields of science and engineering. Scientific simulations are used to crash-test an automobile before it is built, to study the interaction between a hip implant and the femur, to evaluate and renovate medieval bridges, to assess the effectiveness of electronic circuit packaging by running circuit-board drop tests, or to build virtual wind tunnels.

Computer Science Department, Purdue University
250 N University Street
West-Lafayette, IN, 47907-2066
{popescu | cmh}@cs.purdue.edu, {skilic | sozen}@purdue.edu, wsmeador@tech.purdue.edu

In particular, finite-element analysis (FEA) plays a fundamental role in engineering because of its ability to integrate multiple physical phenomena, such as fluid flow, fluid/solid interaction, and material behavior. FEA systems compute a variety of physical parameters over the time span of the simulation, such as position, velocity, acceleration, stress, and pressure. The visual presentation of the results is either handed off to generic post-processors or else is studied in specific contexts in the field of scientific visualization.

Three dimensional computer graphics has advanced tremendously, driven mostly by the popularity of its applications in entertainment. Consumer-level priced personal computers with add-in graphics cards can produce high-quality images of complex 3D scenes at interactive rates or can run sophisticated animation software systems to produce, off-line, video sequences that very closely approach photorealism. Because of the specifics of the applications that commissioned their development, animation systems are mainly concerned with minimizing the production effort and maximizing the entertainment value of animations. They focus on the rendering quality, on the expressivity of the animated characters and are less concerned with closely following the laws of physics.

Our team had the goal of producing a visualization of the September 2001 attack on the Pentagon that is both physically and visually accurate (Figure 1, Figure 3 and accompanying video). The obvious solution is to take advantage of the strengths of both simulation *and* animation systems. The project had two distinct parts. During the first part we designed, tested and then ran at full scale the FEA simulation of the aircraft impacting the building structure. For this part we used LS-DYNA [5], a commercial FEA system often used for crashworthiness simulations. In the second phase the efforts were focused on producing a high-quality visualization of the massive data resulting from the simulation. In order to do so we created a scalable link between the FEA system and a commercial animation system (3ds max [19]). The link can be directly reused to create animations with physical fidelity

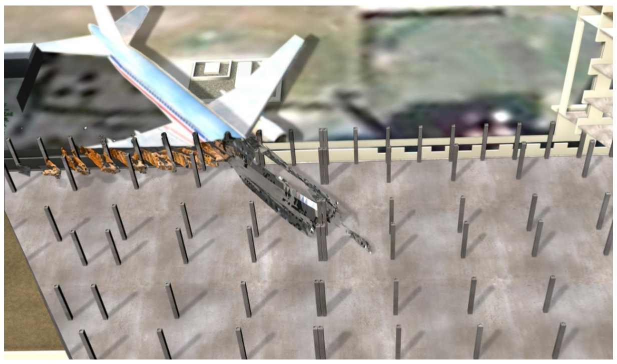


Figure 1 Animation frame. The top floors are not shown to reveal the simulated aircraft / concrete columns impact.

regardless of the scientific or engineering domain.

1.2 Motivation

A high-quality visualization of the results of a simulation first requires that the objects whose interaction is simulated be rendered using state-of-the-art rendering techniques. The second requirement is that the simulation be placed in the context of the immediate surrounding scene. For this the scene has to be modeled and rendered along with the simulation results.

Such a visualization makes the results and conclusions of the simulation directly accessible to others than the specialists that designed the simulation, without sacrificing scientific accuracy. This will make scientific simulations powerful tools that will routinely be used in a variety of fields including national security, emergency management, forensic science, and media.

A good visualization ultimately leads to improvements of the simulation itself. High-quality images quickly reveal discrepancies with experimental data observed over the years or recorded specifically for fine tuning the current simulation.

1.3 Process overview

Figure 2 gives an overview of the process that converted the heterogeneous data documenting the event into the desired visualization.

The first step in creating the simulation was to generate the element meshes suitable for FEA. To keep the scene complexity within manageable limits, only the most relevant components of the aircraft and of the building were meshed. Then, the material model parameters were tuned during test simulations to achieve correct load deflection behavior. The FEA code was run on the full resolution meshes to simulate the first 250 milliseconds of the impact over 50 states.

