Version 2; Sunday, September 14, 2003

Integrating Modeling, Simulation, and Visualization

Chris Hoffmann, Voicu Popescu, *CS & CRI*
Sami Kilic, Mete Sozen, *CE & CRI*
Purdue University

Abstract

We describe the work of an interdisciplinary team of researchers in geometric computing, computer graphics, and civil engineering to produce a visualization of the September 2001 attack on the Pentagon from physical simulation. The immediate motivation for the project was to understand the behavior of the building under the impact. The longer term motivation is to establish a path for producing high-quality visualizations of large scale simulations that combine state-of-the-art graphics with state-of-the-art engineering simulation.

An immediate challenge was to manage the complexity of the scene to fit within the limits of commercial simulation software systems and available supercomputing resources by balancing model complexity and the significance of detail to the overall event. The second challenge was to integrate the results of the simulation into a high-quality visualization. Here, we implemented a custom importer that simplifies and loads the massive simulation data into a commercial animation system. The surrounding scene is modeled using image-based rendering techniques and is also imported in the animation system where the visualization is produced.

Since we used commercial packages whose detailed workings we cannot modify, a key issue was to federate the simulation and the animation systems. The difficulty here is to account for the fact that the two systems use different conceptualizations of geometry and animation. Reconciling the different views should be done in ways that achieve scalability. The reusable link we created between the two systems allows communicating the results to non-specialists and to the public at large, as well as facilitating communication in teams with members having diverse technical backgrounds.

1. INTRODUCTION

1.1 Problem Description

Simulations have become essential tools in many fields of science and engineering. Scientific simulations are used to crash-test an automobile before it is built, to study the interaction between a hip implant and the femur, to evaluate and renovate medieval bridges, to assess the effectiveness of electronic circuit packaging by running circuit-board drop tests, or to build virtual wind tunnels. In particular, finite-element analysis (FEA) plays a prominent role in engineering. FEA systems compute a variety of physical parameters over the time span of the simulation, such as position, velocity, acceleration, stress, and pressure. The visual presentation of the results is either handed off to generic post-processors or else is studied in specific contexts in the field of scientific visualization.

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

Our team had the goal of producing a visualization of the September 2001 attack on the Pentagon that combines commercial codes for FEA and for animation. The obvious solution is to take advantage of the strengths of both simulation *and* animation systems. The project had two distinct parts. During the first part we designed, tested and then ran at full scale the FEA simulation of the aircraft impacting the building structure. For this part we used LS-DYNA [5], a commercial FEA system often used in crashworthiness assessment simulations. This choice is appropriate because LS-DYNA can handle geometric and material nonlinearities as well as fluid–structure interaction [8]. Another choice could have been MSC-DYTRAN [6]. Both codes implement explicit time marching schemes. Implicit time integration schemes would be inappropriate for impact problems that involve high-frequency response [7]. In the second phase the efforts were focused on producing a high-quality visualization of the massive data resulting from the simulation. In order to do so we created a scalable link between the FEA system and a commercial animation system (3ds max [18]). The link can be directly reused to create animations with physical fidelity regardless of the scientific or engineering domain.

Ironically, the data for the aircraft model came from a 3ds max model that was hand-imported into the FEA analysis. Based on our experience, we propose several ideas how to make this link at least semi-automatic, and what tools can be used to simplify this process, the most time-consuming part of the simulation and analysis step. This would go a long way towards closing the loop and accelerating the overall cycle of modeling → simulation → animation → model refinement.

1.2 Purpose

We posit that high-quality visualization of scientifically accurate simulation data has the following important advantages:

1. Effective communication of the results and insights to the public at large,
2. Ability to build a case for the merits of a project, especially for large-scale projects that require grass-roots support,
3. Better collaboration between members of interdisciplinary teams.

The images gathered by the Hubble Space Telescope are an example of the value of high-quality images vis-à-vis points 1 and 2.

A high-quality visualization of the results of a simulation requires first that the objects whose interaction is simulated be rendered using state-of-the-art rendering techniques. A second requirement is that the simulation be placed in the context of the immediate surrounding scene. For this the scene has to be modeled and rendered along with the simulation results. Such visualization makes the results and conclusions of the simulation directly accessible to others than the specialists that designed the simulation, without sacrificing scientific accuracy. This will make scientific simulations powerful tools that will routinely be used in a variety of fields including national security, emergency management, forensic science, and media.

A good visualization ultimately leads also to improvements of the simulation itself. High-quality images from a simulation quickly reveal any discrepancies with observed experimental data. We next give an overview of the process that converted the heterogeneous data documenting the event into the desired visualization.

1.3 Overview of the Work

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

The visualization part of the project began with modeling the Pentagon building from architectural blueprints using a CAD tool. The geometric model of the building and the surroundings were enhanced with textures projected from high-resolution satellite and aerial imagery using a custom tool. The 3ds max aircraft model used for visualizing the approach was readily available. The 3.5 GB of state data describing the mesh deformations was simplified, converted and imported into the animation system through a custom plugin. The imported meshes were aligned with the surrounding scene and enhanced with rendering material properties. Finally the integrated scene was rendered from chosen camera paths. Figure 1 and Figure 2 show the simulation results at the same time step from near identical views, once using our system and once the post processor, respectively.

Prior work is discussed next. The remainder of the paper is organized as follows. Section 3 describes the simulation; section 4 describes modeling the part of the scene not involved in the simulation; section 6 covers importing the simulation data into the animation system. Results are presented for each section separately. Discussion and directions for future work conclude the paper.

2. PRIOR WORK

Baker et al. [1] describe the simulation of a bomb blast and its impact on a neighboring building. The scenario investigated matches the 1996 attack on the Khobar towers. Two computational codes were used. The blast propagation was computed using CTH [3] at the Army's research lab in Vicksburg [10]. Results of the CTH calculation are used as initial pressure loadings on the buildings and Dyna3D [4] is then used to model the structural response of the building to the blast. The results were visualized in the Dyna3D postprocessor and VTK (visualization toolkit [11]) using standard visualization techniques such as slicing and isosurfacing. The researchers report the difficulty of visualizing the large data sets; the solutions employed are reducing resolution, decimation and extraction of regions of interest. Enhancing the quality of the visualization using photographs is mentioned as future work.

A considerable body of literature in nuclear engineering is dedicated to simulating the crash of an aircraft into a concrete structure. Provisions for aircraft impact on reinforced concrete structures are incorporated into the Civil Engineering codes used for the design of nuclear containment structures. A full-scale test was conducted by Sugano et al. [2] to measure the impact force exerted by fighter aircraft (F-4D Phantom) on a reinforced concrete target slab. The study provided important information on the deformation and disintegration of the aircraft.

However, this study crashed an aircraft into a reinforced concrete slab. This study gives experimental evidence that the airframe and the skin of the aircraft alone are not likely to cause the major damage on reinforced concrete targets.

