Fidelity in Visualizing Large-Scale Simulations

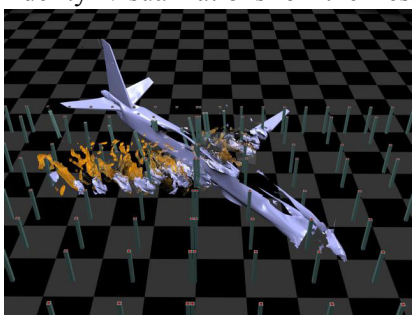
Voicu Popescu and Christoph Hoffmann

Computer Science, Purdue University
West Lafayette, IN 47907, USA

Abstract

Computer simulations are powerful tools frequently used today in many important applications, for example to build safer buildings, to crash-test an automobile before it is built, to stabilize the Pisa tower, to design artificial joints that are comfortable and durable, or to investigate what-if scenarios to avoid and best recover from natural or man-made disasters. The simulation codes have reached a very high-level of sophistication and, by running on powerful computing machinery, can accurately track with infinitesimal time steps dozens of physical properties of millions of interacting elements under extreme conditions. In order to take fully advantage of the bounty of information concealed in the data produced, visualization is a uniquely powerful tool since it caters to the sense that provides our highest bandwidth connection to the surrounding world.

Unfortunately, simulation results are usually examined with graphics and visualization tools that are one or several steps behind the state-of-the-art. We describe our efforts of producing high-fidelity visualizations of the results of large-scale simulations using the latest commercial rendering and animation systems. To this effect we built a scalable and reusable link between the software worlds of animation and simulation. Our system also offers a set of tools that allow integrating the results of the simulation in the surrounding scene, of great importance when the intended audience extends beyond the researchers that designed the simulation. We built our system as part of the efforts of a larger, interdisciplinary team to produce a high-quality, physically accurate visualization of the September 11 attack on the Pentagon.



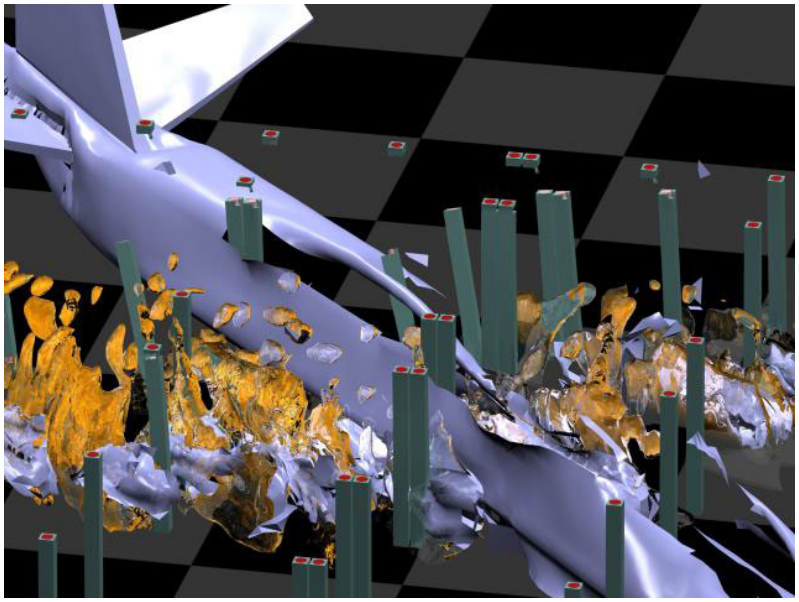


Figure 1 Simulation data visualization using our system.

Introduction and Motivation

Physical simulations have become an important tool to understand and analyze critical events, such as an airplane crash or an earth quake, and, routinely, to analyze the performance of devices. Using finite element analysis (FEA), several commercial programs offer sophisticated tools for evaluating and simulating physical events. By measuring specific

quantities such as stress and strain, or simply by scrutinizing the unfolding of an event in small time steps, important insights can be had that illuminate specific strengths and weaknesses of structures and mechanical systems, as well as provide clues on how to improve the performance of engineering designs.

Much work has been done to simplify and automate formulating an FEA problem from a given shape design, particularly in the context of discrete manufacturing. Furthermore, considering only the geometric aspect of this problem, geometric mesh generation addresses the key problem of converting a boundary-based shape representation, familiar from computer-aided design (CAD), into an equivalent point- and element-based representation suitable for describing the shapes to be analyzed by the FEA, as well as representing ambient space for flow problems. This work has migrated into many commercial software packages, such as Ansys, Fluent, Elfini, to name a few. Once the FEA has completed, the results of the analysis are then visualized using a variety of tools geared to the FEA representation. These standard visualization tools are adequate to show salient features of the analysis in a manner that is familiar to engineers and specialists, and are well-suited to discrete manufacturing.