The visualization part of the project began with modeling the Pentagon building from architectural blueprints using a CAD tool. The geometric model of the building and the surroundings were enhanced with textures projected from high-resolution satellite and aerial imagery using a custom tool. The 3ds max aircraft model used for visualizing the approach was readily available. The 3.5 GB of state data describing the mesh deformations was simplified, converted and imported into the animation system through a custom plugin. The imported meshes were aligned with the surrounding scene and enhanced with rendering material properties. Finally the integrated scene was rendered from the desired camera paths.

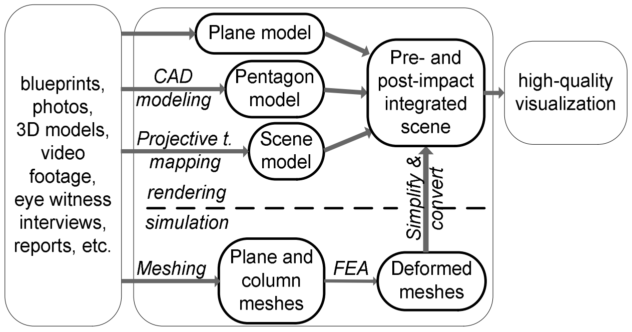


Figure 2 Process overview.

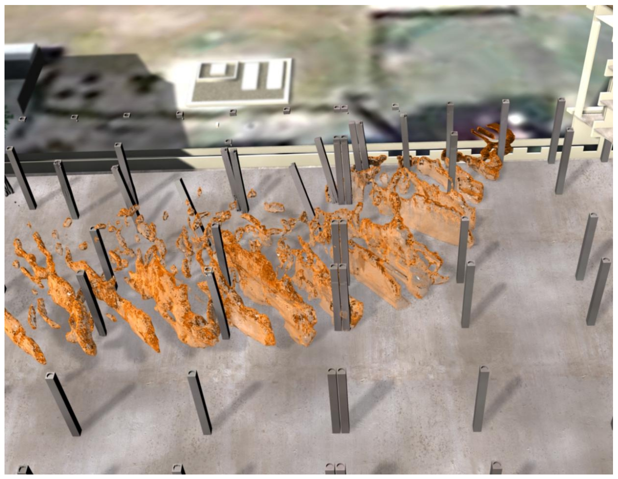


Figure 3 Visualization of the jet fuel.

Prior work is discussed next. The remainder of the paper is organized as follows. Section 3 describes the simulation; section 4 describes modeling the part of the scene not involved in the simulation; section 5 covers importing the simulation data into the animation system. Results are presented for each section separately. All the timing data was obtained on Pentium 4 Xeon, 2 GHz, 2 GB workstations. Discussion and directions for future work conclude the paper.

2. PRIOR WORK

Baker et al. [1] describe the simulation of a bomb blast and its impact on a neighboring building. The scenario investigated matches the 1996 attack on the Khobar towers. Two computational codes were used. The blast propagation was computed using CTH [3] at the Army's research lab in Vicksburg [6]. Results of the CTH calculation are used as initial pressure loadings on the buildings and Dyna3D [4] is then used to model the structural response of the building to the blast. The results were visualized in the Dyna3D postprocessor and VTK (visualization toolkit [7]) using standard visualization techniques such as slicing and isosurfacing. The researchers report the difficulty of visualizing the large data sets; the solutions employed are reducing resolution, decimation and extraction of regions of interest. Enhancing the quality of the visualization using photographs is mentioned as future work.

A considerable body of literature in nuclear engineering is dedicated to simulating the crash of an aircraft into a concrete structure. Provisions for aircraft impact on reinforced concrete structures are incorporated into the Civil Engineering codes used for the design of nuclear containment structures. A full-scale test was conducted by Sugano et al. [2] to measure the impact force exerted by fighter aircraft (F-4D) on a reinforced concrete target slab. The study provided important information on the deformation and disintegration of the aircraft. A simplified computational model was also developed in order to capture the global response of the impact. This study provided us the experimental evidence that the airframe and the skin of the aircraft alone are not likely to cause the major damage on reinforced concrete targets.

To place the simulation in context we had to model and render the surroundings of the Pentagon. Research in image-based rendering (IBR) has produced several successful approaches for rendering complex large-scale natural scenes. The QuicktimeVR [9] system models the scene by acquiring a set of overlapping same-center-of-projection photographs that are stitched together to form panoramas. During rendering the desired view is confined to the centers of the panoramas. In our case it was important to allow for unrestrained camera motion so we dismissed the approach.