To contextualize the simulation, we had to model and render the surroundings of the Pentagon. Research in image-based rendering (IBR) has produced several successful approaches for rendering complex large-scale natural scenes. The QuicktimeVR [13] system models the scene by acquiring a set of overlapping same-center-of-projection photographs that are stitched together to form panoramas. During rendering the desired view is confined to the centers of the panoramas. In our case it was important to allow for unrestrained camera motion so we dismissed the approach.

Image-based rendering by warping (IBRW) [14] relies on images enhanced with per-pixel depth. The depth and color samples are 3D warped (reprojected) to create novel views. Airborne LIDAR sensors can provide the depth data at appropriate resolution and precision. In the case of our project no depth maps of the Pentagon scene were available and we could not use IBRW. In light field rendering the scene is modeled with a database containing all rays potentially needed during rendering. The method does not scale well: the number of images that need to be acquired and the ray database grow to impractical sizes for large-scale scenes.

An approach frequently used for modeling large urban scenes combines images with coarse geometry into a hybrid representation. A representative example is the Façade system [15] which maps photographs onto buildings modeled with simple primitive shapes. The system was used to model and realistically render a university campus environment. The relatively simple geometry of the Pentagon building and the availability of photographs of the area motivated us to choose a hybrid geometry / images approach as described in section 4.

3. LARGE SCALE SIMULATION

The impact of the aircraft on the Pentagon building was simulated using LS-Dyna, which is a non-linear finite element code. LS-Dyna is capable of modeling fluid-structure interaction by employing a Lagrangian mesh for solid elements coupled with an Eulerian mesh for fluids.

The motion of the fluid is based on the Navier Stokes equations used in computational fluid dynamics. The fluid mass is transferred among Eulerian cells through advection. The fractional occupancy of each cell is reported by LS-Dyna as the solution progresses from one time step to the next. Figure 3 shows the part of the Eulerian mesh consisting of nonempty cells at the initial state.

When different parts of the model that approach each other come into contact, the time step size is scaled down further in order to capture the non-linear behavior of the large deformations and material failure more accurately. The running time of the computation is significantly influenced by time step reduction upon contact.

In the overall analysis of the problem, the largest amount of time was spent on mesh generation and model assembly. For the largest mesh size the studied simulation took 4 days of running time. The long running time is explained by having to use a very short time step in order to correctly capture the large deformations, and to allow coupling the Eulerian and Lagrangian meshes.

The results of the simulation are validated on two basic notions. The first one is qualitative and is based on a visual inspection of the behavior of individual components of the model. The tail shell buckling of the aircraft fuselage in the large simulation of the Pentagon building study shown in Figure 1 is a typical example of visual inspection. Similarly, the fluid dispersion that occurs after the wings of the aircraft hit the columns of the building provides an initial assessment of the admissibility of the results. The fluid properties used in the simulation represent typical kerosene jet-fuel contained in the wing tanks of the aircraft.

The second notion is based on quantitative evaluation of the response of the components of the model against benchmark case studies. For the larger simulation of the Pentagon building with spirally reinforced concrete columns, a case study was created for investigating the behavior of a single column impacted by a block of fluid. The constitutive concrete material model was calibrated such that the response of the column agreed with the results available from experimental studies and from hand-calculation models that are well established.

The standard models of concrete available with LS-DYNA were not sufficiently accurate for our purpose. So, we adjusted the non-linear material model for concrete and selected a suitable erosion criterion. Without going into detail in this article, we used test cases of impacting a single concrete pillar with a block of fluid and we adjusted the model parameters accordingly, drawing on our experience with concrete behavior [9].

The fluid properties are adjusted to represent water. The main physical properties of the fluid utilized in the Eulerian approach are density and viscosity. An equation of state defines the pressure-volume relationship for the compressible fluid and its initial thermodynamic conditions.

The full simulation consisted of approximately 1M nodes and took 68 hours of run-time on an IBM Regatta system for a simulation time of 0.25 second. The reinforced concrete columns in the vicinity of the impact area were modeled with higher fidelity than the columns far away, and the Eulerian mesh for the liquid was finer than in earlier simulation runs we made.

4. MESH GENERATION

For the simulation we generated a mesh by a set of custom programs. This choice reflected the difficulty obtaining meshing tools that can generate hexahedral meshes for complex geometries. The program tools were specific to the type of object generated, that is, separate tools generated individual column meshes, meshes for the wings, meshes for the aircraft body, and so on. This choice made meshing time-consuming. However, it was clear to us from the outset that the mesh densities should be parametrically adjustable, and we wanted full control over how to do that.

With full control over the mesh, we were able to experiment with different types of meshes and different densities of the mesh. For instance, instead of eroding shell elements with a maximum strain imposed, one can replicate the nodes and assign a lower strain failure limit to coincident nodes. With this alternative, individual shell elements would simply separate rather than entire shell elements eroding, so that the mass of the system remains constant.

The mesh of the aircraft was obtained from a 3ds max model (Figure 5). The body was simplified to an ovoid cylinder fitted to two swept cones. The wings were composed from four hexahedral sections. The floor of the main cabin was added in as were stringers and ribs. The geometry was derived from a small set of hard points whose coordinates were read from the 3ds max model.

Typically, finite element meshes simplify the geometric model by eliminating small details. This can be justified by the insignificant contribution those details make to the overall structural behavior and integrity. In this spirit, the round of the leading wing edges was eliminated.

5. SURROUNDING SCENE

Modeling the surrounding scene has forensic relevance since it enables a virtual reenactment of the events. The reenactment is important for corroborating and validating various eye-witness accounts, and for interpreting the low-resolution, slow-shutter video footage of the events recorded by the surveillance cameras. Modeling the surrounding scene also places the simulation results in context to make it easily understood by someone who was not closely involved with the investigation. The physically accurate, visually realistic animations we so produced document the tragic events.

As described earlier, we modeled the surrounding scene using a hybrid geometry / image-based approach. From the architectural blueprints we produced a CAD model of the building. The damage in the collapsed area was modeled by hand to match available photographs. The region surrounding the Pentagon was modeled as a large plane. The geometric models were enhanced with color using high-resolution satellite [16] and aerial imagery [12].

The coloring of the geometric primitives (triangles) using the photographs is done by projective texture mapping [17], which is equivalent conceptually to transforming the camera into a projector. The rays emanating from the camera deposit the pixel colors on the surface of the model to automatically create individual texture maps which are then used during rendering. First one has to establish the position and orientation of the camera with which each of the reference photographs was acquired. Camera matching is illustrated in Figure 6. In a second step texture maps that uniformly sample each triangle are created from the reference photograph. Note that the reference photograph cannot be used directly as a texture by projecting the vertices back in the camera view. The perspective distortion of the reference photograph has to be eliminated first.

