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

Paul Rosen, Voicu Popescu, Christoph Hoffmann, and Ayhan Irfanoglu

Abstract—In this application paper we describe the efforts of a multi-disciplinary team towards producing a visualization of the September 11 Attack on the North Tower of New York's World Trade Center. The visualization was designed to meet two requirements. 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 aircraft 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. We built upon a previously developed translator. The translator was substantially extended to enable and control visualization of fire and of disintegrating elements, to better scale with the number of nodes and number of states, to handle beam elements with complex profiles, and to handle smoothed particle hydrodynamics liquid representation. The resulting translator is a powerful automatic and scalable tool for high-quality visualization of FEA results.

Index Terms—I.3.4 Three-Dimensional Graphics and Realism, I.3.8 Applications.

INTRODUCTION

VISUALIZATION has long been recognized by experts in a variety of domains as an indispensable tool. Visualization systems are typically developed in collaboration with domain experts and achieve—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.

This is a serious limitation when the interest in the visualization transcends a single domain, as is the case, for example, for computer simulations. Simulation codes and computing hardware are now sufficiently powerful to enable high-fidelity simulations that track in detail complex interactions in large scenes. 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 to produce 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 the 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 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, smoothed particle hydrodynamics (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).

- P. Rosen is with the Computer Science Department, Purdue University, West-Lafayette, IN 47907. Email: rosen@cs.purdue.edu.
- V. Popescu is with the Computer Science Department, Purdue University, West-Lafayette, IN 47907. Email: popescu@cs.purdue.edu.
- C. Hoffmann is with the Computer Science Department, Purdue University, West-Lafayette, IN 47907. Email: cmh@cs.purdue.edu.
- A. Irfanoglu is with the Civil Engineering Department, Purdue University, West-Lafayette, IN 47907. Email: ayhan@ecn.purdue.edu.

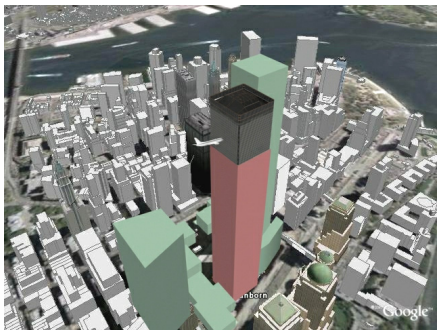


Figure 1. Simulation integrated into lower Manhattan scene using Google Earth.

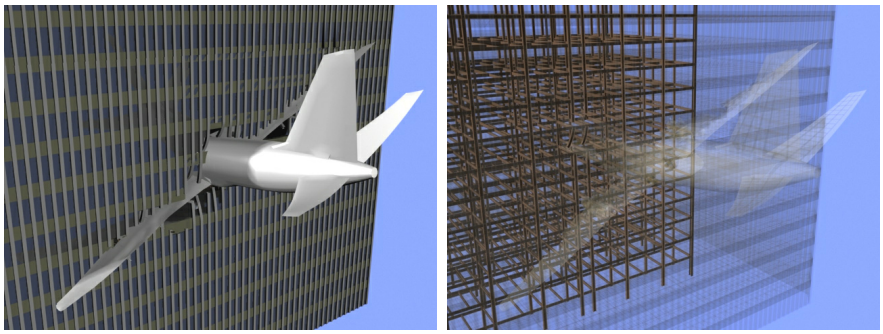


Figure 2. Impact visualization from outside the building. Right image highlights core column damage by rendering all other elements with transparent materials.

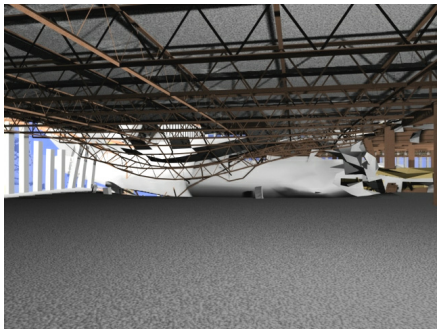


Figure 3. Region between façade and building core, during (*left*) and after (*right*) impact.

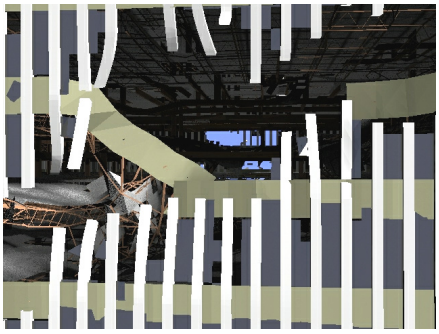


Figure 4. Visualization of façade breach.

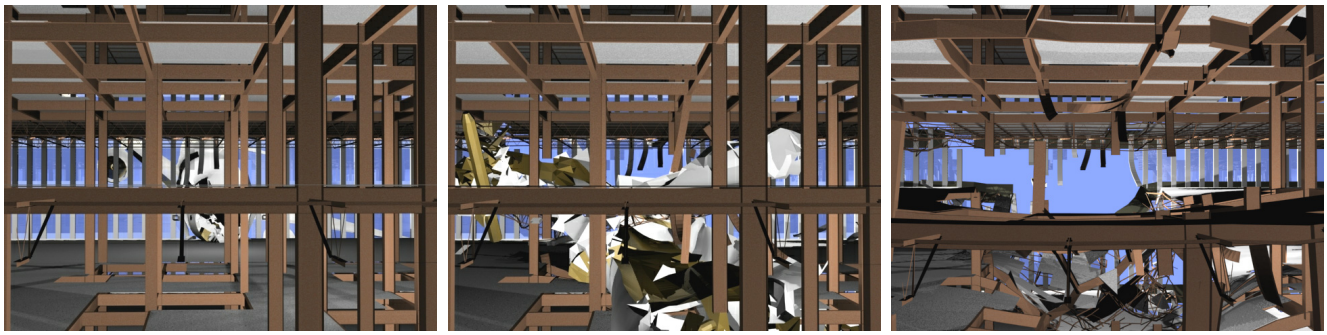


Figure 5. Visualization of the two floors most affected by the impact, chronologically, from left to right.

