A High-Quality Physically-Accurate Visualization of the September 11 Attack on the World Trade Center

Paul Rosen, Voicu Popescu, Christoph Hoffmann, Ayhan Irfanoglu
Purdue University

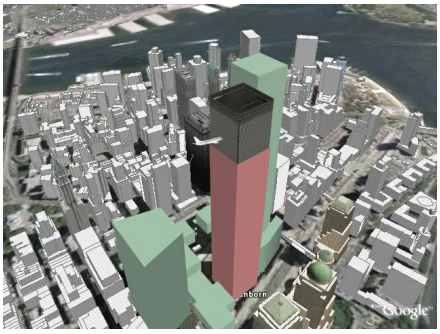

Figure 1. Simulation integrated into lower Manhattan scene using Google Earth.

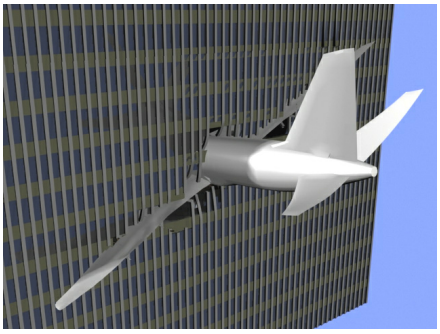

Figure 2. Impact visualization from outside the building. Right image highlights core column damage by rendering all other elements with transparent materials.

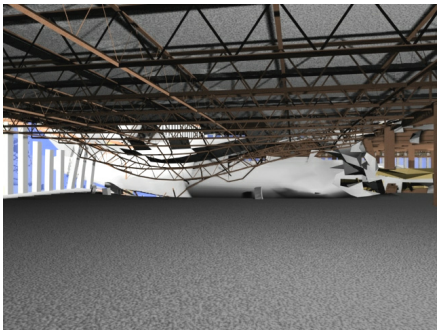

Figure 3. Region between façade and building core, during (*left*) and after (*right*) impact.

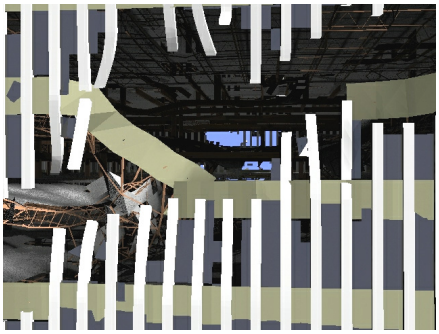

Figure 4. Visualization of façade breach.

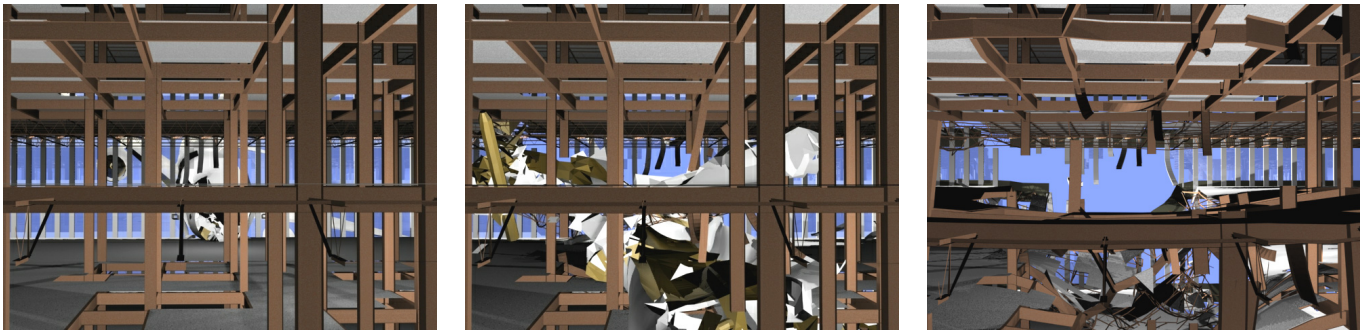

Figure 5. Visualization of the two floors most affected by the impact, chronologically, from left to right.

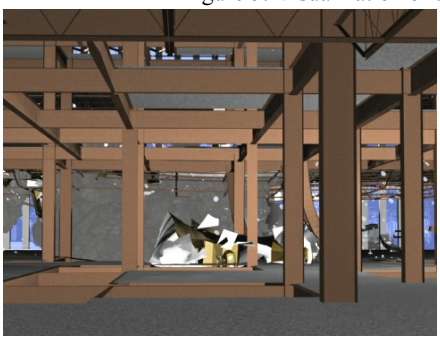

Figure 6. Dust and glass debris generated automatically from eroding elements.