In our case camera matching was complicated by not having the camera at hand for intrinsic parameter calibration. In addition to the six extrinsic parameters of the camera pose we also calibrated for the camera's focal length. We searched for the seven parameters using the downhill simplex method and a manually established initial guess. On a 3000 x 2000 pixel image, with 10 correspondences, the matching error was on average 3.5 pixels.

Once the view is known, building the individual texture maps is done according to the following main steps.

- Find triangles visible in the photograph
- For each visible triangle
  - allocate texture
  - set each visible texel by projecting in reference photograph

The visible triangles are collected by rendering in an item buffer that stores ids and depth. The texture resolution is determined using the photograph area of the particular triangle. The texture is defined in model space, so the texels uniformly sample the triangle which removes the perspective projection of the reference photograph. The visible texels are determined using the item buffer. Partially visible triangles and invisible triangles are textured from other photographs.

The building and ground plane model consisting of 25 K triangles was sprayed with a 3000 x 2000 pixels photograph. The resulting texture mapped model produced realistic visualizations of the Pentagon scene. Figure 7 shows an image rendered from a considerably different view than the view of the reference photograph, which is shown in Figure 6. The total disk size of the texture files is 160 MB. The difference when comparing to the 24 MB of the reference photograph is due to the texels outside of the triangle, to the texels corresponding to the hidden part of the triangle, to the thin triangles that have a texture larger than their area and to our simple merging of individual texture images that vertically collates 10 images to reduce the number of files. For now we rendered the scene offline so the large total texture size was not a concern. For real time rendering, the texture size has to be reduced. A simple greedy algorithm for packing the textures involving shifts and rotations is likely to yield good results. The rotation can be propagated upstream to the spraying to avoid the additional resampling.

6. INTEGRATION

The simulation results files are directly imported in 3ds max via a custom plugin Figure 8. The 954 K nodes of the FEM define 355 K hexahedral (*solid*) elements used to model the column core and the fluff, 438 K hexahedral elements for the liquid elements, 15 K quadrilateral (*shell*) elements used to define the fuselage and floor of the aircraft, and 61 K segment (*beam*) elements used to define the ribs and stringers of the aircraft and the rebars of the columns. The importer subdivides the simulation scene into objects according to materials to facilitate assigning rendering materials.

6.1 Solid objects

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

However, importing the mesh deformation into the animation system proved to be a serious bottleneck. The mesh deformations are saved by the FEA code as node positions at every state. The animation system supports *per vertex* animation by controllers that move a vertex on a linear trajectory. Since the node movement is not linear in general, one could create a controller for each of the 50 positions of each of the remaining 700 K nodes to interpolate linearly between consecutive node positions. But doing so takes days of computing time and the resulting scene file is unusable. The practical limit on the number of controllers is about 1M. We reduce the number of animation controllers in two ways. First, the importer does not animate nodes with a total movement (sum of state to state movement) below a user-chosen threshold (typical value 1 cm). Second, the trajectories of each node are simplified independently by eliminating (i.e. not creating) controllers for the nearly linear parts. After simplification, 1.8 M controllers remained. So, we distributed the simulation scene over three files, each covering one third of the simulation. Materials and cameras can of course easily be shared among several files. Importing the solid objects takes 2 hours total, out of which 1 hour is needed for the third part of the simulation. Once the solid objects are loaded, the animator assigns them standard 3ds max materials.

6.2 Liquid objects

The liquid data saved at every state contains the position of the nodes of the Eulerian mesh and the fractional occupancy values at that state. The liquid could be directly rendered from the occupancy data using volume rendering techniques. We chose to build a surface boundary representation first in order to take advantage of the rendering capabilities of the animation system. For every state the importer selects the Eulerian mesh elements that have a liquid occupancy above a certain threshold (typical value 25%). The internal faces are eliminated similarly to the solid object case. Once the liquid is imported, the animator uses 3ds max tools including mesh modifiers and complex ray-traceable materials to produce compelling visualizations of the liquid. In Figure 9 refraction, surface reflections, attenuation and variable opacity provide realism. Rendering at VGA resolution takes approximately 5 minutes.

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

The first approach is in the spirit of animation systems where the same geometric entity suffers a series of transformations over the animation time span. The state of the geometric entity is known at the simulation states; it can be computed by thresholding or isosurfacing the occupancy data. In order to define a morph that produces the animation frames in between the states, correspondences need to be established. This is challenging since the liquid can change considerably from one state to another; it implies that there are different numbers of vertices, different local topologies (drops, liquid chunks separating and reuniting). We have attempted to implement this approach using the Eulerian mesh as a link between states. Because the occupancy values vary considerably from one frame to another, many small liquid objects are generated. This leads to a large number of position controllers.

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

7. DISCUSSION AND FUTURE WORK

The most massive impacting element was the fuel. The fuselage of the aircraft has little strength under axial impact, as confirmed by the simulation and validated by actual experiments [2]. The simulation clearly shows that the structural damage occurs only when the fuel mass hits. The simulation can be extended to cover a longer period of time, with denser states, involving higher resolution meshes; other possible extensions are modeling the building and aircraft in more detail and including the effects of the explosion, of the high temperatures and of the combustion.

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

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

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

Based on our experience with mesh generation we have begun to devise an interactive script language by which to specify hexahedral meshes. The language can partially automate the meshing of complex geometries and supports this task with operations on meshes. Scalability is achieved by using a file operations when necessary. Conceptually, therefore, the FEA model is created directly from the meshing operations. We believe that this concept can be extended to a more automated mesh approach that, at the same time, closes the loop between meshing and model acquisition/inspection in 3ds max. Namely, we envision a set of plug-ins that allows a user to select a component geometric structure in 3ds max, or designate a part of a model by drawing on it, and then generates the corresponding script for this structure. The resulting mesh can then be re-imported into 3ds max.

8. ACKNOWLEDGEMENTS

We would like to thank Scott Meador for his help with 3ds max and Jim Bottum and Gary Bertoline for providing access to supercomputing facilities here and elsewhere in the US. William Whitson helped with the supercomputer runs; Hendry Lim and Mihai Mudure implemented texture spraying; Mary Moyars-Johnson and Emil Venere publicized this work; Amit Chourasia modeled the Pentagon building; Jason Doty produced the first video illustration of our project and Raj Arangarasan helped with an earlier implementation. This work was supported by ITaP, Computer Science Purdue, NSF, ARO and DARPA.

References

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

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

[3] J. M. McGlaun, S. L. Thompson and M. G. Elrick 1990. "CTH: A three dimensional shock wave

physics code", Int. J. Impact Engng., Vol. 10, 351 – 360.