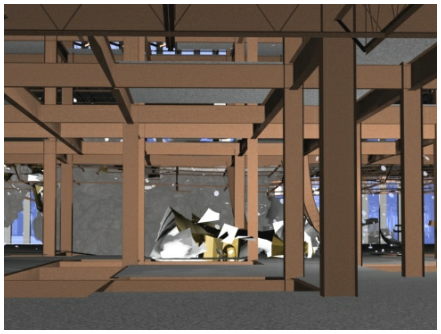


Figure 6. Dust and glass debris generated automatically from eroding elements.

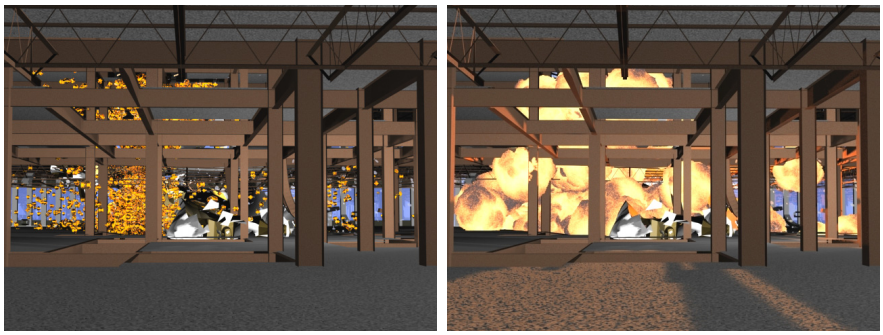


Figure 7. Fire visualization (*right*) generated automatically from SPH simulated jet fuel (*left*). The shape of the fire follows the particle distribution.

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 inserting 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 there-

fore 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 render the actual beam profile. 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 visualization produced succeeded beyond our expectations at communicating the simulation results to the public at large, as attested by the over two million downloads so far, by the hundreds of mass media accounts of our work, and by the request from the September 11 National Memorial & Museum to permanently exhibit our visualization as part of the main narrative of the future museum. We believe that our visualization approach also has great potential in the context of other application domains such as law enforcement, emergency response, and defense.

The translator is based on a LS-DYNA to 3ds Max translator we have originally developed for producing a visualization of the September 11 Attack on the Pentagon [4-6]. The original translator is described in detail in Section 0 which discusses prior work. We have substantially extended the translator to support:

- SPH liquid simulation,
- automatic fire visualization controlled by SPH elements,
- automatic dust and debris visualization controlled by eroding elements,
- beam elements with complex profiles,
- out-of-core per-vertex animation.

These additions are essential for the application at hand and complete the translator. Simply reusing the translator developed for the Pentagon in the new context would have produced a crude visualization:

- without the jet fuel since the Pentagon simulation modeled the fuel with an Arbitrary Lagrange-Eulerian (ALE) mesh,
- with all beams shown with circular cross-sections, regardless of their true I, T, or C cross-sections, since the only beams in the Pentagon simulation were rebars with circular cross-section,
- with drastically approximated deformations and trajectories as required to make the earlier in-core vertex animation tractable,
- and without fire and debris visualization.

This paper is primarily an application paper as it presents the successful application of prior graphics techniques to the purpose of visualizing a simulation of the 9/11 Attacks. However, the paper also describes a system, which can be employed in many other scenarios. The now complete translator is a powerful, scalable, and general tool suitable for visualizing FEA simulations in general. 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 approach of automatically translating data to be visualized in forms that can be handled by state-of-the-art animation systems could and should be applied to contexts beyond visualization of simulation data. First, translation is certainly simpler than replicating functionality. Second, the application will benefit from the frequent and substantial advances of animation systems, often for free, and occasionally at the small cost of updates to the translator. Third, the visualization researcher and the domain experts can focus from day one of their collaboration on researching novel visualization algorithms that *advance* the state of the art rather than on replicating it.

The remainder of the paper is organized as follows. Prior work is discussed next. The FEA simulation is described in Section 0. Section 0 describes the translation of the simulation data into an animation system 3D scene suitable for visualization. Section 0 describes using the animation system's resources to produce the visualization. Section 0 gives an overview of the implementation, and Section 0 provides conclusions and sketches possible directions of future work.