Image-based rendering by warping (IBRW) [10] relies on images enhanced with per-pixel depth. The depth and color samples are 3D warped (reprojected) to create novel views. Airborne LIDAR sensors can provide the depth data at appropriate resolution and precision. In the case of our project no depth maps of the Pentagon scene were available and we could not use IBRW. In light field rendering the scene is modeled with a database containing all rays potentially needed during rendering. The method does not scale well: the number of images that need to be acquired and the ray database grow to impractical sizes for large-scale scenes.

An approach frequently used for modeling large urban scenes combines images with coarse geometry into a hybrid representation. A representative example is the Façade system [11] which maps photographs onto buildings modeled with simple primitive shapes. The system was used to model and realistically render a university campus environment. The relatively simple geometry of the Pentagon building and the availability of photographs of the area motivated us to choose a hybrid geometry / images approach as described in section 4.

3. LARGE SCALE SIMULATION

FEA codes are among the most flexible and competent tools for simulating physical phenomena. A simulation is described by providing a geometric description, a set of constitutive models that capture non-linear material behavior, initial conditions, and the interaction of various components of the model through contact algorithms. The geometric description is in terms of nodes (points in 3-space) and elements (beam, shell and volume) partitioning the geometric objects. The elements have associated material properties that describe their behavior under strain. The simulation code integrates differential equations that express the material characteristics and the interaction and energy exchange between materials in contact (or in a field). Failure of elements in the simulation is achieved by imposing a maximum strain limit in the material model, and eroding elements that reach the limit. These elements are not considered in the dynamic equilibrium of the model in the following time steps. Physically this means that the material tears or breaks at that locale. This approach enables the wings to cut through the reinforced concrete columns. Erosion of elements is a technique that is essential for simulating penetration problems.

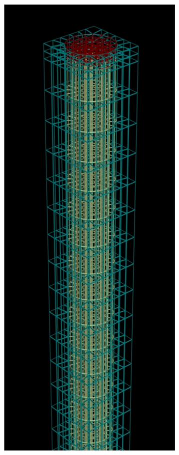


Figure 4 column FEM.

Based on careful consideration, our simulation hypothesis is that the most massive structure, causing the bulk of the damage through its kinetic energy, has been the liquid fuel (kerosene) in the tanks of the aircraft. At impact, the plane was carrying an estimated 5,200 gallons of fuel and had a speed estimated at 480 mph. Damage inspection revealed that the performance of the building depended crucially on the spirally reinforced concrete columns of the building.

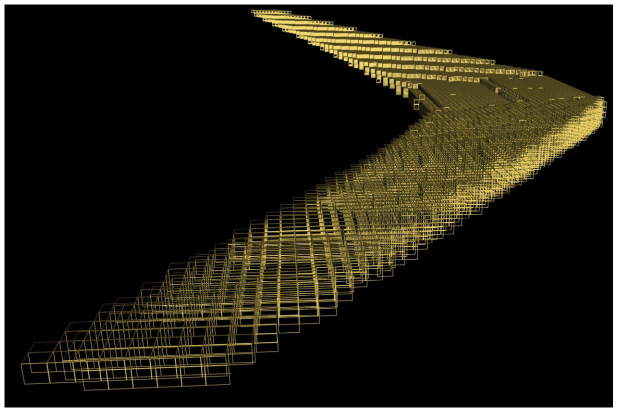
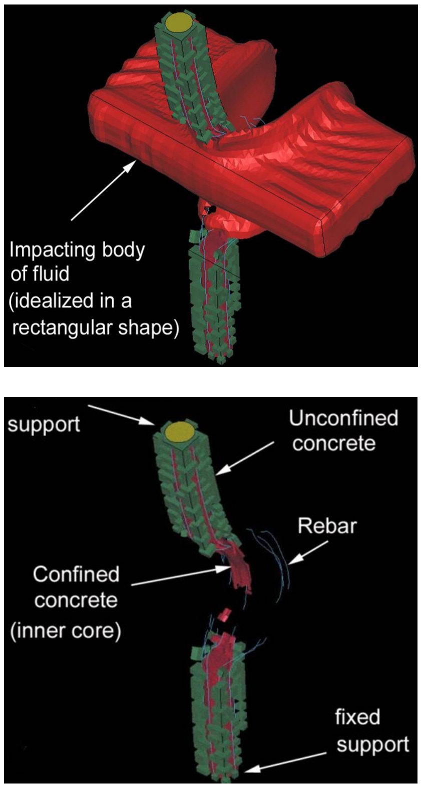


Figure 5 Eulerian mesh cells with >25% liquid occupancy.