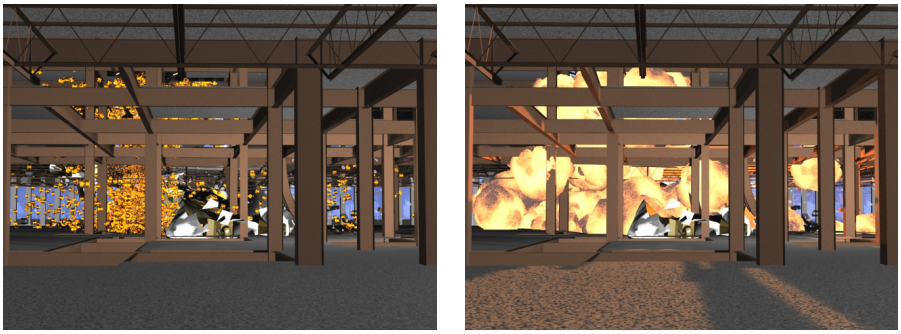

Figure 7. Fire visualization (*right*) generated automatically from SPH simulated jet fuel (*left*). The shape of the fire follows the particle distribution.

ABSTRACT

In this application paper we describe the efforts of a multi-disciplinary team towards producing a visualization of the September 11 2001 Attack on the North Tower of New York's World Trade Center. The visualization was designed to meet two conditions. First, the visualization had to depict the impact with high fidelity, by closely following the laws of physics. Second, the visualization had to be eloquent to a non-expert user. This was achieved by first designing and computing a finite element analysis (FEA) simulation of the impact between the Boeing 767 and the top 20 stories of the building, and then by visualizing the FEA results with a state-of-the-art commercial animation system. The visualization was enabled by an automatic translator that converts the simulation data into an animation system 3D scene by removing simulation data with little visual relevance and by adding details important for the visualization that were ignored by the simulation. We built upon a translator developed part of our previous efforts to produce a high-fidelity visualization of the September 11 Attack on the Pentagon. The translator was extended to handle beam elements with a variety of complex profiles, to handle smoothed particle hydrodynamics (SPH) liquid representations, to enable and control visualization of fire and of disintegrating elements, and to better scale with the number of nodes and number of states.

Keywords: finite-element analysis, visualization, high-fidelity, scalability, translation, 9/11 Attack on World Trade Center.

1 INTRODUCTION

Visualization has long been used by experts in a variety of domains as a powerful tool that assists them in their mission. The visualization system is typically developed in collaboration with domain experts and achieves—automatically or with user guidance—the detection and isolation of relevant data subsets, which are then converted into salient visual representations. The rules of this conversion and the resulting visualization language are well familiar to the domain experts, for whom it facilitates broad-band assimilation of information. However, this same visualization language can be cryptic to anyone outside the narrow circle of domain experts, which is a problem when the interest in the visualization transcends a single domain.

This is a serious limitation in the case of computer simulations. Simulation codes and computing hardware are now sufficiently powerful to enable high-fidelity simulations that track in detail complex interactions in large and complex scenes. The results of such simulations are often of great interest to a group of users with heterogeneous expertise, yet the visualization modules of simulation systems typically cater only to the experts who designed the simulation.

This application paper describes the collaborative efforts of visualization and civil engineering researchers towards producing a simulation of the September 11 2001 Attack on New York's World Trade Center. The interest in such a simulation transcends civil engineering and includes emergency response, defense, and society in general. Therefore we pursued two major goals. On the one hand the simulation had to follow the laws of physics as closely as possible. On the other hand, the simulation results had to be presented through a visualization that is eloquent to users outside of civil engineering. The goals of simulation fidelity and of broad accessibility to the simulation results through visualization are somewhat contradictory and are rarely pursued in combination.

Employing state-of-the-art numerical simulation code has the advantage of a high degree of confidence in the fidelity of the physical simulation, but suffers from the disadvantage of lower fidelity visualization provided by post-processing modules that are one or several steps behind the state-of-the-art in general purpose visualization. Employing a state-of-the-art animation system on the other hand enables high-quality visualization but there is little confidence that the rendered imagery faithfully depicts the actual events. We combine the advantages and avoid the disadvantages of both approaches by computing the simulation in a state-of-the-art commercial simulation system and by visualizing the simulation results using a state-of-the-art commercial animation system.