PRIOR WORK

An earlier effort with similar goals to ours was the simulation of the impact of a bomb detonation on nearby buildings [7]. 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 [8] and using the Visualization Tool Kit (VTK) [9]. The researchers mention enhancing the visualization with photographs as future work 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

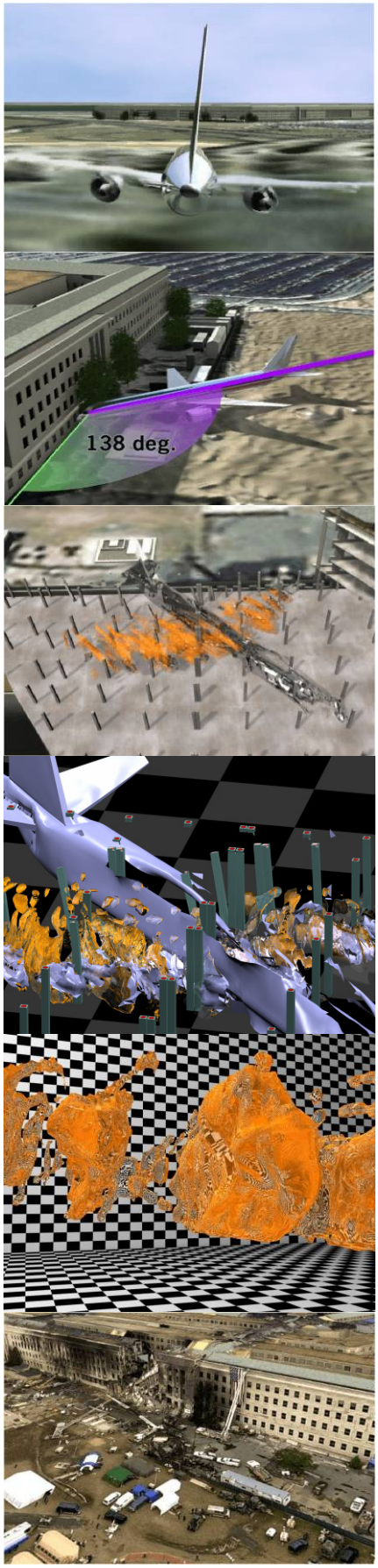


Figure 8. Snapshots from the Pentagon visualization.

building codes for nuclear containment structures. A full-scale experiment with an actual fighter aircraft was conducted by Sugano et al. [10]. The experiment provided impact force/deflection measurements used to validate subsequent simulations.

Simulation of the 9/11 Pentagon Attack

In a previous effort, our team produced a visualization of the 9/11 Attack on the Pentagon [4-6]. 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. The aircraft FEA model included detailed models of the fuel tanks, which concentrated most of the kinetic energy of the aircraft. The jet fuel was modeled with an ALE mesh. Unlike in the case of the World Trade Center where the buildings tragically collapsed eliminating the possibility of a precise measurement of the damage caused by the impact, in the case of the Pentagon, the effect on the impact on the structural columns was precisely recorded by the structural engineers that inspected the building shortly after the attacks. The simulation computed a column destruction pattern similar to the one observed.

The visualization was produced by importing the LS-DYNA simulation results into 3ds Max through a custom plug-in that translates the simulation data into an animation scene. Solid objects represented in the FEA with hexahedral elements were converted into triangle meshes after removing the internal faces to reduce the geometry load. Fractures can expose initially hidden faces, so the simplification had to consider the entire simulation timeline.

Thin surfaces represented with quadrilateral shell elements were converted into triangle meshes straightforwardly. The Pentagon scene had only one type of beam elements: beam elements with a circular profile used to model the column rebars. These beam elements were translated into animation scene splines which allowed for an easy control of the thickness and level of tessellation.

The ALE mesh representing the liquid is a set of hexahedral elements decorated with fractional occupancy values. The ALE mesh was translated into surface boundary representation of the liquid in three steps. First, the occupied ALE mesh elements were selected by thresholding with a user-specified minimum fractional occupancy value. Then the internal faces were eliminiated as described above to obtain a coarse liquid mesh still suffering from the blockiness inherited from the ALE mesh. In a final step, the coarse liquid mesh is refined by applying 3ds Max geometry modifiers to obtain a realistic liquid visualization.

Although the geometry load did not pose problems for the animation system, handling the per-vertex position data proved to be more challenging. In order to animate the geometry in the animation system according to the deformations computed by the simulation, the geometry

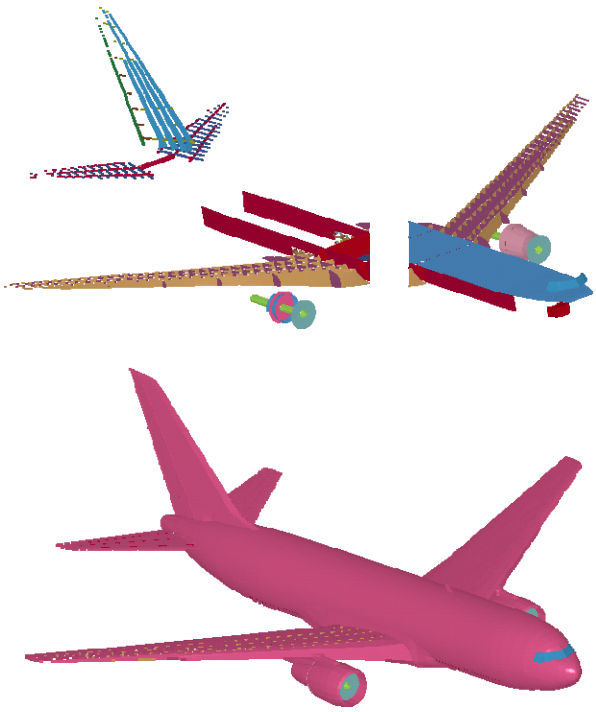


Figure 9. Layers of the aircraft finite element model.

vertices were automatically assigned position controllers based on the simulation output. A position controller defines the position of a vertex at a given animation time step. In order to reduce the total number of controllers, both non-lossy and lossy simplification schemes were employed to eliminate intermediate position controllers for vertices moving on or approximately on a straight line. Even so, the large number of position controllers required remained the most severe bottleneck in the translator.