Accordingly we concentrated on modeling the columns and the fuel. Figure 4 shows the finite element mesh (FEM) for the spirally reinforced concrete columns. We modeled the confined concrete core, the steel rebars, and the unconfined concrete cover (fluff). The column hexahedral elements are 7.5 x 7.5 x 15 cm in size. The column is anchored by the floor and ceiling supports (red in the figure). The fuel was modeled using an Arbitrary Lagrangian-Eulerian (ALE) formulation that integrates the Navier-Stokes equations of fluid dynamics for the motion of the liquid fuel. The Eulerian mesh is able to expand in order to enclose the splashing liquid fuel. Automatic mesh motion is achieved by following the mass-weighted average velocity of the ALE mesh. The fuel is specified in the Eulerian mesh in terms of per-cell fractional occupancy values. Figure 5 shows the liquid in the initial configuration. The Eulerian mesh elements are 15 x 15 x 15 cm in size.


Impacting body of fluid (idealized in a rectangular shape)
support
Unconfined concrete
Rebar
Confined concrete (inner core)
fixed support
Figure 6 Test simulation for tuning column / liquid interaction.

A 3ds max model of the Boeing 757 was obtained from a game company [15] and formed the basis for creating the FEM of the plane. The meshed model includes fuselage stringers reinforcing the body of the plane and ribs in the wing structure, as well as the fuselage floor. Those are the structural elements of the plane deemed to be most significant. Our custom mesh generation tool uses a set of hard points in the wings and the body. The mesh generation is parametric, which allows for conveniently generating meshes at various resolutions.

In order to calibrate the columns used in the simulation, a reinforced concrete column was analyzed under impact loading. Figure 6 (top) shows the calibration column subjected to high-speed impact with liquid. The liquid mass was idealized as a block (shown in red color). Figure 6 (bottom) illustrates the damaged state of the column after impact. Erosion of elements in the column allowed us to model penetration of the fluid and the splitting of the column into two pieces after impact. Different failure strain limits were used for the unconfined fluff cover and the confined concrete inner core. The steel rebars were also assigned failure strain limits in order to model the rupture behavior of the reinforcement.

The mesh density balances accuracy and model size to maximize resolution and fidelity while staying within software and hardware limitations. At 954 K nodes, the simulation took approximately 6 hours per recorded state on an IBM Power-4 platform with 8 processors and 64 GB of memory. The integration step size was 0.1 milliseconds, and we recorded 50 states, 5 milliseconds apart. The disk size of each state is 70 MB, for a total of 3.5 GB.

4. SURROUNDING SCENE

We decided to model the surrounding scene for two reasons. First we wanted to visualize the trajectory of the plane immediately before the collision. Second, we wanted to place the simulation results in context to make it easily understood by someone who was not closely involved with the investigation.

As described earlier, our approach was dictated by the available data documenting the scene. From the architectural blueprints we produced a CAD model of the building. The damage in the collapsed area was modeled by hand to match available photographs. The region surrounding the Pentagon was simply modeled with a large plane. The geometric models were enhanced with color using high-resolution satellite [16] and aerial imagery [8].

In order to apply a photograph to a geometric model two problems need to be solved. First one has to find the pose of the camera in a model-defined coordinate system (camera matching). Second the photograph pixels need to be mapped to the model triangles that are visible to the camera (projective texture mapping [17]). The basic functionality is available in animation systems such as 3ds max and Maya. We decided to implement our own camera matching / projective texture mapping tool to have more control over the camera matching and to create a conventionally texture mapped model. Such a model with individually texture mapped triangles can then be easily combined with other models (namely the approaching aircraft and the results of the simulation) and allows using multiple reference photographs with good control over the triangle to photograph assignment.