We generated finite element models of the Boeing 767 and of the top 20 stories of the North Tower of the World Trade Center (WTC-I) consisting of beam, shell, SPH, and solid elements. The finite element models were used to compute an FEA simulation of the impact using LS-DYNA [1]. The simulation tracked the impact over one second of real time. Simulation results were saved for 400 output time steps, thus every 2.5ms. The simulation results were imported into 3ds Max [2] where, relying on state-of-the-art geometry, material, light, and visual effects editors a high-quality visualization of the simulation results was produced (see Figures 2-7 and accompanying video). Once the light, material, and effects parameters were tuned, producing the visualization was fully automatic: materials were applied automatically to the simulation data and visual effects such as fire, dust, and glass debris were seeded automatically based on simulation data. The visualization was placed into context by modeling and importing the WTC plaza buildings into a Google Earth [3] 3D scene of lower Manhattan (Figure 1).

The importer translates the simulation data into an animated 3D scene amenable to high-quality visualization, effectively linking the worlds of simulation and animation. The process of translation implements two tasks. First, simulation data with little visualization relevance is discarded. Examples include removing internal faces of structures modeled with opaque solid elements, discarding values of physical quantities not intended to be visualized (i.e. pressure, momentum), and simplifying planar surfaces that are not affected by the impact and are therefore excessively tessellated by shell elements.

The second task implemented by the translation is to enhance or add detail with high visual relevance that was crudely approximated or even ignored due to its little simulation relevance. For example, once an element erodes, the simulation code simply eliminates it from subsequent computations. Assuming that the element breaks into many small fragments, this is an acceptable approximation from the simulation standpoint, since the expense of tracking each fragment individually is not justified by the fragment's impact on the simulation. However, for example, if the element models a concrete wall that turns into dust when submitted to excessive stress, the eroding element has a large visual impact which should be captured by the visualization.

Another example of visual detail added during translation is the geometry needed to model the actual beam profile since the simulation uses only two nodes and a normal per beam element regardless of the beam profile (the shape is accounted for by physics equation in the FEA) and drawing all beam elements as segments is unacceptable. A third example is the addition of fire visualization. Our simulation did not take into account the effects of the explosion and of the ensuing fire. The SPH elements modeling the jet fuel are used by the translator to automatically control fire visualization matching the dispersing fuel.

The importer used for this work is based on a LS-DYNA to 3ds Max importer we have originally developed for producing a visualization of the September 11 Attack on the Pentagon [4]. We have extended the importer to support:

- beam elements with complex profiles,
- SPH liquid simulation,

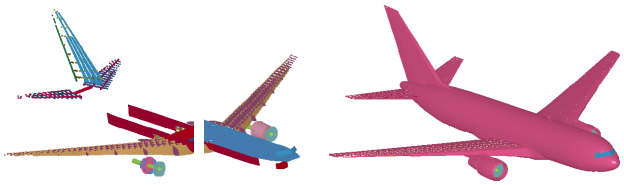

Figure 8. Layers of the aircraft finite element model.

- automatic fire visualization controlled by SPH elements,
- automatic dust and debris visualization controlled by eroding elements,
- out-of-core simulation visualization.

The resulting importer is a powerful, scalable, and general tool suitable for visualizing other simulations. Although the translator was developed in the context of LS-DYNA and 3ds Max, it can be extended to support other pairs of simulation/animation systems. The visualizations produced achieve the dual goal of physical and visual fidelity, surpassing what can be produced with typical postprocessors yet tracking the physical entities with the rigor of state of the art simulation code.

The remainder of the paper is organized as follows. Prior work is discussed next. The FEA simulation is described in Section 3. Section 4 describes the translation of the simulation data into an animation system 3D scene suitable for visualization. Section 5 describes using the animation system's resources to produce the visualization and presents the visualization results. Section 6 gives an overview of the implementation, and Section 7 provides conclusions and sketches possible directions of future work.

2 Prior Work

An earlier effort with similar goals to ours was the simulation of the impact of a bomb detonation on nearby buildings [5]. The specifics of the simulation match the 1996 attack on the Khobar towers. A first simulation of the blast computed the initial pressure loadings on the building, which were then used in an FEA simulation to model the structural response of the building to the blast. Visualization was performed using the postprocessor of the simulation system [6] and using the Visualization Tool Kit (VTK) [7]. The researchers mention enhancing the visualization with photographs as future work in order to improve the communication of the simulation results.