Once the simulation data was imported into the animation system, a high-quality general purpose visualization was produced by leveraging the animation system's sophisticated material editors and state-of-the-art rendering algorithms.

An important goal of the project was to integrate the simulation within the immediate surroundings of the scene in order to provide context. This is particularly important since for simulation purposes the Pentagon was reduced to its most relevant structural components and without context it is difficult to grasp which part of the Pentagon building was affected, how extensive the damage was relative to the entire building, and what trajectory the plane had before the impact. The Pentagon building was modeled from blueprints, the Pentagon grounds were modeled with a simple plane, and realism was achieved efficiently and effectively by projective texture mapping with high resolution satellite and aerial photography. The animation system also served the role of integration platform for combining the simulation data and the surrounding scene.

Figure 8 shows the approach scene (images 1 and 2 from the top, computer graphics (CG) only), the simulation data integrated into the surrounding scene (image 3, simulation & CG), the simulation data shown in isolation (images 4 and 5, simulation only), and the scene after the impact (image 6, CG only). 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 WTC visualization effort not only involved a different scenario, but also required major extensions of the translator. One extension added automatic, simulation-controlled fire, dust, and debris visualization. Another extension added support for SPH liquid representation, the second of the two liquid representations commonly used in simulations. A third extension enhanced the beam visualization capability to handle beam elements with complex profiles. Last but not least, the position controller bottleneck was eliminated by moving to an out-of-core per-vertex animation approach, which achieves great scalability with the complexity of the FEA models and the number of simulation states. As stated in the introduction, simply reusing the translator developed for the Pentagon simulation in the present context of the WTC simulation would have produced a crude visualization, unsuitable for describing the WTC events to the general public in detail.

Another difference between the present and the previous work is that for the present effort a 3D model of the surrounding scene was readily available through Google Earth.

FEA Simulation

We modeled a Boeing 767-200 using both graphics models as well as published aircraft literature. The FEA model (Figure 9) 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 10). 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, and 87,188 SPH, 248,433 beam, 93,733 shell, and 674 solid elements. The solid elements were used to model the titanium shafts of the two engines and the titanium undercarriage. 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.

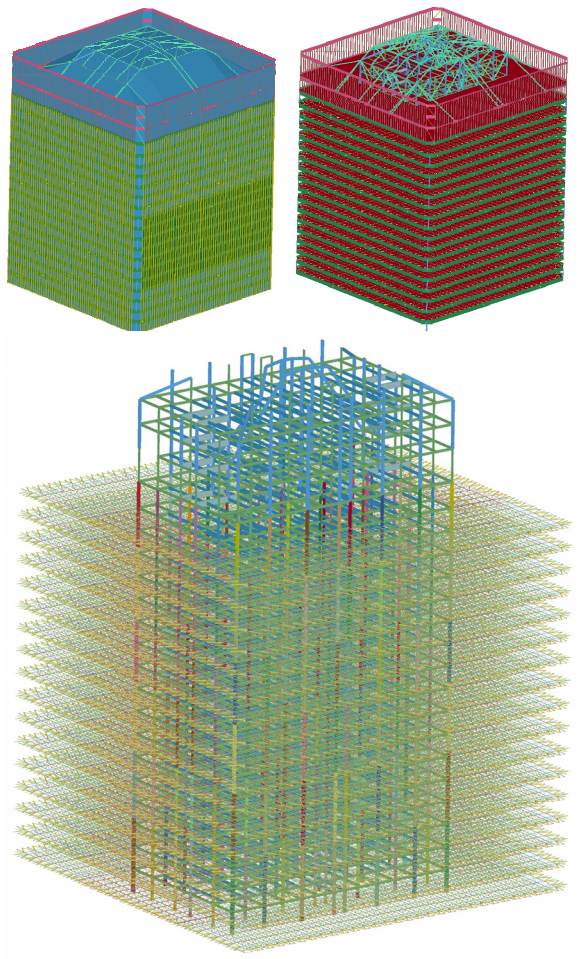


Figure 10. Layers of the WTC-I finite element model.

The façade damage computed by the simulation (Figure 11) is in agreement with the observed damage [11]. 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. For additional details we refer the reader to our civil engineering report [12].

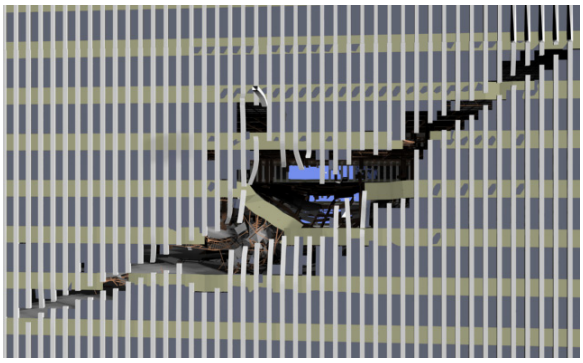


Figure 11. Façade damage visualization.

FEA To Animation Translation

In this section we describe how the various types of FEA elements are translated into animation system geometry and how they are animated according to the simulation. Quadrilateral shell and hexahedral solid elements are imported and translated into 2 and 12 faces [4].

Beam Elements

The FEA models contain beams with L, square, I, and T profiles (Figure 12). The simulation represents a beam element with two nodes and a normal, from which the translator 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 added at the cost of a twofold increase in geometry complexity.

Element connectivity is not encoded in the simulation output files. The translator 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.

The geometry resulting from the shell, solid, and beam elements is automatically filtered through a 3ds Max 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).

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 13).