We find the camera pose using correspondences between the photograph and the geometric model. Since the camera used to take the photographs is not available we also calibrate for the focal length. The focal length is the only intrinsic parameter of the camera model used: the center of projection is assumed to project in the center of the pixel grid, the pixels are assumed to be square and the lens distortion is ignored. We use this idealized model since the scene is flat (the height of the Pentagon building is small compared to its horizontal dimensions) and nearly coplanar points make the calibration for complex camera models numerically unstable. We search for the seven unknowns using the downhill simplex method. The starting position is obtained by rendering the model and manually adjusting the view such that the rendered image roughly matches the photograph. Convergence is achieved in negligible time. For a 3000 x 2000 pixels image the camera matching error is on average 3.5 pixels for 10 correspondences.

Once the view is known, the camera is transformed in a projector and the photograph pixels are deposited on the surface of the triangles to create individual texture maps. The algorithm proceeds as follows (Pseudocode 1). *IB* stores the IDs of the triangles that are seen by the photograph.

```
Render model from camera view CV
  in item buffer IB and z buffer ZB
For each triangle T in IB
  Project T in CV
  Find longest edge e, corr. height h
  Allocate e x h texture map TM
  Set texture coordinates for T
  For each texel t in TM
    If t outside T continue
    Project t in CV at p
    If hidden by ZB continue
    Set t to photograph pixel p
  Set edge texels
```

Pseudocode 1 Texture generation

A texture map is generated for each visible triangle. The texture is aligned with the longest edge of the projected triangle and with its corresponding height. The lengths of the two segments in pixels give the texture resolution. By choosing the resolution this way, the texture subsamples the photograph in the part of the triangle near the camera and supersamples it at the far end. Subsampling implies losing some of the color information of the photograph. We have experimented with setting the texture resolution to the maximum sampling rate encountered at the near end of the triangle. The Pentagon building model contains long triangles and the conservative resolution produced excessively large textures. Using the z buffer the texture is set only for the part of the triangle actually seen in the photograph. This is important when other images are used to complete the texture of the triangle. To correctly handle triangles that have a thin projection, the texels traversed by the edges are set the same way (without the triangle membership test).

The building and ground plane model consisting of 25 K triangles was sprayed with a 3000 x 2000 pixels photograph. The resulting texture mapped model produced realistic visualizations of the Pentagon scene. Figure 8 shows an image rendered from a considerably different view than

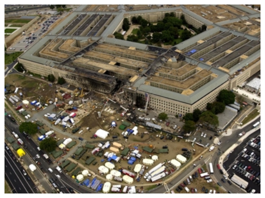


Figure 7 Photograph

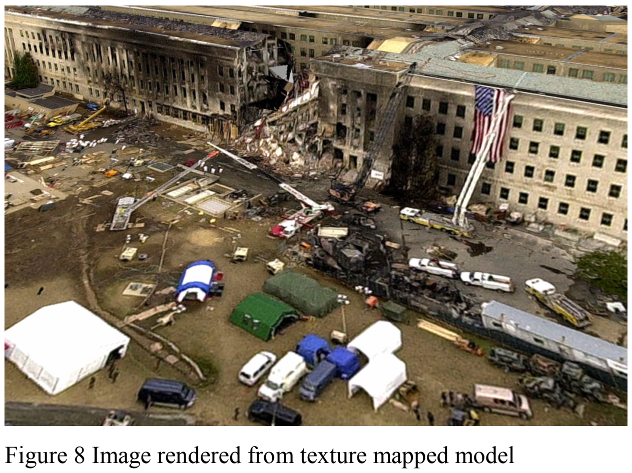


Figure 8 Image rendered from texture mapped model

the view of the reference photograph, which is shown in Figure 7. The total disk size of the texture files is 160 MB. The difference when comparing to the 24 MB of the reference photograph is due to the texels outside of the triangle, to the texels corresponding to the hidden part of the triangle, to the thin triangles that have a texture larger than their area and to our simple merging of individual texture images that vertically collates 10 images to reduce the number of files. For now we rendered the scene offline so the large total texture size was not a concern. For real time rendering, the texture size has to be reduced. A simple greedy algorithm for packing the textures involving shifts and rotations is likely to yield good results. The rotation can be propagated upstream to the spraying to avoid the additional resampling.

5. INTEGRATION

The simulation results files are directly imported in 3ds max via a custom plugin. The 954 K nodes of the FEM define 355 K hexahedral (*solid*) elements used to model the column core and the fluff, 438 K hexahedral elements for the liquid elements, 15 K quadrilateral (*shell*) elements used to define the fuselage and floor of the aircraft, and 61 K segment (*beam*) elements used to define the ribs and stringers of the aircraft. The importer subdivides the simulation scene into objects according to materials to facilitate assigning rendering materials.