Considerable nuclear and civil engineering research effort is dedicated to the simulation of the crash of an aircraft into a concrete structure, in order to derive safe building codes for nuclear containment structures. A full-scale experiment with an actual fighter aircraft was conducted by Sugano et al. [8]. The experiment provided impact force/deflection measurements used to validate subsequent simulations.

In a previous effort, our team produced a visualization of the 9/11 Attack on the Pentagon. Like for the current work, the initial motivation of understanding the performance response of the building from a civil engineering stand point was augmented with producing a high-quality visualization that speaks to the public at large. The Pentagon building was simplified to the spiral-reinforced columns, which are the building's main structural components. Most of the kinetic energy of the aircraft was concentrated in the jet fuel, which was modeled with an Arbitrary Eulerian Lagrangian (ALE) mesh. The simulation showed a column destruction pattern similar to the one actually observed. The visualization footage was used by local, national, and international news agencies, and it continues to be the all-time most downloaded video file off Purdue University's web site.

Compared to the Pentagon visualization, the present effort not only involved a different scenario, but required developing the importer in new directions. First, whereas the Pentagon columns were modeled mainly with solid elements, the WTC-I was modeled primarily with beams, which required developing the beam visualization capability (see Section4.1). Second, the current effort required adding support for visualization of SPH liquid simulation (see Section 4.2). Third, the current effort eliminated the node animation bottleneck which would have been even more severe in the present context due to the complex beam profiles (see Section 4.3). Finally the present effort added automatic, simulation controlled, fire, dust, and debris visualization.

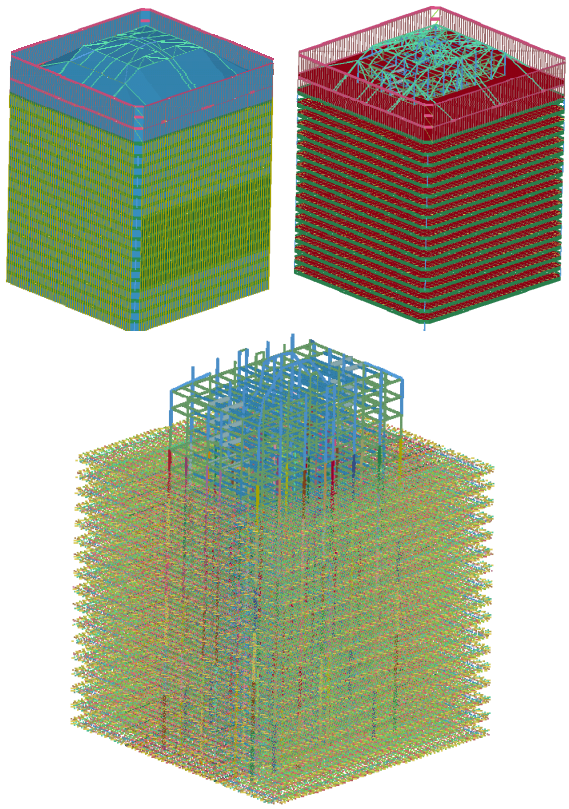

Figure 9. Layers of the WTC-I finite element model.

3 FEA Simulation

We modeled a Boeing 767-200 using both graphics models as well as published aircraft literature. The FEA model (Figure 8) includes structural elements, including ribs, stringers, keel beam, floor and more. The model was calibrated using Sugano's method as well as weight distribution information.

A model of the North Tower (WTC-I) was built by the civil engineering members of the team. It included all structural elements as well as the concrete floors (Figure 9). All stories were modeled, including those underground. The simulation restricted to the upper 20 floors of the building, however with increased detail meshing near the impact region so as to achieve high accuracy of the results. The aircraft and WTC-I model total 332,862 nodes; 87,188 SPH, 248,433 beam, 93,733 shell, and 674 solid elements. The titanium shafts of the two engines and the titanium undercarriage were modeled with solid elements. The FEA computation took 166 hours on an IBM Regatta with 16 Power-5 processors. The simulation data files for the 400 saved states comprise 20GB of disk space. The simulation begins at the moment of impact and covers 1 second of real time. Debris begins to re-emerge through the opposite face of the building at approximately 0.36s.