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

[5] LS-DYNA, Livermore Software Technology Corporation, Livermore, California, URL: http://www.lstc.com/

[6] MSC-DYTRAN, McNeal-Schwendler Corp., URL: http://www.mscsoftware.com/

[7] K. J. Bathe, Finite element procedures, Prentice Hall, pp 1037, second edition, 1995.

[8] M. Souli, ALE and fluid-structure interaction capabilities in LS-DYNA, 6th International LS-DYNA users conference: Simulation 2000, Michigan, pp 15-37, 2000.

[9] S. A. Kilic and M. A. Sozen, Evaluation of effect of August 17, 1999, Marmara earthquake on two tall reinforced concrete chimneys. American Concrete Institute, Structural Journal, Volume 100, No. 3, pp 357-364, May-June 2003.

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

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

[12] "Pentagon Building Performance Report", American Society of Civil Engineers (ASCE), 2003, 88 pages.

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

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

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

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

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

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

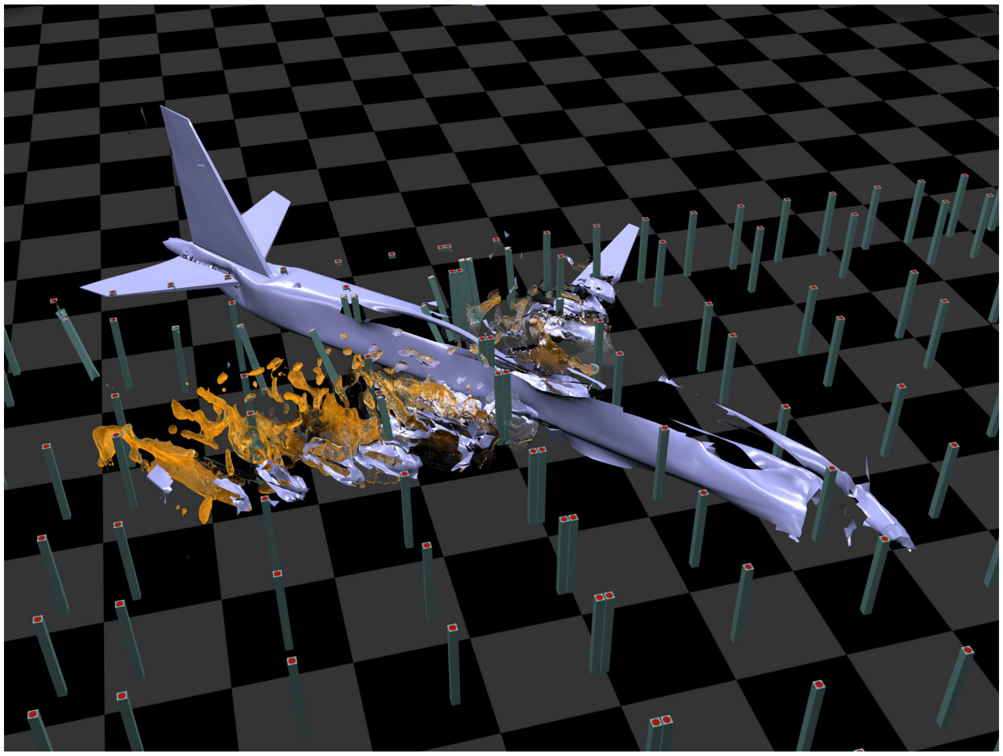


Figure 1 Visualization of the simulation produced with our system

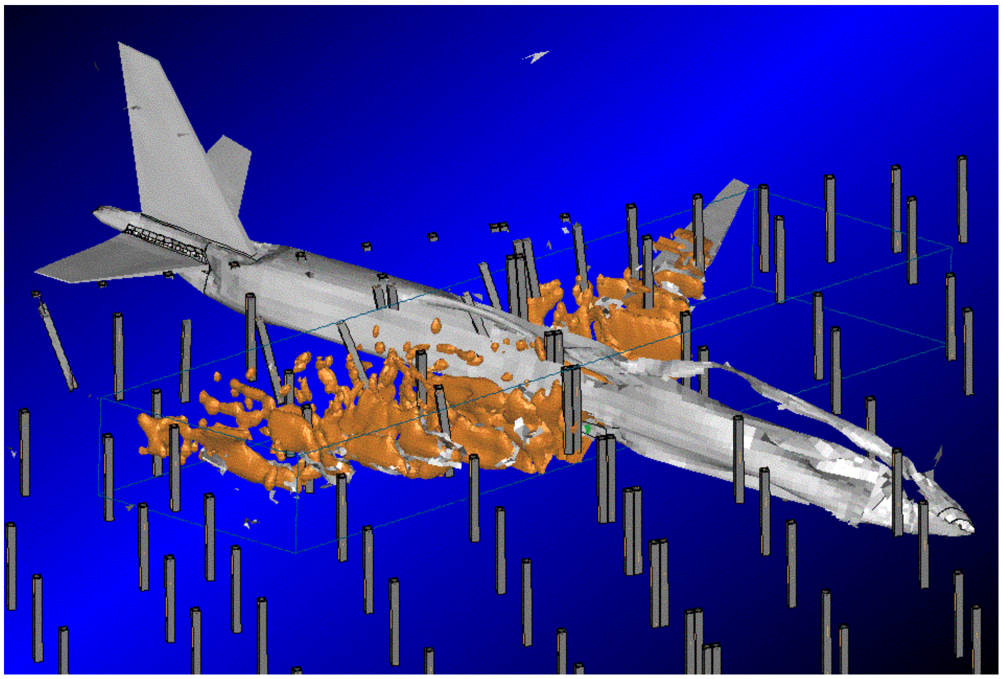
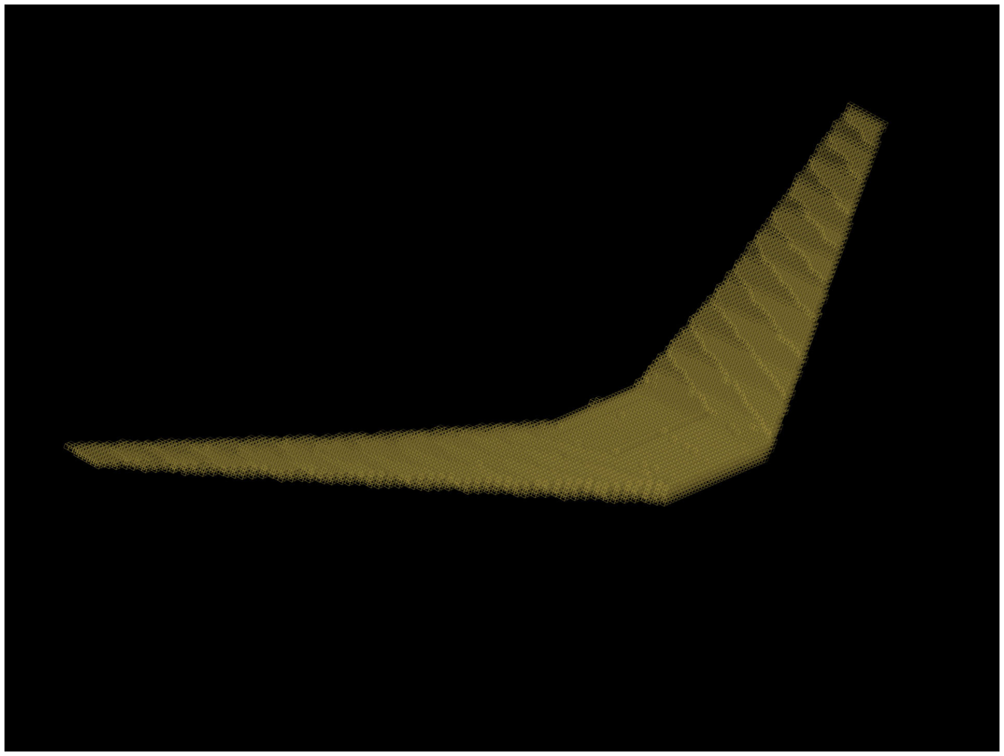