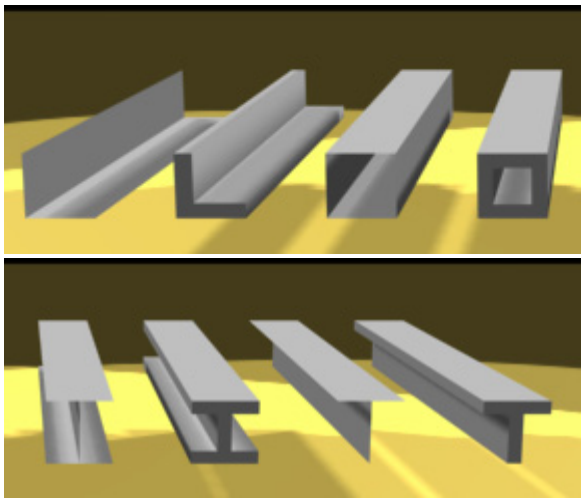


Figure 12. Beam profile types: L, square (*top*), I and T (*bottom*), thin and thick versions.

Out-of-Core Per-Vertex 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. As previously mentioned, 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 into 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 translator, and then renders the frame. Typically the visualization examines the transition between two simulation states over several frames. This is achieved by interpolating the simulation data. Only two states are loaded into memory at any time. No position controllers are needed since there is an animation scene file for every frame. This out-of-core approach removes the position controller bottleneck and scales well with the complexity of the FEA models and the number of simulation states.

The reader might wonder whether a linear interpolation does not introduce excessive errors and whether more sophisticated interpolation schemes should not be preferred. Although FE nodes (and thus vertices) do not move on a straight line with constant speed between states, the visualization module is ill-equipped and is not responsible for tracking the nodes with physical accuracy. The visualization simply reflects the simulation data encoded in the set of states provided as input.

The simulation data is saved with sufficiently small time steps such that the sum of all the states captures the simulation well. In our case, the civil engineering researchers found that saving a state every 2.5 milliseconds provides ample time resolution to faithfully record the simulation. Note that the numerical codes use a much smaller (and adaptive) integration time step (i.e. 6.5-6.65 microseconds), so the interval between the states does not affect the simulation accuracy. The LS-DYNA post-processor simply shows the states with abrupt transitions between consecutive states. We simply allow the user to slow down the visualization timeline while maintaining the fluidity of the visualization.

VISUALIZATION

Once the animation system geometry corresponding to the simulated entities is created, one can leverage the comprehensive arsenal of material, light, and camera libraries and editors available in the state-of-the-art animation system. Once created, the materials are assigned automatically by the script that generates the animation scene for each frame. For example the liquid shown in Figure 13 was rendered with a complex ray-traceable material with depth-depending opacity, refraction, and surface reflection.

Although the raw power of computing hardware and the sophistication of simulation codes have increased tremendously, the simulation node and element budgets remain finite. Strategies that make best use of available resources by allotting them to entities most relevant to the simulation will always be preferred to brute force approaches. Relevance to *simulation* does not always imply and is not always implied by relevance to *visualization*, which means that some approximations acceptable from the simulation stand point lead to a substantial loss of visualization fidelity. The visualization module is called upon to alleviate this discrepancy.

Simulation-controlled visualization effects

A first example is the addition of effects with great visualization relevance that were ignored by the simulation, such as debris, dust, and fire.

When an element undergoes excessive stress, the simulation code marks it as eroded, which effectively eliminates the element from subsequent computations. From the simulation's standpoint this is a reasonable approximation in many cases such as that of glass elements that shatter into small glass fragments or of concrete elements that pulverize into dust. The impact of the resulting fragments is small relative to the high cost of tracking them.