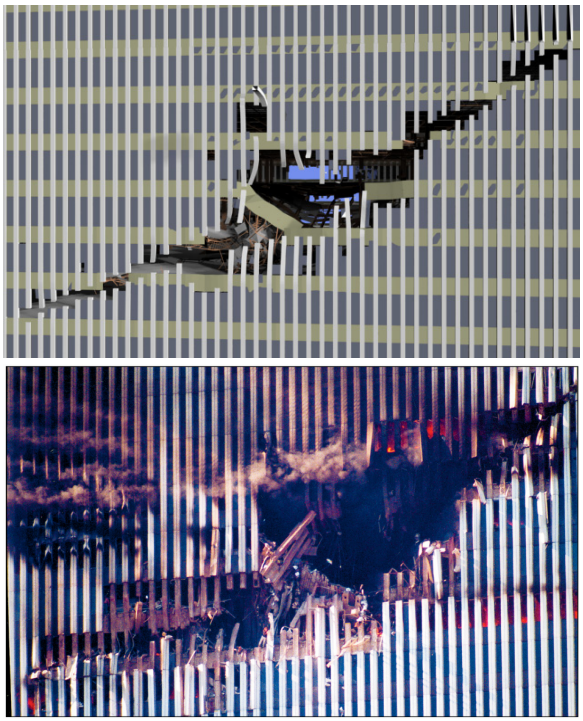

Figure 10. Comparison between our visualization (*top*) and actual photograph (*bottom*, © 2001 Roberto Rabanne [9]).

The façade damage computed by the simulation is in agreement with the observed damage (Figure 10). The core columns are essential to the structural integrity of the building, but no detailed data exists recording their performance. Instead, FEA simulations were used to assist estimating the impact response and the post-impact state of the core's structural elements. Based on this evidence it was found that some of the core columns were vulnerable to failure, and that a simple construct not dependent on exact determination would explain the collapse of the structure. The simulation study concentrated on the core structure of the tower and its possible behaviour under impact and thermal loads. The study did not seek to rule out other plausible mechanisms initiating collapse of the WTC-I tower, such as one due to loss of lateral bracing and buckling of perimeter columns induced by failure of open-web floor joists under thermal loads, or other failure mechanisms.

4 Translation of FEA Output into Animation Scene

Quadrilateral shell and hexahedral solid elements are imported and translated into 2 and 12 faces as previously reported [4].

4.1 Beam Elements

The FEA models contain beams with L, square, I, and T profiles (Figure 11). The simulation represents a beam element with two nodes and a normal, from which the importer reconstructs the actual profile. For example a thin/thick I beam element is modeled with 6/12 vertices per end point. In the visualizations shown in this paper the material thickness was ignored, but could be easily added at the cost of a factor of 2 increase in geometry complexity.

Element connectivity is not encoded in the simulation output files. The importer recovers beam element connectivity in *NlogN* time, where *N* is the number of beam elements, by sorting the beam elements on the node indices of their endpoints. Two beam elements with similar normals that share a simulation node will also share the vertices created at the shared endpoint, achieving C^0 continuity. C^1 discontinuity is masked by shading.

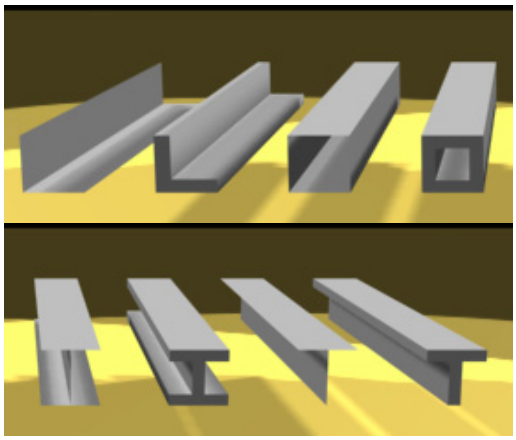

Figure 11. Beam profile types: L, square (*top*), I and T (*bottom*), thin and thick versions.

The geometry resulting from the shell, solid, and beam elements is automatically filtered through a 3ds Max geometry simplification tool which re-triangulates planar regions and achieves on average a factor of 2 non-lossy geometry load reduction (e.g. from ~2.3 to 1.2 million triangles).