For communicating the results of a FEA to a public audience, however, the traditional way of visualizing is not optimal. When evaluating an abnormal event such as a plane crash, public and policy interests are also in play, and properly communicating the event to the wider audience suggests that the visualization of the results should be ready to be grasped by people who are unfamiliar with, e.g., false color schemes, and other traditional tools for accentuating engineering quantities. Moreover, a realistic and engaging visualization of the simulation results is useful as well for integration into virtual reality training and documentary presentation.

In this paper, we address how to visualize with high fidelity FEA results for a forensic simulation of an event by combining the physical quantities and events computed by the FEA with real-world, natural environments, or even synthetic environments familiar form computer games. In particular, we discuss how to convert the FEA representation to ones suitable for commercial animation systems, how to add special effects, and how to situate the results in the surrounding, natural environment. Some aspects of integrating visualization with simulation have been discussed in [2]. and have focused on the overall problems of managing simulation complexity and federating commercial FEA systems with commercial visualization and animation systems.

We discuss our subject in the context of a case study, the September 11 attack on the Pentagon. One of our collaborators at Purdue University had been invited to participate in the forensic study of the damage the Pentagon building sustained in that attack. To better understand the performance of the building, eventually reported in [1], several simulation studies were done in 2002 using the commercial FEA code LS-Dyna [7, 8]. The simulations culminated in two runs on an IBM Regatta supercomputer, and the data so obtained was first analyzed using the standard post-processor of LS-Dyna. See [3], Phase I, for the visualizations done using traditional post-processing. It was clear at the time that the visualizations so obtained were interesting, but were inferior to visualizations common in computer graphics. This motivated us to investigate how to achieve high fidelity in visualizing the simulation using 3DS Max [21], a commercial rendering and animation software package.

Prior Work

Large-scale simulations have become an accepted research tool in science and engineering. They require accurate models of the physical properties under considerations and appropriate geometric details. Accurate material models and algorithms are devised and validated from experiments and theoretical considerations. They are discussed in text books and major sections of the literature. Of equal importance, but not as well known, is how to achieve high fidelity in visualizing the

results so that salient properties can be readily discerned and the overall implications of the simulation can be communicated widely. The goals for such visualization strategies include scalability when preparing high-quality, realistic renderings, and the incorporation of the surrounding environment in which the simulated events take place. The literature on scientific visualization concerns itself, to a large part, with the analysis of acquired images from devices such as MRI or CT scans, and with the extraction of significant features such as stream lines and vortex cores from simulated flow fields.

Our data for this paper is derived from the simulation of the 9/11 events using LS-Dyna. This simulation is situated in the context of aircraft crash experiments that have been carried out and described by Sugano and others [5]. The insights from these earlier experiments include understanding the time-dependent force exerted through the impact on a building or other structure, as well as a risk assessment whether the building or structure is breached. The experiments usually include smaller planes, and the results of those crashes are then transposed to larger plane crashes by some scaling argument. On experimental grounds it has been found that the engine of a jet fighter incurs the most significant damage in a crash. Elsewhere [22] we have argued that in the case of the Pentagon attack it was not the engines that did the major structural damage. In fact one of the engines was lost at the generator outside, before the impact of the plane into the Pentagon. However, we are unaware of significant crash experiments and analyses that investigate consequent damage to structures impacted by a fully fueled commercial jet liner. This specific question must await future experiments to be fully understood experimentally.

An FEA system such as LS-Dyna writes, as part of the simulation, a data base of nodal positions, pressures, stresses, and other engineering quantities at time intervals chosen by the user. There are visualization packages devoted to interpreting and rendering such FEA data visually, notably the VTK tool kit [10]. These visualization packages do not offer tools that incorporate environmental settings or simplify animation scripting of, e.g., the flight path prior to impact. Animation and rendering software such as 3DS Max, on the other hand, offer many sophisticated tools to render the environment, construct pre- and post-event animations, and add special effects.

3DS Max and other such rendering and animation systems do not offer specific tools to import FEA data. In fact, the shape representation of 3DS Max and LS-Dyna are conceptually very different, the former based on visible surfaces, the latter based on representing the interior of objects as well as their surface. Since 3DS Max allows programmatic extensions to enhance and customize system functionality, there is in principle the possibility of writing a plug-in that extracts the needed FEA data from the pre-computed data base, reformats it, and then renders it in the context of the surrounding environment and its other components. Such a plug-in has to be scalable in order to be able to ingest the FEA data in a reasonable amount of time.