Figure 2 Similar view produced with the post processor



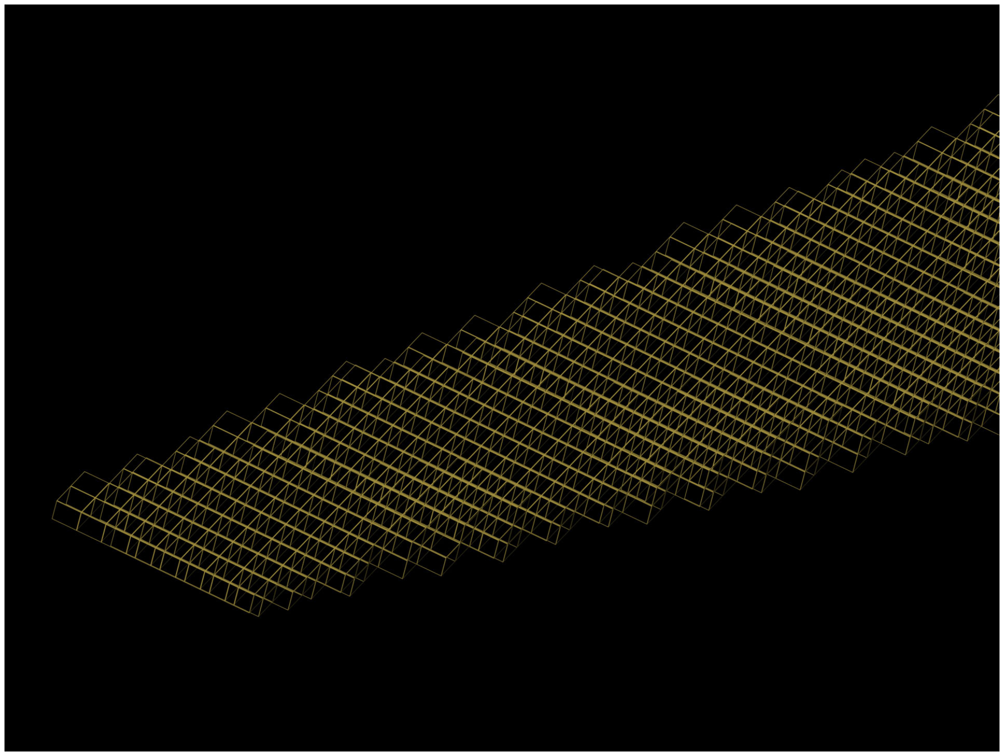


Figure 3 Eulerian mesh that defines the liquid fuel in the wing and central tanks: overall view (top) and detail (down)

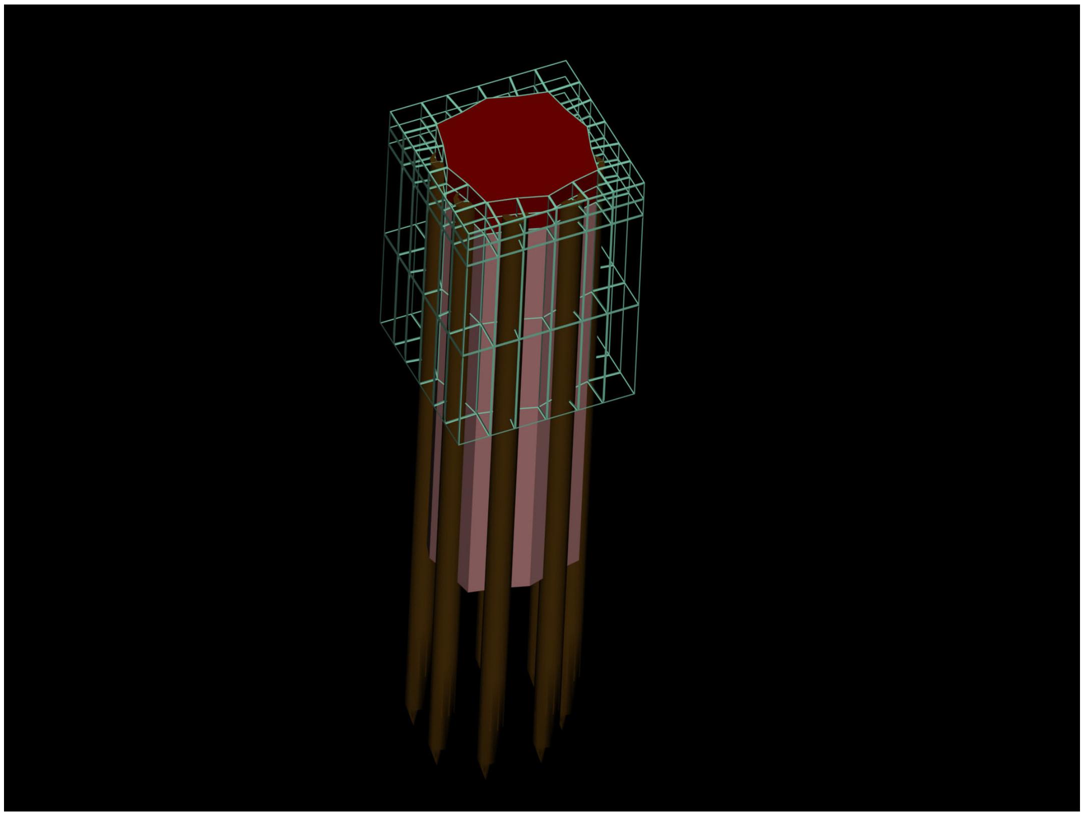
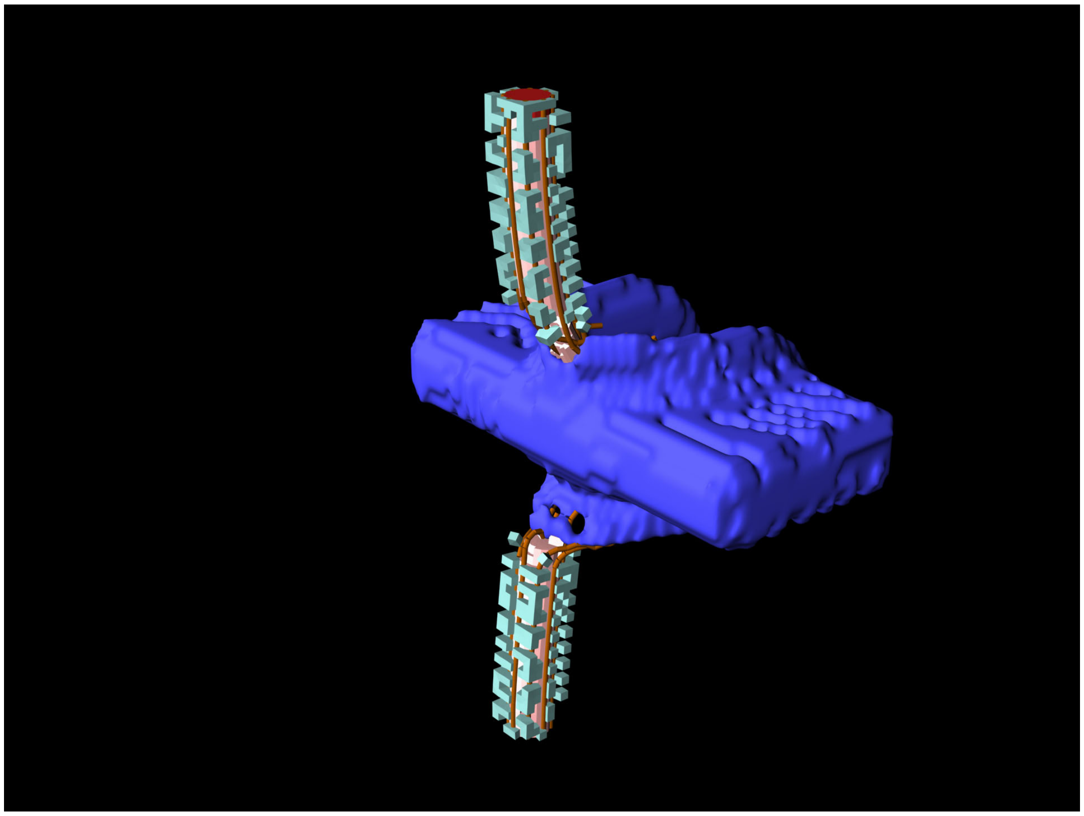