4.2 SPH Elements

The jet fuel was simulated using SPH because of its advantage over ALE of easier set up in the context of the highly compartmentalized aircraft fuel tanks. The SPH elements are first imported as individual spheres of constant radius, then nearby spheres are automatically fused into a mesh using a 3ds Max geometry modifier, and finally a liquid material is applied to the resulting mesh (Figure 12).

4.3 Out-of-Core Animation

Animation systems such as 3ds Max support per-vertex animation through position controllers that define the position of a vertex at a given frame. In our previous work [4], the imported simulation scene was animated using position controllers. However, the approach proved to scale poorly with the number of states and the number of nodes. Indeed, position controllers are used to animate a few control points per character and animation systems are not designed to support large numbers of position controllers. Even by resorting to aggressive lossy compression of node trajectories, which have the undesirable effect of modifying the result of the simulation, the number of remaining position controllers was still too high resulting in a slow and unresponsive animation system.

The simulation data specifies one position for every node for every saved state. In our case, 400 states times 332,862 nodes amounts to more than 133 million node positions. By taking into account that a simulation node is converted on average to 5 animation scene vertices, the number of vertex positions approaches one billion.

In order to comfortably support this large number of vertex positions, we abandoned the position controller approach in favor of an out-of-core approach to the animation of the imported simulation data. A script controls the creation of a scene file for every desired visualization frame by invoking the importer, and then renders the frame. Only the simulation data corresponding to a single time step is loaded into the animation system at any time. The approach removes the position controller bottleneck and scales perfectly well with the number of states The out-of-core approach allows decoupling the time step used to save simulation data from the time step of the visualization frames. Typically the

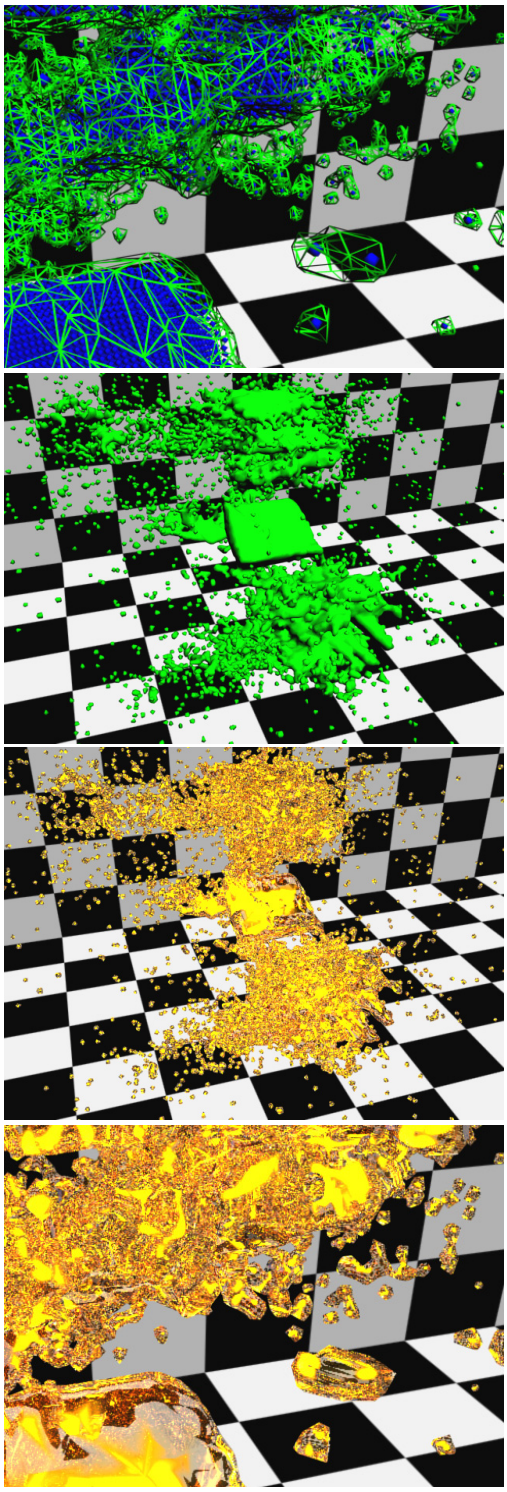

Figure 12. From the top: close-up visualization of conversion of SPH-element spheres (*reduced size, blue*) into fuel mesh (*green*), diffuse fuel mesh, ray traced fuel mesh, and ray traced fuel mesh close-up.

visualization examines the transition between two saved states over several frames, which is achieved by interpolating the simulation data.