Industrial Light and Magic has worked on combining physical simulation with sophisticated visual presentations that include a cinematic environment. This is the closest work relating to our effort . For example, the animation of Star Wars Episode V includes importing a rigid-body dynamics simulation of the motion of the bracelets on the arms of one of the synthetic actors [24]. In this proprietary work, the simulation software has been written from scratch and carefully tuned for the specific problem at hand. It cannot provide a general solution such as the one we present in this paper based on federating general-purpose simulation and animation software.

Simulation and Meshing

For the impact simulation, LS-Dyna [8] was selected as the FEA code since it is widely used in automobile crashworthiness investigations and has the ability to model and analyze large-scale deformations and nonlinear material behavior. The forensic investigation after the attack, conducted by the Association of Civil Engineers (ASCE), provided the raw data for the observed

damage to the reinforced concrete columns of the building [1]. The column damage pattern in the first floor of the building provided therefore a ready basis for confirming the simulation results.

The fidelity of the simulation is influenced by the mesh size. A coarse discretization used for the elements of the Lagrangian mesh reduces the fidelity for contact and yielding behavior. Similarly, a coarse discretization used for the elements of the Eulerian mesh reduces the fidelity for mass transfer computations for the advection process, Euler-Lagrangian coupling, and the accuracy of fluid flow. For the simulation, we wanted complete control over the mesh and therefore created our own set of meshing tools. This allowed us to balance the desire for high resolution meshes, and the accuracy so obtained, with the necessity to accommodate the limitations of the computing platform, an IBM Regatta.

A completely automated meshing program that derives an FEA mesh from a geometric model is a significant undertaking; e.g., [25,26]. However, much can be accomplished when the meshing is semi-automatic. We wanted a predominantly hexahedral mesh with only a few triangular elements. Because we combined separately meshed parts, we chose early-on to generate the mesh in two passes: The first pass generates a mesh description in an intermediate format that is independent of the one LS_Dyna expects. A second pass then translates the intermediate representation into a form that is suitable for input to LS_Dyna. Thus, we can easily change the FEA package, for instance when comparing fidelity and performance of different FEA systems.

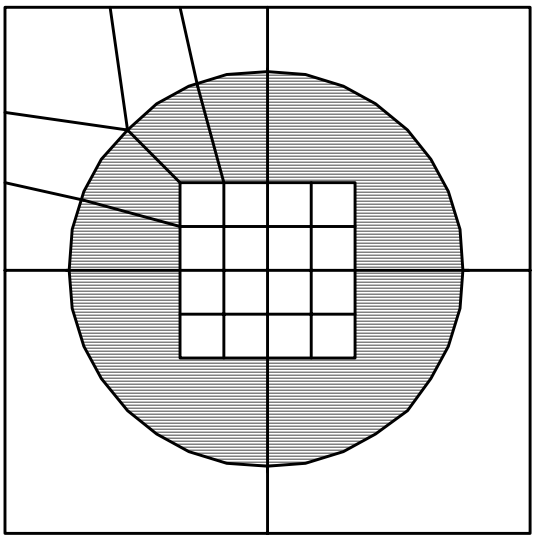


Figure 2 Column meshing for finite element analysis.

The structure of most buildings can be done using meshes of only a few primitive shapes and carrying out various operations combining those primitive meshes. The Pentagon is no exception. However, the columns of the Pentagon have a particular structure that is a significant factor in the building performance and must be modeled. There is a round core of concrete confined by a spiral rebar further secured with longitudinal rebars. This very strong core is faced by an outer concrete shell which is not reinforced and is comparatively weak. Accordingly, we meshed the columns using the topology shown in Figure 2 with rebars arranged around the perimeter of the inner core. Using a different material model for the core and the facing allowed us to model the behavior of the aggregate structure accurately.

The airplane was modeled for the most part using quadrilateral shell elements. This is rather easy for the main cabin and the wings. Beams, stringers and ribs can be added easily by sharing nodes in longitudinal or transversal direction and defining beam elements. The floor of the cabin was also modeled since its rib structure is a significant part of the overall integrity of the fuselage.

The fuel in the tanks of the aircraft was modeled since it provided a large part of the kinetic energy of the impact. We used an Eulerian mesh surrounding the wing tanks and the center tank. The mesh moves and deforms with the moving plane. The action of the liquid is computed by tracking the percentage fluid volume contained in each mesh cell. Initially, the percentage liquid occupancy is the percent volume to which a cell lies in the interior of the wing tanks. Since the

hexahedral elements of the wings are twisted, that is, since they are bounded by faces that are not planar, the volume computation is not entirely routine. We computed the initial percent assignment by generating a very fine subdivision of the wing and center tank cells. In a fine subdivision, the face curvature can be ignored and the volume of each element can be determined rather accurately by an elementary estimate. Each such element was then represented by its centroid, and its volume was added to the fluid occupancy of the Euler cell in which the centroid was situated. This is a simple way to estimate the intersection volume of each Euler cell with the wing and center tanks, but since the summation uses the centroid of the elementary volume, it is possible that some Euler cells are over-full. We accounted for that possibility by smoothing the cell occupancy, distributing excess volume to the neighboring cells that could accommodate it. Only a few passes were needed to resolve over-full elements.