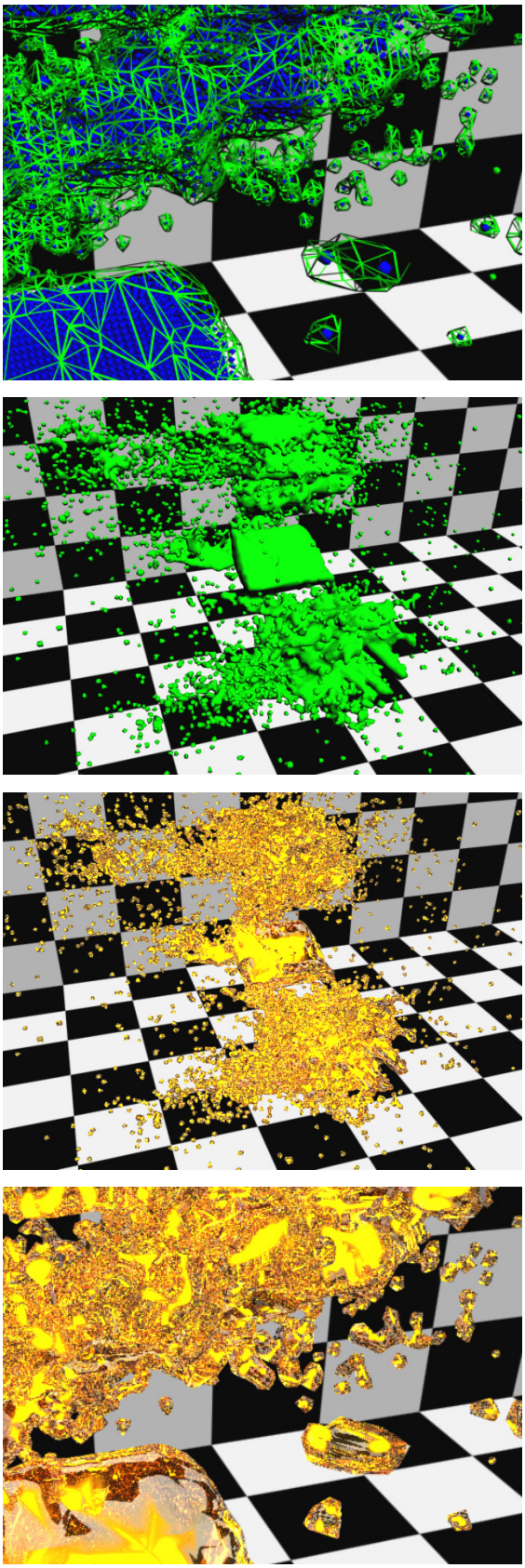


Figure 13. From the top: close-up visualization of conversion of SPH-element spheres (*reduced size, blue*) into fuel mesh (*green*), diffuse fuel mesh, ray traced fuel mesh, and ray traced fuel mesh close-up.

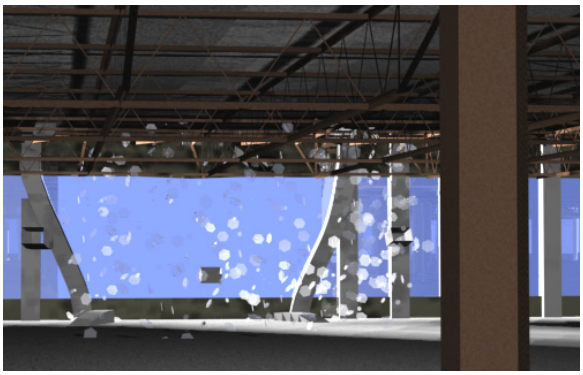


Figure 14. Glass debris effect.

Our simulation did not take into account the ensuing fire, which is acceptable considering that the main civil engineering task for the simulation was to estimate the damage to the core columns as a result of the impact. The one second of real time covered by our simulation was most likely not sufficient for the fire to raise the temperature of the core columns sufficiently in order to substantially modify their load bearing capacity.

However, even though debris, dust, and fire have a limited importance for the simulation of the one second impact, these effects have an important visual contribution and should therefore be added to a version of the visualization that strives to match what was seen during the actual events. In addition to helping fulfill the simulation's mission of documenting the events, the dust, debris, and fire ensuing from the impact could be the starting point for additional simulations that cover the minutes that followed the impact. Such additional simulations could answer important questions regarding visibility, availability of evacuation and rescue paths, and survivability.

While it is certainly possible to employ a skilled user of the animation system in order to decorate the visualization with the missing effects, such an approach suffers of two important disadvantages. First, the approach is time consuming due to the manual effort involved. In the typical context of multiple simulation runs systematically investigating a range of initial condition values, any manul effort in the visualization of the deluge of simulation data is prohibitively expensive. Second, the animator's involvement introduces subjectivity that might be undesirable.

We have developed a method for adding visual effects automatically, based on the simulation data, thus bypassing the inefficiency and subjectivity disadvantages. The translator generates a pseudomesh from the simulation data and a script that uses the pseudomesh during rendering to control visual effects such as erosion (i.e. glass debris and dust), and fire.

In the case of erosion, the pseudomesh vertices are created by the translator by down-sampling the set of nodes of the elements that erode (i.e. glass shell elements for the glass debris, and other eroding elements for dust). The erosion pseudomesh has a birth frame and a death frame specified by the translator. In the case of fire, the pseu-

domesh vertices are derived from the SPH nodes and the mesh persists throughout the simulation.

The translator also generates a script which creates a particle array using the pseudomesh 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. Wind and gravity implemented as space warpers 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.

The result is a complex visualization with plausible debris, dust and fire effects, created automatically from the simulation data. The shape of the fire follows the distribution of SPH elements modeling the fuel (Figure 7), the glass debris is generated by the disintegrating glass panes of the façade (Figure 14), and the dust clouds are created by eroding concrete and beam elements (Figure 6).

Integration into surrounding scene

A second example of post-simulation contribution to the visualization is placing the simulation results in the context of the surrounding scene. Typically the simulation budget is entirely allotted to the entities most actively involved in the simulated event. Limiting the visualization to the simulated entities prevents the effective dissemination of the simulation results. Modeling the surrounding scene and combining it with the simulation results leads to a powerful visualization suitable to non-expert users.

In the case of our previous work on simulating the Attack on the Pentagon, the simulation of the impact between the airframe and the building columns was placed in the context of a complete computer graphics model of the Pentagon building and a ground plane textured with satellite and aerial imagery [4]. This enabled an easy-to-understand visualization of the scene before, during, and after the impact.

In the case of the present work, we relied on Google Earth [3] to produce a zoom-in video sequence from planetary to neighborhood scale (see accompanying video and Figure 1). In order to show the simulation in the context of lower Manhattan we modeled the WTC buildings, including the bottom floors of the WTC-I tower not involved in the simulation, and imported them into Google Earth. Once an appropriate view was selected in Google Earth, the view parameters and the corresponding image were used in the animation system to define a camera and a matching background image, respectively. The camera and the matching background allowed visualizing the simulation in the context of the lower Manhattan scene.