Figure 4 Finite element mesh of concrete column consisting of confined concrete core (pink), rebars (brown), outer concrete fluff (light blue) and anchor (red). Erosion of elements and column destruction caused by impacting block of water.

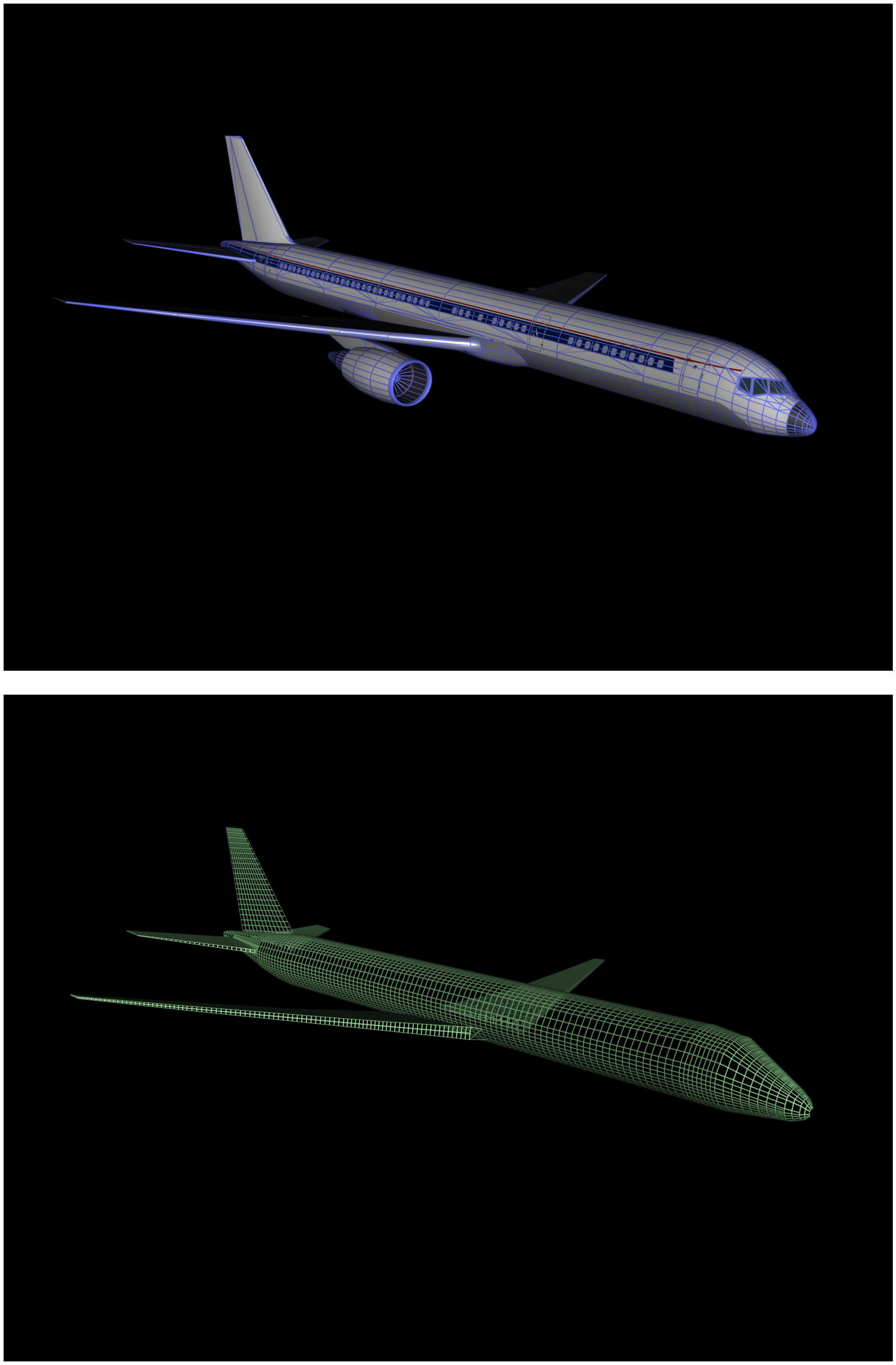


Figure 5 Airplane model (top) and constructed finite element mesh (bottom).