Based on these considerations, a script language was developed to drive a meshing program. Using the script language, meshes in the intermediate representation are created for simple shapes and can be manipulated. Key operations on meshes include identifying nodes, so combining two separate meshes, renumbering nodes and elements, and replicating and repositioning meshes. Meshes can also be deformed. An important aspect of the scripting approach is to allow symbolic expressions that evaluate to coordinates and parameters so that different meshing structures can be parameterized using only a few parameters. The script interpreter therefore is a simple CAD system coupled with a programming language interpreter. The approach is quite efficient, both in user time and in mesh generation computation. For instance, a good mesh for a fighter jet crash can be put together in a few hours from a small set of dimensional measurements.

Conceptually, our mesh generation approach is one in which the geometric models are constructed as part of the mesh generation. In many cases, the geometric model precedes mesh generation and meshing is conceptualized as an automated follow-on that takes a CAD model as input and produces a mesh for it as output. This second approach is more complicated and much work has been done in this area to automate it. A particular difficulty is that the geometric model usually has much more detail than the FEA analysis needs, and it would be useful to first simplify the shape. Unfortunately, shape simplification is a difficult problem that has not yet found a satisfactory solution

Visualization Challenges

Large scale simulations produce massive multidimensional datasets. Although some scalars can be examined and presented using tables and graphs, the 3D visualization of the simulation scene has long been used as a powerful means of conveying the results of such simulations. In the case of FEA simulations that analyze the mechanical interaction of entities under the extreme conditions of a high-kinetic-energy impact, visualization is an indispensable tool. In order to capitalize on high-fidelity finite element modeling, material and contact behavior, the visualization should have high fidelity as well.

The fidelity of the visualization can be measured in two dimensions: rendering quality and visual realism. Rendering quality is determined by factors that include complexity of light, material and shading models, pixel and color resolutions, and level of antialiasing. The rendering quality has to be sufficient to reveal all information contained in the simulation results. Detail is essential for developing, confirming and/or dismissing theories and hypotheses about the simulated events. Moreover, a high-quality rendering of the simulation results is a powerful debugging tool useful for designing new simulation codes.

Visual realism enables the dissemination of the simulation results to non-specialists. Presenting the simulation such that the entities involved are easily recognized and associated with their real scene counterparts improves the communication in interdisciplinary teams, improves decision

making in defense and emergency management, increases the effectiveness of virtual training, and provides powerful yet accurate opinion forming material suitable for use in courts of law and mass media. Visual realism also helps the specialists that designed the simulation compare the results of the simulation to the results observed during real experiments. In the case of complex scenes, not every part of the scene can nor should be included in the simulation. The computational resources should be allotted to accurately following the behavior of the entities most relevant to the event studied. An important aspect of visual realism is the modeling, integration and rendering of the surrounding scene with the simulation scene.

Besides producing high-quality images of the simulation, a successful visualization system has to have important basic capabilities including scene file management, material editors, convenient view selection tools, interactive rendering mode (albeit at draft quality), lights design tools, and network rendering.

Visualizing the enormously complex simulation scene is a challenge even for the latest graphics workstations and software systems. Although the geometric complexity of the scene is manageable, animating every node at every state has proved to be a serious bottleneck. Another important challenge is rendering the animated liquid. Global illumination models are inappropriate since they require laborious reflection and refraction rendering algorithms. Liquid animation is challenging. The liquid behavior is computed by the simulation code for each state. A slow motion visualization of the simulation requires rendering the scene at intermediate time steps. Interpolating the liquid between the simulation states in a way that achieves continuity is a problem.

Modeling the surrounding scene realistically is another challenge. Usually there is no 3D data available for the surrounding scene. The scene has to be constructed from heterogeneous data such as outdated blueprints, photographs, low resolution video, and witness accounts.

Currently there is no visualization software system - commercial or public domain - that overcomes these challenges and produces high-fidelity visualizations of large scale simulations. Various components exist but the differences between the way they conceptualize geometry, material properties and animation are difficult to reconcile. Our work can be thought of as a first cut at building such a system.