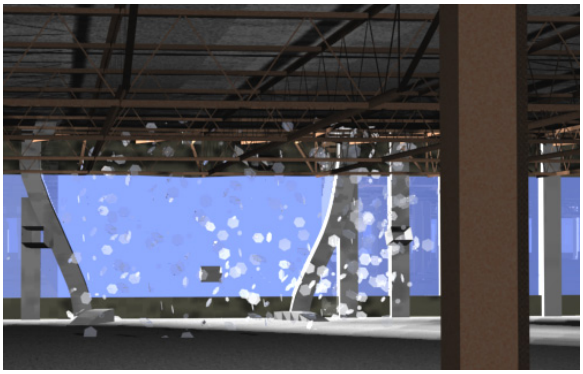

Figure 13. Glass debris effect.

5 Visualization in Animation System

Materials, lights, cameras, and camera trajectories are created relying on the sophisticated support available in the animation system. The materials are assigned automatically by a script to each scene as the simulation data is processed out-of-core.

The importer first generates a pseudo mesh from the simulation data which is then used by a script to control visual effects such as erosion (i.e. glass debris and dust), and fire (see Figure 13, Figure 6, and Figure 7). The pseudo mesh vertices are used for particle generation. In the case of erosion, the pseudo mesh vertices are created by the importer by down-sampling the set of nodes of the elements that erode (i.e. glass shells for the glass debris, and all other eroding elements for dust). The erosion pseudo mesh has a birth frame and a death frame specified by the importer. In the case of fire, the pseudo mesh vertices are derived from the SPH nodes and the mesh persists through the entire simulation.

Then the importer generates a script which creates a particle array using the pseudo mesh vertices as seeds, and the animator attaches the effect to those seeds. There are two types of particle arrays used. Erosion uses a *standard* particle array which shoots particles out of the seed. Space warpers (wind and gravity) then affect the motion of the particles. Fire uses a *sticky* particle array that creates particles and leaves each one of them attached to the position of its (animated) seed.

6 Implementation

The visualization is produced according to the pipeline depicted in Figure 14. The FEA simulation results produced by LS-DYNA are saved in the output files $State_1$ to $State_n$. The importer is implemented in C++ as a 3ds Max plug-in, taking advantage of the open software architecture of commercial animation systems. The importer loads, interpolates, and translates the simulation results into n x m 3ds Max scenes. The variable m allows "slowing" down the simulation in the visualization by creating intermediate frames by interpolation. At most one frame is created from each scene, and some scenes can be skipped. In the example in Figure 14 the visualization rendered progresses three times faster than the slowest pace possible. The 3ds Max scene files are created only once and then reused to produce visualizations with a variety of speeds, camera angles, effects, and materials. The visualization rendering is controlled by a 3ds Max script that loads, sets, and renders the appropriate 3ds Max scene for each visualization frame.

Visualizations were rendered on two Intel Dual-Core Xeon workstations. One workstation has 16GB of RAM and 2.6GHz CPUs, while the other has only 4GB of RAM with faster 3.2GHz CPUs. Each workstation has 2 CPUs, each CPU has 2 cores, and each core runs two hardware threads, for a total of 16 threads. One instance of the renderer uses 1 or 2 threads for simple scenes

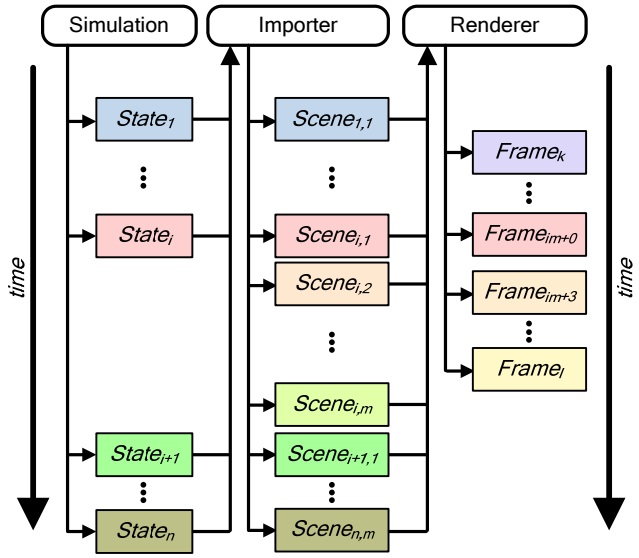

Figure 14. Implementation overview.

without visual effects (i.e. no dust, debris, or fire) and without fuel meshes, 4 threads for scenes with visual effects, and 8 threads for scenes that require ray tracing fuel meshes. The visualization frame rendering times vary from 2-3 minutes for scenes without visual effects and without jet fuel, to 3-5 minutes for scenes with visual effects, and to 5-60 minutes for scenes with fuel meshes (depending on the degree of dispersion of the jet fuel).