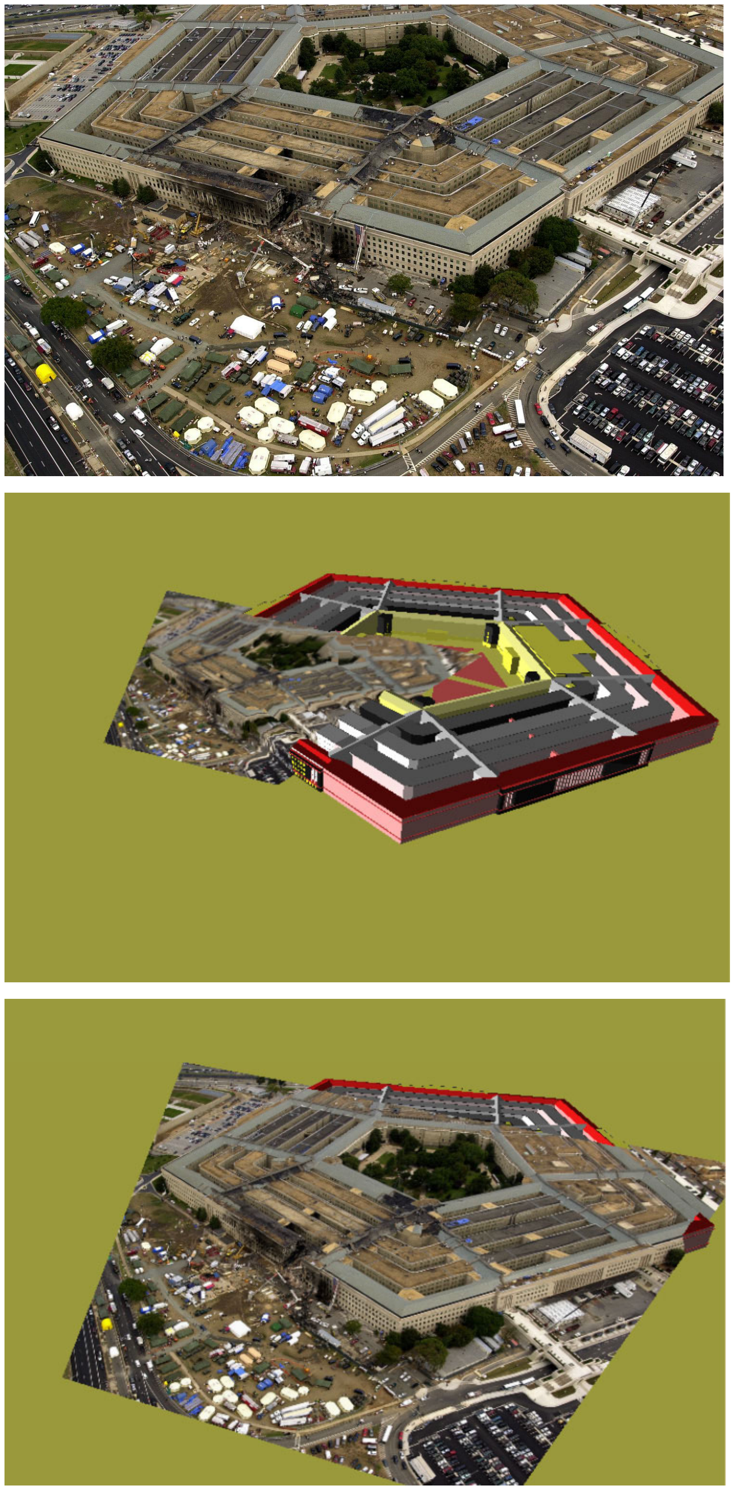


Figure 6 Reference photograph (top), pose used to start the camera matching search (middle) and solution found (bottom).

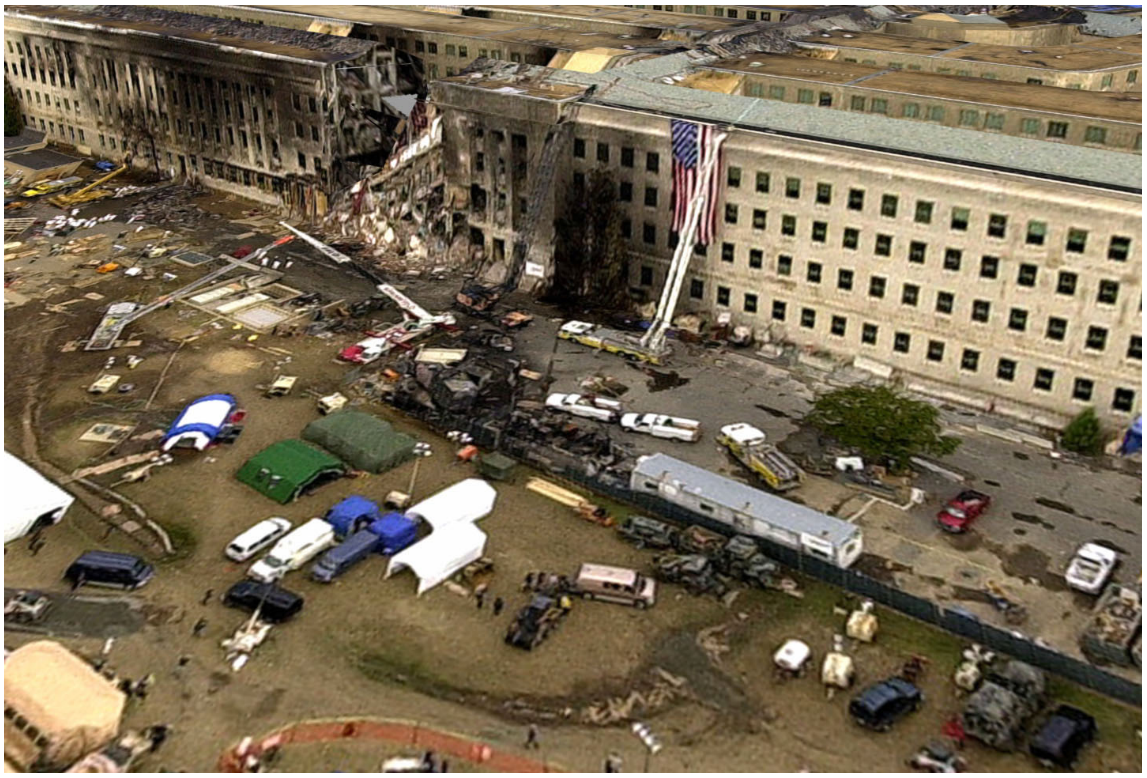


Figure 7 Novel view rendered from the texture-sprayed model.