Solutions

Commercial animation systems offer a good number of the desirable features, including state-of-the-art high-fidelity rendering, material editing, view selection and camera animation, light modeling, geometry editing, distributed rendering and convenient scene management. In addition those systems have a relatively open architecture. We decided to use a commercial animation system as the basis for our system.

The first problem that we had to be overcome was importing the simulation results data into the animation system. For this we implemented a custom plugin, a custom programmed extension of the animation system, that loads the simulation data directly into the animation system. The scene is first subdivided by materials. The user can choose to load only the objects made of certain materials, and/or a subset of the states. Figure 3 shows the data of one simulation state imported into the commercial animation system. No processing has been done yet, all surfaces including the cells of the ALE mesh containing liquid are shown in wireframe. The wireframe visualization has been useful in illustrating the granularity of the finite elements used to run the simulation.

The plugin removes the internal faces of the objects built of hexahedral elements that are irrelevant during rendering. Elements erode over the course of the simulation and faces that are originally hidden can become visible at the fracture. For this the objects are subdivided according

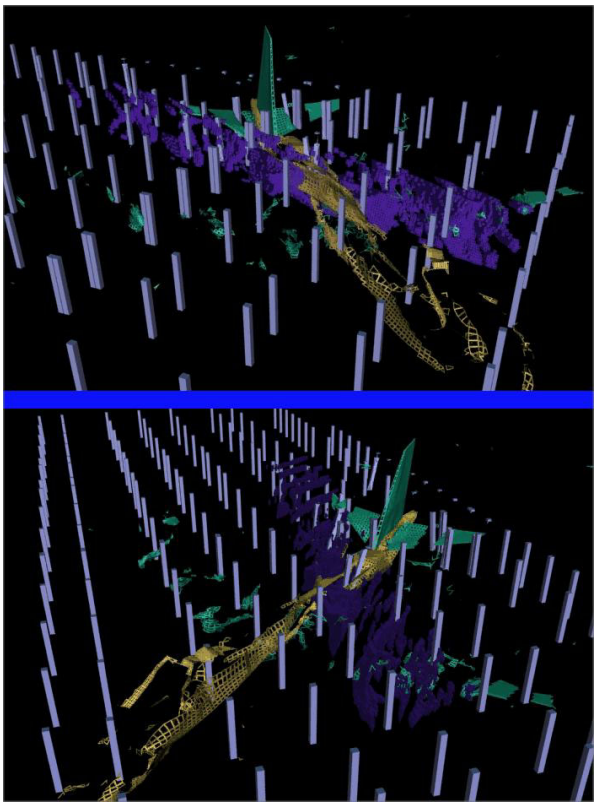


Figure 3 Simulation state imported using the plugin.

to the state when they erode. The internal faces of the sub-objects can be safely eliminated, in linear time using a hash table. Shell (quadrilateral) elements are imported as a mesh of triangles or quads and the beam (segment) elements are imported as renderable splines.

The FE code computes the position of the node at every state. Adding one position controller for every node and for every state takes days and produces unusable scene files that take hours to load and save and require minutes to update each frame. Note that position controllers are needed only when a node moves off the line defined by the previous and next controller. The user can intuitively tune the collinearity threshold which is measured in distance units (millimeters in our case) to control the accuracy of the trajectory approximation. Maximum fidelity is achieved when the threshold chosen is the machine precision, which is still economical in the number of controllers for the stationary part of the scene.

In LS-Dyna, liquid can be modeled using an arbitrary Lagrange-Euler formulation (ALE) or else using a smooth particle hydrodynamics approach (SPH). We chose the former for the 9-11 simulation because of its greater maturity at the time. This is the imported data format

For a given state, the fractional occupancy values are transformed in a mesh (Figure 4) defining the surface of the liquid either by selecting all the cells that have at least a minimum occupancy value and eliminating the internal faces or by building an isosurface. The animation system offers much support for designing the liquid material including custom attenuation-with-distance functions, surface reflectivity, and refraction index. Raytracing the liquid mesh with such a material produces convincing visualizations (Figure 5). This degree of realism is not always needed as it could hide important details or blur the distinction between surfaces. For example, rendering the liquid as a Lambertian opaque object helps visualize the shock waves.

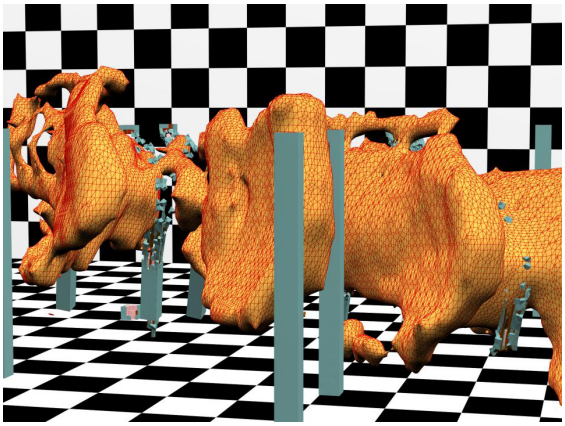


Figure 4 Liquid visualization. Mesh used to render the jet fuel shown in wireframe.