IMPLEMENTATION

The visualization is produced according to the pipeline depicted in Figure 15. The FEA simulation results produced by LS-DYNA are saved in the output files $State_1$ to $State_n$. The translator is implemented in C++ as a 3ds Max plug-in, taking advantage of the open software architecture of the commercial animation system. The plug-in

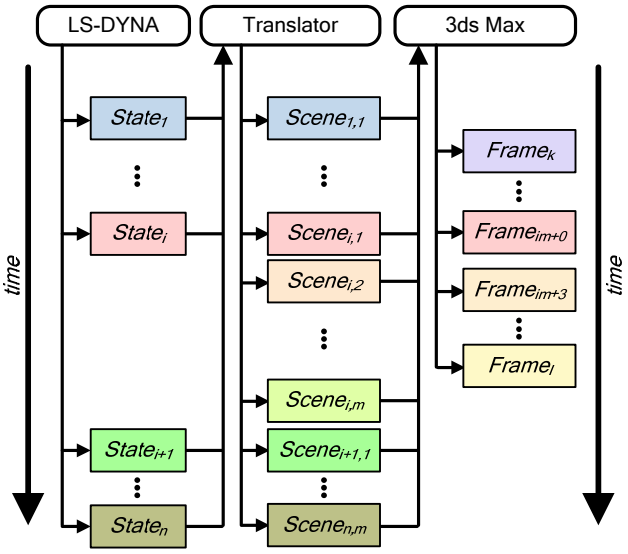


Figure 15. Implementation overview.

code, user manual, and examples are available on the project's website [15]. The translator 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 15 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 2 hardware threads, for a total of 16 threads. One instance of the renderer uses 1 or 2 threads for simple scenes 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).

Figure 16 gives the number of triangles for the visualization scene according to the simulation state (1 state corresponds to 2.5ms real time). Whereas the number of triangles visualizing the beam, shell, and solid elements remains relatively constant (see "non-liquid" series in the graph), the geometric complexity of the fuel meshes increases considerably as the fuel disperses ("liquid" series). Even so, the total number of triangles does not exceed 3 million for the first 300 states.

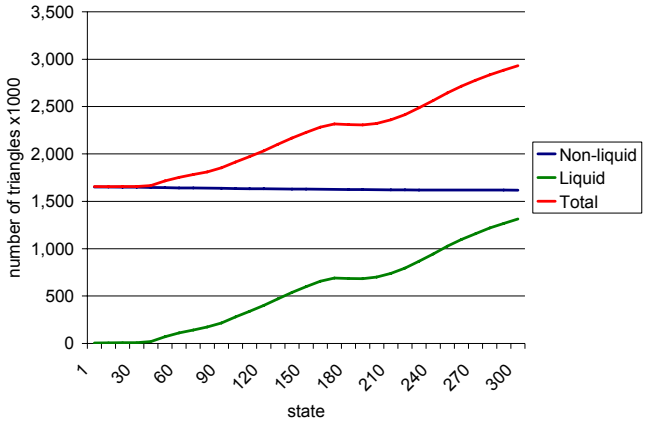


Figure 16. Variation of the geometric complexity of the visualization scene with the simulation state.

CONCLUSIONS

Federating state-of-the-art animation and FEA simulation systems is a powerful approach to communicating the results of simulations, especially when the user is not a domain expert. The initial goal of achieving visual fidelity without sacrificing physical fidelity has been reached.

We derive confidence in the physical fidelity of our visualization from the fact that it was computed directly from the simulation data. Confidence in the simulation itself is derived from the fact that the simulation was calibrated based on experimental data, from a general agreement with core damage estimates computed by other groups [13], and from the similarity between the computed and actual façade damage (see our description of the simulation from an engineering perspective [12]).

The images throughout this paper and the accompanying video attest to the quality of the visualization and its accessibility to the non-expert viewer. The visualization has been downloaded over 3 million times, it has been relayed by hundreds of mass media outlets, and it has been requested by the National September 11 Memorial & Museum for permanent exhibition.

Visualization is a tool that has been long used to uncover simulation errors. Although the goal of our visualization mode is suitability for general audiences and not debugging, our visualization helped the civil engineers on our team uncover a problem with the finite element mesh. Figure 17 shows how the visualization revealed incorrect normals for some of the façade beam elements, which led to an incorrect attachment to the beam elements below. Since the post-processor either shows the beams as disconnected elements with the true profile or as connected segments (i.e. wire frame), the incorrect normals cannot be detected with the post-processor. This anecdotal evidence leads us to believe that the use of visualization as a simulation debugger can be further improved, tracing a possible future research path.

The Pentagon and the WTC projects demanded a largely orthogonal set of translator features. Visualizing the WTC simulation with the earlier version of the translator would have been challenging and the result crude. The added features have proven to be important when the visualization user is the general public and we foresee that they will be important for future users from domains such as defence and emergency response. Most LS-DYNA output is now supported. The translator 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 translators that couple other pairs of simulation/animation systems, by reusing the translator.