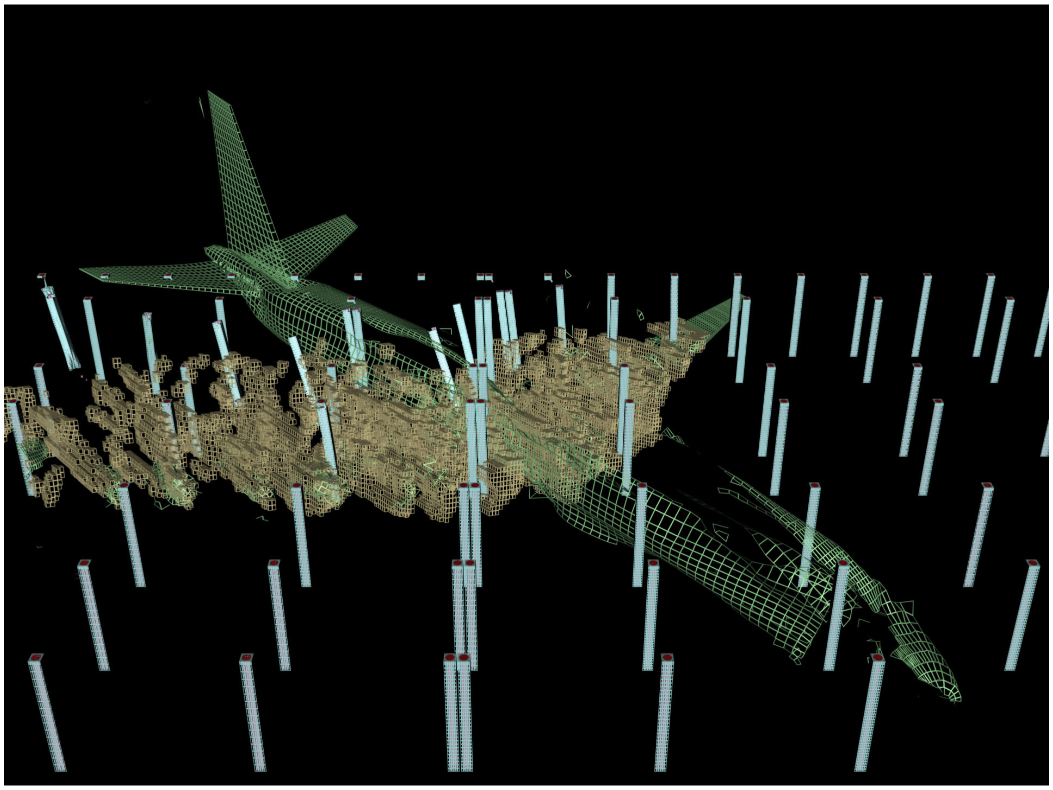


Figure 8 Wire frame visualization of the simulation results

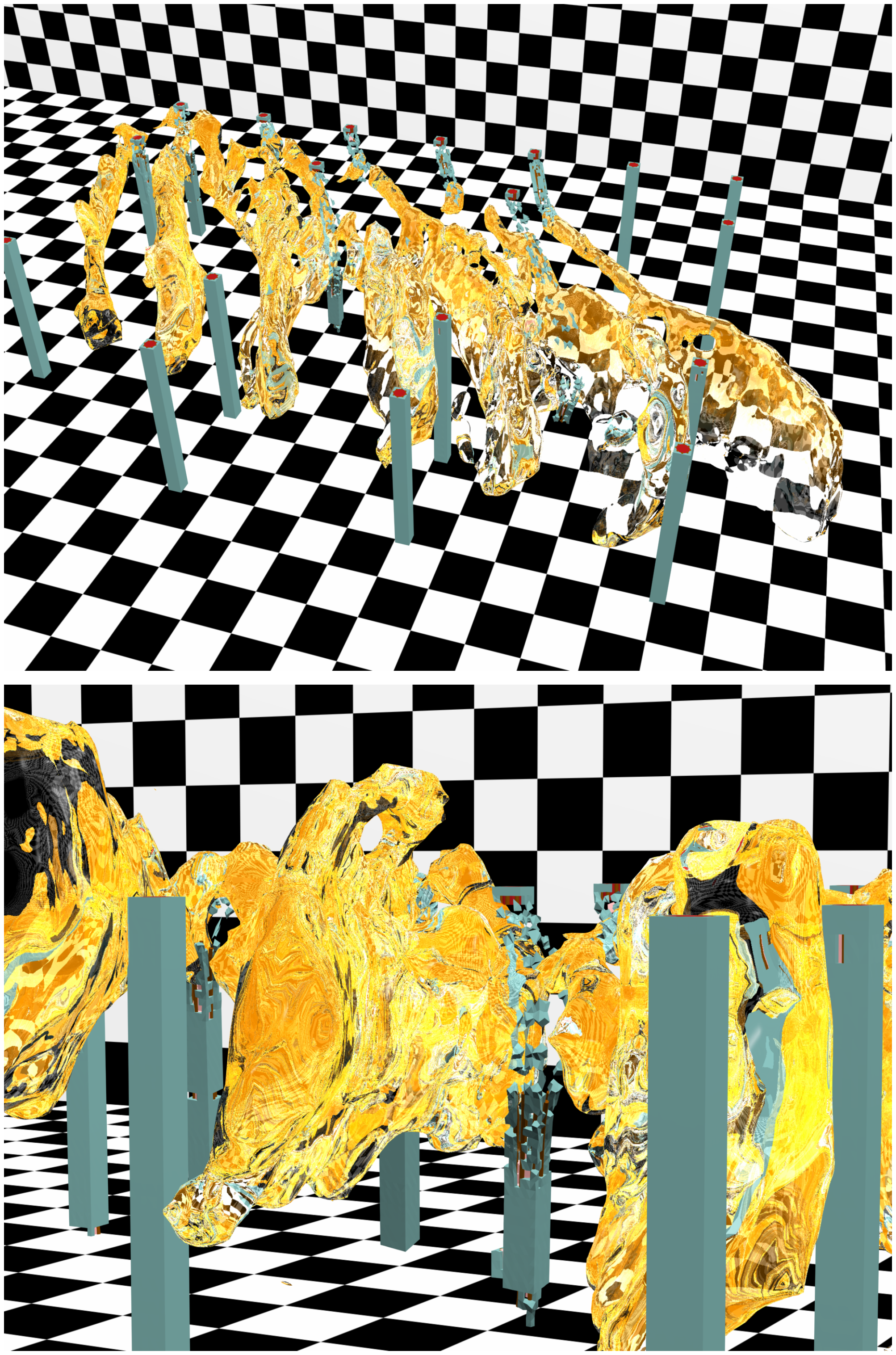


Figure 9 Liquid / column impact visualization.