Animating the liquid is more difficult than animating solid objects. Although the Eulerian grid offers a connection between states, the occupancy values vary considerably. This makes the liquid shells very different from one state to the next. A morph that continuously changes the shell at

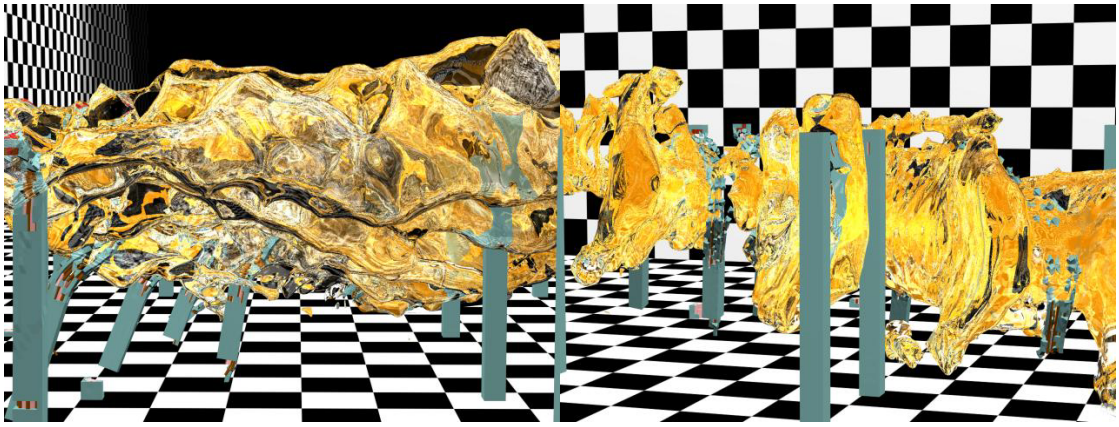


Figure 5 Liquid visualization including reflections and refractions (raytracing).

state k into the shell at state k+1 is hard to develop. We opted for a simpler solution that worked well. Instead of representing the liquid as a shell deforming over the course of the animation, we interpolated the occupancy values and the Eulerian grid node positions for every frame of the animation. Visibility controllers (one per liquid shell) ensure that the appropriate shell is visible for the current frame. This technique does not allow extremely slow animations since that would exceed a practical geometry budget. In our case the technique worked well for up to 5 interpolated states, which totals (5+1)*50 = 300 animation frames, or 10 sec replay time at 30 fps.

The plugin binary is a 200 KB dynamically linked library that the user copies in the plugin folder of his animation system. The plugin will work with future releases of the animation system. Much of the plugin is independent of the input data format. Future plugins that handle the output of other simulation codes can therefore be constructed at a fraction of the effort to implement this first plugin.

Image-based rendering (IBR) is a relatively recent direction of computer graphics research and is well suited for realistically modeling and rendering the surrounding scene. IBR techniques can be subdivided into classes according to the type and amount of geometry they use to model the scene. At one extreme are light field methods [14, 15] that model the scene exclusively with images. Registered photographs are converted in ray databases that queried during rendering for the rays defining the desired view. Unfortunately the technique is less suited for very large scenes. Image-based rendering by warping (IBRW) [12] uses images enhanced with per pixel depth as rendering primitives. Several depth extraction technologies are available. When depth is available, IBRW is a good solution for realistically rendering natural scenes. View morphing and panorama rendering limit the camera motion and are of less interest for our application.

Figure 6 Approach scene. The plane and building are rendered using a CAD model. The ground plane is textured from a satellite photograph.

Hybrid techniques rely on a coarse geometric model of the scene that is colored using photographs [13]. We chose this technique since the required scene description data was available.

Figure 7 Photograph projected on geometric model of Pentagon building.

From blueprints we built a model of the Pentagon building. The terrain surrounding the Pentagon was modeled simply by a large plane.

High-resolution aerial and satellite imagery was used to color the model. For the approach of the plane we mapped a satellite image [18] onto the ground plane (Figure 6). The satellite image covers the surrounding area well, but it contains no information about the façade. We texture mapped the façade conventionally with stock textures available in the animation system. The airplane was rendered using a B757 model used for computer games [17].