5.1 Solid objects

Ignoring the liquid for now, the 12, 2 and 1 triangles per solid, shell and beam elements respectively imply about 4.3 M triangles for the solid materials in the simulation scene. This number is reduced by eliminating internal faces, which are irrelevant during rendering. An internal face is a face shared by two hexahedral elements. Because elements erode, faces that are initially internal can become visible at the fracture area. For this an object is subdivided according to the simulation states; subobject *k* groups all the elements that erode at state *k*. Discarding the internal faces of each subobject is done in linear time using hashing. This reduces the number of triangles to 1.3 M, which is easily handled by the animation system.

However, importing the mesh deformation into the animation system proved to be a serious bottleneck. The mesh deformations are saved by the FEA code as node positions at every state. The animation system supports *per vertex* animation but creating 50 position controllers for each of the remaining 700 K nodes takes days and the resulting scene file is unusable. The practical limit on the number of animation controllers is about 1 M. The number of animation controllers is reduced in two ways. First, the importer does not animate nodes with a total movement (sum of state to state movement) below a user chosen threshold (typical value 1 cm). Second, the trajectories of each node are simplified independently by eliminating (i.e. not creating) controllers for the nearly linear parts. We have experimented with two ways of simplifying the trajectory. In the first approach, a controller is removed if the resulting trajectory is, at every state, within a threshold (typical value 1 cm) of the original trajectory. This enforces the threshold globally at the cost of an order Ns^2 running time where N is the number of nodes and s is the number of states. Our second approach considers triples of states A, B, C and removes the controller for B if B is closer than 1 cm from the line AC. The next triple considered is A, C, D if B is removed and B, C, D if not. When the threshold is considerably less than the amount a node moves between states, the result is virtually the same as in the case of the first approach, with the benefit of an order Ns running time. After simplification, 1.8 M controllers remained. We distributed the simulation scene over three files, each covering one third of the simulation. Materials and cameras can of course easily be shared among several files. Importing the solid objects takes 2 hours total, out of which 1 hour is needed for the third part of the simulation. Once the solid objects are loaded, the animator assigns them standard 3ds max materials.

5.2 Liquid objects

The liquid data saved at every state contains the position of the nodes of the Eulerian mesh and the fractional occupancy values at that state. The liquid could be directly rendered from the occupancy data using volume rendering techniques. We chose to build a surface boundary representation first in order to take advantage of the rendering capabilities of the animation system. For every state the importer selects the Eulerian mesh elements that have a liquid occupancy above a certain threshold (typical value 25%). The internal faces are eliminated similarly to the solid object case. Once the liquid is imported, two 3dsmax modifiers are applied. For now the modifiers are applied manually by the animator; in the future this step will be moved inside the importer. The first is the "relax" modifier, which changes the apparent tension of the surface by moving vertices toward an average center point. By relaxing the mesh, it rounds the edges without adding or removing faces. After relaxing the mesh, a "smooth" modifier is added to average the object's normals, which creates a surface that reacts well to light, reflection, and refraction.