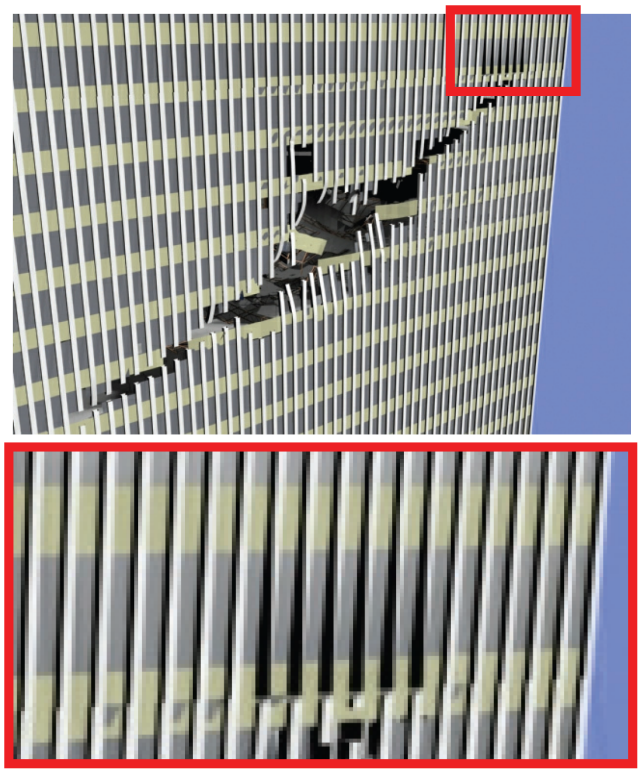


Figure 17. FEA model error uncovered by the visualization (*top*) and magnified frame fragment (*bottom*).

In the current implementation the resolution of the visualization is defined by the resolution of the simulation. The size of the triangles modeling the surfaces of the interacting entities is given by the size of the underlying finite elements. Since simulation codes scale more poorly with the number of nodes than animation systems with the number of triangles, we foresee that the geometry processing capability of the animation system will remain underutilized. We will explore ways of fully exploiting the rendering capability by decoupling the resolution of the visualization from the resolution of the simulation.

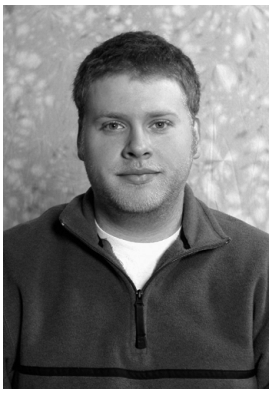
Our work does *not* advocate stripping post-processors of all visualization capability. Post-processors will continue to play their role as a rapid inspection tool, catering to the domain experts that design and run simulations. However, the post-processor should not be expected to produce high-quality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

ACKNOWLEDGMENTS

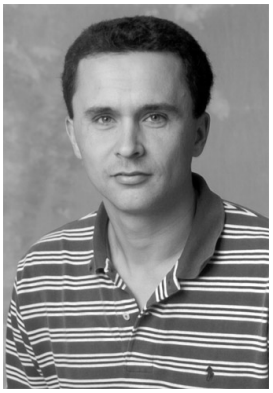
We would like to thank Profs. Sozen and Pujol from Purdue's Civil Engineering department who helped with the simulation and calibration of its models and parameters. Ingo Brachmann and Oscar Ardila-Giraldo constructed the models for the WTC-I. Tom Miller and Joe Farris helped with 3ds Max and AfterBurn. Scott Meador provided great suggestions regarding visualization in 3ds Max. We are grateful to Steven Tally who authored the press release that instantly triggered widespread and intense interest in our work. This 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 Purdue's Nanotechnology Computation Network'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] C. Hoffmann, V. Popescu, S. Kilic, M. Sozen, "The Pentagon on September 11th," *IEEE Computing in Science and Engineering*, pp 52-60, Jan., 2004.
[6] V. Popescu, C. Hoffmann, "Fidelity in Visualizing Large-Scale Simulations," *Journal of Computer Aided Design*, Elsevier, 37, pp 99-107, 2005.
[7] 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.
[8] 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.
[9] The Visualization Toolkit, URL: http://public.kitware.com/VTK
[10] T. Sugano et al., "Full-scale aircraft impact test for evaluation of impact force", Nuclear Engineering and Design, Vol. 140, 373-385, 1993.
[11] "World Trade Center Building Performance Study: Data Collection, Preliminary Observations, and Recommendations", Federal Emergency Management Agency, FEMA 403, May 2002.
[12] A. Irfanoglu and C. Hoffmann. "An Egineering Perspective of the Collapse of WTC-I". Journal of Performance of Constructed Facilities, ASCE, 06/2007, in press, available at URL: http://cobweb.ecn.purdue.edu/~ayhan.
[13] National Institute of Standards and Technology (NIST). "Final Report of the National Construction Safety Team on the Collapses of the World Trade Center Towers". NIST NCSTAR 1, Gaithersburg, MD, 2005.
[14] Youtube, URL: http://www.youtube.com.
[15] P. Rosen and V. Popescu, "3ds max importer plug-in for LS-DYNA data; code, user manual, and simulation examples", http://www.cs.purdue.edu/cgvlab/projects/dynaPlugin.htm



Paul Rosen is a Ph.D. candidate at the Computer Science Department of Purdue University. His research interests include high-fidelity visualization of simulation data, camera model design, and shape perception study using 3D images.



Voicu Popescu is an associate professor of computer science at Purdue University. He received a Ph.D. degree in computer science from the University of North Carolina at Chapel Hill in 2001. His research interests span visualization, computer graphics and computer vision, and include high-fidelity visualization of large scale simulations, 3D scene acquisition, camera model design, effective distance learning, and shape perception study using 3D images.

Christoph Hoffmann is a professor of computer science and the director of the Rosen Center for Advanced Computing at Purdue University. His research interests include computer-aided design, high performance computer simulation, design system architectures, geometric and solid modeling, geometric constraint solving, and grid computing. He is a member of the editorial boards of the journals of Computer Aided Design, of Computer-Aided Geometric Design, and of ACM's Transactions on Computer Graphics.

Ayhan Irfanoglu is an assistant professor of civil engineering at Purdue University. He received a Ph.D. from the California Institute of Technology in 2000. His research interests include large-scale structural analysis and simulation, earthquake and blast engineering, structural health monitoring, and engineering seismology.