The scene after the attack was modeled using a high resolution photograph taken from a helicopter (Figure 7). The CAD model was altered manually to match the destruction visible in the photograph. Then the photograph was sprayed onto the ground plane and modified CAD model. The process consists of two steps, camera matching and projective texture mapping. Camera matching finds the position and view direction for each of the photographs used to texture the model. We selected manually correspondences between the photograph and the geometric model. The search converged quickly to solutions with only 2-3 pixel errors for 3000 x 2000 pixel images taken with an unknown camera. Projective texture mapping effectively sprays the photograph pixels onto the polygons of the model and automatically builds individual texture maps. The resulting sprayed model produces very realistic animations which instantly place the simulation in context (Figure 8).

The scene file can be exported to a format suitable for hardware accelerated rendering such as VRML, which allows for the interactive exploration of the visualization results. In that case, the liquid has to be rendered in a simpler but much more efficient way such as alpha blending.

A final, important step in conveying information through visualization is to present the user with the views he wants. Often, especially in the case of complex 3D scenes, the user does not know exactly which view reveals best the important properties of the phenomena under scrutiny. Typically the user finds it necessary to freely navigate through the 3D scene, in search of an ideal view. The standard approach is for the user to specify a new view by incrementally changing the current view, using keyboard, mouse or joystick. This approach is artificial and non-intuitive, thus it is an ineffective way of navigating the 3D scene.

Figure 8 Novel view rendered from the projectively texture-mapped model.

Virtual Reality technology provides limited support for an immersive exploration of the 3D scene. Using polarizing glasses or head-mounted displays, the user is presented with a stereo pair of views that she/he fuses into a spatial image. The six degrees of freedom of the head are tracked and the rendering system quickly updates the images accordingly. Unfortunately virtual reality technology is still relatively crude. Few or no researchers work for extended periods of time in the virtual environments. Latency, low resolution, inconvenient eyewear, and isolation from real world (collaborators, computer, notepad) are factors that have

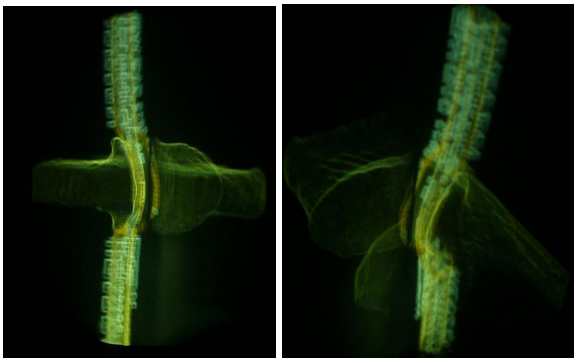


Figure 9 Photographs of volumetric display showing 3D images of liquid-column impact data.

been found to rapidly fatigue the user.

A promising alternative are 3D displays that show a 3D scene so that it can be viewed by one or several users simultaneously, without glasses. We have experimented with good results with an autostereoscopic display [23]. A rapidly revolving screen is located inside a glass dome and displays in rapid succession slices of the 3D image to be displayed. Because of the latency of the eye, the 3D image is perceived as a spatial sculpture of light. The image can be viewed from any angle simultaneously by as many viewers as can be placed around the 20 inch glass ball. The color depth of our device is limited to 3 pixels (8 colors) but the effect is visually stunning. The latency-free motion parallax obtained by freely translating the head and walking around the display are unparalleled by any classic 2D or VR display technology. The data imported into the animation system using our plugin can be re-exported in a format suitable for the 3D display, and this high-lights another important use of the plugin as a data conversion tool. Figure 9 shows two photographs from different angles of the 3D display displaying the simulation data of a liquid-column impact simulation.

Summary and Future Work

We have demonstrated high-fidelity visualization of a large scale simulation using a commercial animation system enhanced with a custom plugin. The plugin has been distributed to civil engineering researchers on campus and they report it to be a useful tool. At first they were using it just to illustrate their work, but now they are using it more and more to inspect the results of their simulations. One of the hurdles was learning the key features of the animation system, a more complex system than the post-processor used before. Tutorials specifically tailored to their needs should speed up the learning process.

The visualization system can be extended and improved in many ways. Various visualization techniques such as boundary and silhouette enhancement, tone shading, and feature halos can be added as rendering plugins. We will investigate importing and rendering particle-modeled fluids. Dust, smoke and fire are important in many simulations, and we plan to address them in the future.

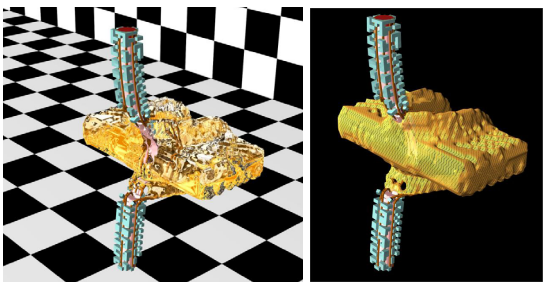


Figure 10 Images rendered by our civil engineering colleagues for illustration (left) and material model validation (right).