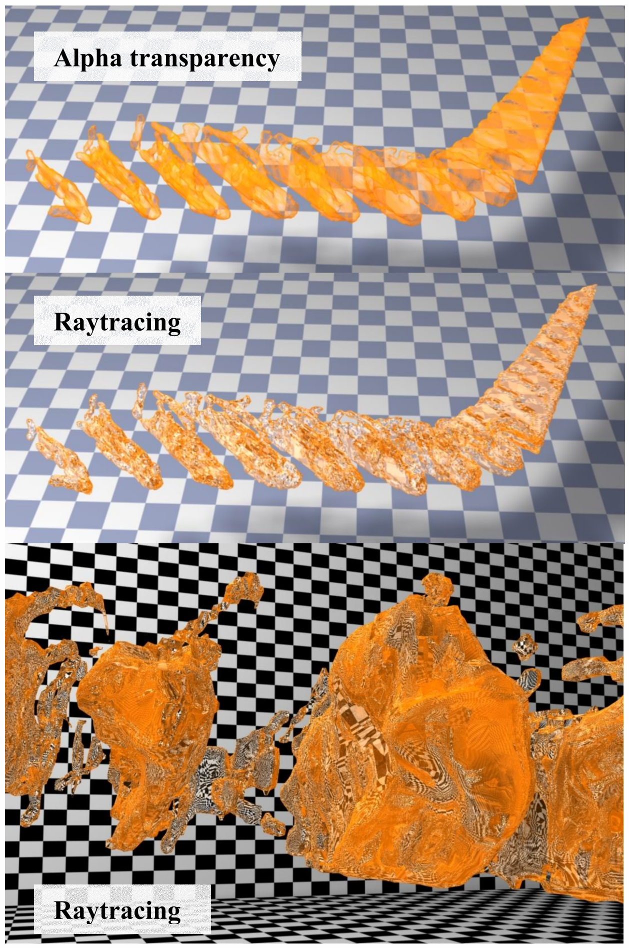


Figure 9 Fuel rendered in animation system

The liquid material is a 3ds max standard material with a falloff map in the opacity channel. This map attenuates the transparency of the object relative to the camera. It makes the object appear transparent where its normals are pointing in the same direction as the camera (the Fresnel effect). Optional raytrace maps are also controlling the reflection and refraction of the liquid material. Figure 9 shows the liquid rendered once without the raytrace maps (5 seconds render time, top image) and then twice with the raytrace maps (5 minutes middle image, 55 minutes bottom image). Refraction and surface reflections improve the realism of the second image while the less expensive technique produces acceptable results when the liquid is integrated with the complex scene.

As in the case of the solid objects, animating the liquid is challenging. There are two fundamental approaches: to consider the liquid a complex object that moves and deforms over the simulation time or to frequently recompute the liquid object from the occupancy data, possibly at every animation frame.

The first approach is in the spirit of animation systems where the same geometric entity suffers a series of transformations over the animation time span. The state of the geometric entity is known at the simulation states; it can be computed by thresholding or isosurfacing the occupancy data. In order to define a morph that produces the animation frames in between the states, correspondences need to be established. This is challenging since the liquid can change considerably from one state to another; this implies different number of vertices, different local topologies (drops, liquid chunks separating and reuniting).

We have attempted to implement this approach using the Eulerian mesh as a link between states. First, for each simulation state, the liquid elements occupied above a given threshold are selected. Then the Eulerian mesh is split in liquid objects *(i, j)* defined by the set of elements that contain liquid from state *i* to *j*, and do not contain liquid at states *i-1* and *j+1*. In other words the object *(i, j)* contains the liquid "alive" between states *i* and *j*. The internal faces are removed and the object, which is a subset of the Eulerian mesh, is animated according to the known positions of the mesh nodes. Because the occupancy values vary considerably from one frame to another, many small liquid objects are generated. This leads to a large number of position controllers.

The approach of defining the liquid with independent objects corresponding to snapshots along the simulation timeline has proven to be more practical. Visibility controllers automatically generated by the plugin define the appropriate life span for each object. To smooth the transition the objects are faded in and out at a negligible cost of 4 controllers per liquid object. Currently the liquid is modeled with one object per state. The 50 liquid objects total 1.5 M triangles. By interpolating the occupancy data one could generate one snapshot for every animation step. When playing back the 50 states over 30 seconds at 30 Hz, 900 liquid objects need to be generated, which exceeds a practical geometry budget. We are investigating generating the liquid objects during rendering. This implies finding a way for applying the modifiers and assigning the liquid material automatically, during rendering.

6. DISCUSSION AND FUTURE WORK

As one of the five member team to inspect the damaged building, Mete Sozen is a coauthor of the Pentagon Building Performance report [8]. The most massive impacting element was the fuel. The fuselage of the aircraft has little strength under axial impact, as confirmed by the simulation and validated by actual experiments [2]. The simulation clearly shows that the structural damage occurs only when the fuel mass hits. The simulation can be extended to cover a longer period of time, with denser states, involving higher resolution meshes; other possible extensions are modeling the building and aircraft in more detail and including the effects of the explosion, of the high temperatures and of the combustion. The bottleneck in the simulation runs was the amount of memory available on the various platforms used. The power of large memory spaces has recently been combined with the convenience of desktop computing. We are in the process of setting up simulation runs on recently acquired Itanium PCs.