The geometry load was on average 1.2 million triangles for the geometry resulting from the beam, shell, and solid elements, plus the triangles that modeled the fuel meshes. As expected, the geometric complexity of the fuel meshes increased considerably as the fuel dispersed. Fuel meshes totaled only 5,000 triangles at time 0.0s when the liquid was concentrated in the fuel tanks, 71,500 triangles at 0.125s, 148,130 at 0.25s, 715,500 at 0.5s and over 1.3 million triangles at 0.75s. Even so, the total number of triangles does not exceed 3 million.

7 Conclusions

Federating a visualization system with FEA simulation is a powerful approach to communicating the results of simulations, especially when the user is not a domain expert. High visual impact is accomplished without sacrificing physical reality. Excepted from this assertion are fire and explosion details, which as chaotic processes, cannot be modeled to duplicate the actual event. Nevertheless, the similarity between the simulated damage to the façade and the observed damage increases confidence that the simulated core damage is likely to be accurate.

From the technical perspective, this type of visualization is also helpful in uncovering errors of detail, in the meshing as well as in the orientation of elements. For instance, since traditional post-processors do not employ sophisticated lighting computations, subtle errors such as incorrect normals can go unnoticed.

One of the shortcomings of the simulation is that the dispersing fuel is treated by LS-DYNA as a non-volatile liquid. However, it created an explosion and subsequent fire. Thus, SPH elements should have a death frame associated that attenuates the mass of liquid over time. This and other fire-related effects should be revisited in future work.

The Pentagon and the WTC projects demanded a largely orthogonal set of importer features. Most LS-DYNA output is now supported by the importer. The importer has a modular architecture consisting of an LS-DYNA output file parser, the translator which is independent of LS-DYNA and 3ds Max, and a scene instantiation module. This should facilitate deriving importers that couple other pairs of simulation/animation systems, by reusing the translator module.

Another line of future work is to develop a translation that supports interactive visualization. The geometry load is easily manageable by a modern graphics card, so a single state could, in principle, be examined interactively after simplifying the material and lighting models. The challenge that remains to be overcome is handling the massive animation data that specifies the positions of millions of vertices independently for each frame.

Our work does *not* advocate stripping post-processors of all visualization capability. Post-processors will continue to play their role as a rapid inspection tool, catering to the domain experts that design and run simulations. However, the post-processor should not be expected to produce high-quality visualizations based on state-of-the-art rendering algorithms. Such visualization tasks should be simply outsourced to animation systems through general and scalable importers.

8 Acknowledgments

We would like to thank Profs. Sozen and Pujol from Purdue's Civil Engineering department who helped with the simulation and calibration of its models and parameters. Ingo Brachmann and Oscar Ardila-Giraldo constructed the models for the WTC-I. Tom Miller and Joe Farris helped with 3ds Max and AfterBurn. Scott Meador provided great suggestions regarding visualization in 3ds Max. The work was supported in part by NSF, DOE, the Tellabs Foundation, and by Purdue's Rosen Center for Advanced Computing. Some of the runs were done using NCN's Regatta and this support is also gratefully acknowledged.

References

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

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

[3] GoogleEarth, URL: http://earth.google.com

[4] V. Popescu, C. Hoffmann, S. Kilic, M. Sozen, S. Meador, "Producing High-Quality Visualizations of Large-Scale Simulations," *Proceedings of IEEE Visualization*, Oct., 2003.

[5] M. Pauline Baker, Dave Bock, Randy Heiland, and Michael Stephens. "Visualization of Damaged Structures", U.S. Army Corps of Engineers Waterways Experiment Station, Technical Report, 1998.

[6] 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.

[7] The Visualization Toolkit, URL: http://public.kitware.com/VTK/

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

[9] "World Trade Center Building Performance Study: Data Collection, Preliminary Observations, and Recommendations", Federal Emergency Management Agency, FEMA 403, May 2002.