An important topic is closing the loop; that is, beginning with a model suitable for rendering, extract from it the FEA model, simulate the model, and then re-import the results into the animation system. A closed loop minimizes the amount of time users need to study a problem. Several approaches come to mind.

A popular approach is to generate meshes automatically from geometric models and use them for the FEA. This approach is particularly attractive in manufacturing where artifacts are created in CAD systems whose performance must be understood using FEA.

Here, difficulties arise because of the possible uncertainties in the CAD model that arise from the internal geometric computations and the fine detail of CAD models that may be unnecessary for the FEA. Full-service CAD systems offer the ability to render mechanical artifacts with high fidelity. However, in forensic situations we do not begin with CAD models of the scene.

An approach better suited to our simulation study would be to build the meshes and the geometric objects simultaneously. For this purpose we have developed the script language in which the models and meshes can be built. This is not ideal because the textual scripts have not been generated using a graphical interface. It should be possible to generate them in the animation system using minimal user interaction. This would require a plugin that allows designating certain parts of models, eliminating some of the features to simplify shape, and annotating elements of the scene.

Acknowledgements

Our many discussions with Sami Kilic from Purdue's Computing Research Institute and Mete Sozen from the Department of Civil Engineering were very helpful and are gratefully acknowledged. The material models of the Pentagon attack simulation were devised by them, and Sami Kilic did the FEA runs and the traditional animations. Scott Meador and Jason Doty of ITaP did the illustrative animations that script the plane approach to the Pentagon and illustrate the context of the simulation. Hendry Lim and Mihai Mudure helped with implementing the projective texture mapping module. Paul Rosen helped display the data on our 3D display. Their help is gratefully acknowledged.

References

1. SEI, *The Pentagon Building Performance Report*, Struct. Engr. Inst. of the Am. Assoc. of Civil Engr., ASCE 2003, 78 pages.
2. V. Popescu, C. Hoffmann, S. Kilic, M. Sozen, S. Meador. "Producing High-Quality Visualizations of Large-Scale Simulations," Proc. Visualization 2003, Seattle, WA, p.
3. URL http://www.cs.purdue.edu/homes/cmh/simulation/
4. 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
5. T Sugano et al. Full-scale aircraft impact test for evaluation of impact force, Nuclear Engineering and Design, Vol. 140, 373-385, 1993.
6. 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.
7. 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.
8. LS-DYNA, URL: http://www.ls-dyna.com/
9. http://www.hpcmo.hpc.mil/Htdocs/UGC/UGC98/papers/3b_chal/
10. http://public.kitware.com/VTK/
11. S. Chen. QuicktimeVR- an image-base approach to virtual environment navigation. In Proc. SIGG. '95, pages 29-38.
12. L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In Proc. SIGGRAPH '95, pages 39-46, 1995.
13. Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and Rendering Architecture from Photographs. In Proc. of SIGGRAPH '96.

14. M. Levoy and P. Hanrahan. Light field rendering. In Proc. of SIGGRAPH '96, pages 31-42, 1996.
15. S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In Proc. of SIGGRAPH '96, pages 43-54, 1996.
16. G. Bell, The future of high-performance computers in science and engineering, CACM 32, 1091-1101, 1989.
17. Casper, Terry, Amazing 3D Graphics, Inc., P.O Box 1821, Payson, Arizona 85547, URL: www.amazing3d.com, 2002.
18. SpaceImaging, URL: http://www.spaceimaging.com/gallery/9-11/default.htm
19. 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).
20. Alias | WaveFront, URL: http://www.aliaswavefront.com
21. Discreet, URL: http://www.discreet.com/products/3dsmax/
22. Christoph Hoffmann, Voicu Popescu, Sami Kilic and Mete Sozen *The Pentagon on September 11th*. IEEE Computing in Science and Engineering, January/February 2004.
23. Actuality Systems. http://www.actuality-systems.com/volumetric3d.php3
24. Siggraph 2002, Yoda and beyond: creating the digital cast of Star Wars Episode II, special presentation 21 July 2002, San Antonio; www.siggraph.org/2002/conference/special/index.html.
25. J. Thompson, Z. Warsi, C. Mastin; Numerical Grid Generation. North Holland, 1985.
26. P. Ang, C. Armstrong; Adaptive shape-sensitive meshing of the medial axis, Engr. With Comp. 18, 253-264, 2002.\