We have implemented a set of tools for integrating the simulation results with the surrounding scene in a commercial animation package. All tools can directly be reused for producing other visualizations. The plugin importer and 3ds max are now commonly used by the civil engineering researchers of our team. Initially the use was restricted to producing illustrations of their work; they are now using it to inspect the result of simulations. Scientific simulation researchers and commercial-simulation-systems developers have shown great interest in the quality of the visualizations and we have initiated several collaborations. Except for the liquid raytracing, the integrated scene could be explored interactively. The VRML format for example does support triangle meshes with per vertex animation and can be rendered with hardware support by many browser plugin or stand alone 3D viewers.

The link created between simulation and animation has to be further developed. The current bottleneck is the animation of the deforming meshes. Paradoxically the animation system performs better if the animation is specified by geometry replication. We will continue to investigate this problem. The importer could be extended to create dust, smoke and fire automatically. For example when a concrete element erodes, it should be turned in fine debris or dust animated according to the momentum that the element had before eroding. The simulation driven recreation of low visibility conditions will be valuable in virtual training. The first effort described here relied as much as possible on the capabilities of the animation system. These can be extended to include classic visualization techniques. Well studied algorithms can be employed and we do not foresee any major difficulty.

Good visualizations facilitate the comparison of the simulation results to observed or recorded real data. Providing tools to assist and then fully automate the comparison is one of our longer term goals. Computer vision techniques are a possibility. This task is greatly facilitated if the experiment scene or actual event scene are captured by depth maps in addition to the traditional photographs. In our case, recording the shape of the columns affected by the impact would have been both easy and very beneficial.

7. ACKNOWLEDGEMENTS

We would like to thank Jim Bottum and Gary Bertoline for making this project possible, William Whitson for his help with the supercomputer runs, Hendry Lim and Mihai Mudure for implementing texture spraying, Mary Moyars-Johnson and Emil Venere for publicizing this work, Amit Chourasia for modeling the Pentagon building, Jason Doty for his help with producing the first video illustration of this project and Raj Arangarasan for his help with an earlier implementation. This work was supported by ITaP, Computer Science Purdue, NSF, ARO and DARPA.

References

[1] M. Pauline Baker, Dave Bock, Randy Heiland. Visualization of Damaged Structures. NCSA, University of Illinois. URL: http://archive.ncsa.uiuc.edu/Vis/Publications/damage.html

[2] T Sugano et al. Full-scale aircraft impact test for evaluation of impact force, Nuclear Engineering and Design, Vol. 140, 373-385, 1993.

[3] McGlaun, J. M., Thompson, S. L. and Elrick, M. G. 1990. "CTH: A three dimensional shock wave physics code", Int. J. Impact Engng., Vol. 10, 351 – 360.

[4] J. O. Hallquist and D. J. Benson, Dyna3D User's Manual (Nonlinear Dynamic Analysis of Structures in Three Dimensions), Report #UCID-19592-revision-3, Lawrence Livermore National Laboratory, Livermore, California, pp. 168, 1987.

[5] LS-DYNA, URL: http://www.ls-dyna.com/

[6] http://www.hpcmo.hpc.mil/Htdocs/UGC/UGC98/papers/3b_chal/

[7] http://public.kitware.com/VTK/

[8] "Pentagon Building Performance Report," Assoc. Of Civil Engr., 2003, 88 pages.

[9] S. Chen. QuicktimeVR- an image-base approach to virtual environment navigation. In Proc. SIGG. '95, pages 29-38.

[10] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In Proc. SIGGRAPH '95, pages 39-46, 1995.

[11] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and Rendering Architecture from Photographs. In Proc. of SIGGRAPH '96.

[12] M. Levoy and P. Hanrahan. Light field rendering. In Proc. of SIGGRAPH '96, pages 31-42, 1996.

[13] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In Proc. of SIGGRAPH '96, pages 43-54, 1996.

[14] G. Bell, The future of high-performance computers in science and engineering, CACM 32, 1091-1101, 1989.

[15] Casper, Terry, Amazing 3D Graphics, Inc., P.O Box 1821, Payson, Arizona 85547, URL: www.amazing3d.com, 2002.

[16] SpaceImaging, URL: http://www.spaceimaging.com/gallery/9-11/default.htm

[17] M. Segal, C. Korobkin, R. van Sidenfelt, J. Foran, and P. Haeberli, Fast Shadows and Lighting Effects Using Texture Mapping. *Computer Graphics*, 26(2), 249-252 (1992).

[18] Alias | WaveFront, URL: http://www.aliaswavefront.com

[19] Discreet, URL: http://www.discreet.com/products/3dsmax/