CAD Visualization by Outsourcing

Voicu Popescu[1] and Christoph Hoffmann[2]

[1]Purdue University, popescu@cs.purdue.edu
[2]Purdue University, cmh@cs.purdue.edu

ABSTRACT

This paper describes an approach for state-of-the-art visualization in Computer Aided Design and Applications (CAD) based on connecting CAD to computer animation. Instead of replicating functionality in rapidly outdated visualization modules for each CAD software system, CAD scene is imported into an animation software system where state-of-the-art visualizations are produced. The approach of outsourcing CAD visualization to animation has the advantages of simplicity—it is easier to translate CAD scene descriptions to formats suitable for import into animation software systems than to re-implement state-of-the-art visualization techniques—, of automatic updates—as animation software advances at a rapid pace driven by popular applications in entertainment, the advances automatically benefit CAD—, and of promoting CAD visualization research—collaborations between CAD and visualization experts can focus exclusively on devising novel visualization techniques as opposed to making state-of-the-art visualization available to CAD. The approach is exemplified on the production of a high-quality visualization of the September 11 Attack on the Pentagon, based on importing FEA output data into an animation software system.

Keywords: visualization importance, CAD output imported into animation software, state-of-the-art visualization.

1. INTRODUCTION

Visualization is an essential tool in Computer Aided Design (CAD), used at all stages of the life of a product. Visualization enables designing, debugging, validating, marketing, maintaining, repairing, updating, and recycling products effectively and efficiently. Recognizing the importance of visualization, most CAD software systems provide visualization functionality. However, typical visualization modules of CAD software systems fall short of realizing the true potential of visualization.

One reason for this is that graphics and visualization progresses at a rapid pace driven by popular applications in entertainment such as 3D games and movies. Graphics and visualization techniques have reached a great level of sophistication that remains unmatched by the visualization modules of CAD software systems. Visualization modules of CAD systems are typically one or more steps behind the state-of-the-art in graphics and visualization, and the gap continues to widen.

A second reason is that visualization modules of CAD software systems are now asked to produce images to be used outside the narrow circle of CAD specialists. If the original goal of such modules was to visually present quantities of highest relevance to CAD specialists, now visualization is asked more and more to produce images that describe the CAD scene realistically as to be readily assimilated by marketers, maintenance crews, clients, decision makers, and the public at large. A notable weakness of visualization modules of CAD systems is the inadequate support for realistic visualization of the CAD scene such that the non-expert can quickly recognize the entities involved by association with their real world counterparts.

On the other hand, commercial computer animation software systems, such as Maya [8] or 3ds Max [1], track the progress of graphics and visualization state-of-the-art with little latency. Animation systems have reached a stage where they can render complex 3D scenes to produce images that can be easily mistaken for photographs (see Fig. 1.). Animation systems provide state-of-the-art tools for creating and controlling cameras and lights, for describing the visual properties of materials in minute detail as to match precisely those of virtually any real world surface, for managing the complexity of 3D scenes, for flexible rendering that allows trading accuracy and efficiency, and for exporting the 3D scene in formats suitable for interactive visualization.

This paper describes an approach to state-of-the-art visualization in CAD based on connecting CAD to computer animation. Instead of replicating functionality in rapidly outdated visualization modules for each

Fig. 1: Photorealistic image rendered using animation software. © Proper Graphics, www.propergraphics.com.

CAD software system, CAD data is *imported* into an animation software system where state-of-the-art visualizations are produced. The approach of importing CAD data into an animation software system that produces state-of-the-art visualizations has at least three major advantages.

First, implementing a custom importer is a much simpler task than implementing state-of-the-art visualization capabilities into the CAD system. The importer essentially translates from one format to another, and, once the hurdle of understanding the two formats is overcome, developing the importer becomes a rather routine software development task, which does not require artistic talent or intimate knowledge of CAD or visualization algorithms.

Second, the animation system tracks almost in real-time the graphics and visualization state-of-the-art. As soon as an advance is assimilated into the animation system, it becomes available de facto to the CAD world, for the small cost of occasional simple updates to the translator.

Third, making the state-of-the-art in visualization automatically available to the CAD field increases the collaboration between visualization and CAD researchers in a way that stimulates research in CAD visualization. As most of those CAD/visualization researchers that ever collaborated or attempted to collaborate with visualization/CAD researchers will attest, an important challenge that needs to be overcome is to make the collaboration useful for both parties. It is usually the case that the bulk of the collaboration effort is spent making well known visualization methods available to the particular problem at hand. If instead all well known visualization methods are available by default to the CAD field, the collaboration with visualization researchers will focus on devising novel visualization techniques that *advance* rather than *replicate* the state-of-the-art.

It is important to note that it is not proposed that all visualization capabilities of the CAD system be removed. For most CAD systems basic visualization is an important part of the interface, and that should not change. For example, a CAD tool for designing a building should still reflect the changes to the design in real time in several orthographic and perspective viewports. However, the CAD tool should not be expected to produce a photorealistic walkthrough of the building or to integrate the new building within its future surroundings in a real world city. As another example, an FEA tool should still be equipped with lightweight pre- and post-processors that allow for the rapid inspection of the results of a simulation, but the post-processor should not be expected to produce visualizations that are eloquent to the non-expert.

2. APPROACH OVERVIEW

Given a CAD software system and a computer animation software system, one approach for visualizing the CAD scene is to save the CAD scene in a format F_0 and to import it into the animation system from a format F_1. In some cases the CAD and the animation systems can import / export the same format, so F_0 and F_1 are the same. For example AutoCad files import directly into 3ds Max. In other cases there is no intersection between the export formats of the CAD system and the import formats of the animation system. In such a case F_0 is different from F_1 and one needs to write a translator F_0F_1, which implies that formats F_0 and F_1 be openly documented.

However, this approach is not always sufficient for establishing a comprehensive, robust, and scalable link between the CAD and the animation systems. Consider for example the case of visualizing LS-DYNA [6] output in 3ds Max. The LS-DYNA database format is publicly available and one could write a translator to VRML (Virtual Reality Modeling Language [12]), which in turn is one of the formats that 3ds Max imports. However, the approach would have several problems. First, the VRML format is not sufficiently rich to express complex materials, which means that some of the LS-DYNA data would be lost. Second, VRML is a multi-purpose 3D graphics format, and, as it is often the case when generality is the object, it is suboptimal in the narrow context at hand. For example, VRML does not allow animating vertices of triangle meshes individually. This could be bypassed by fragmenting a mesh in sub-meshes whose motion is well approximated by a rigid body motion, but the solution comes at the cost of substantial loss in performance. Third, the VRML importer in 3ds Max is far from efficient, robust or complete. Commercial software typically excels at importing/exporting its data in its own format, and comparatively fewer resources are allocated to developing high performance importer/exporters for other formats. For example, it has been our experience that some scenes exported from 3ds Max in the VRML format cannot be loaded back into 3ds Max.

A better approach for visualizing a CAD scene into an animation system is to either extend the CAD system with a custom exporter that outputs the scene in the native format of the animation system, or to extend the animation system with a custom importer that inputs the CAD scene in its native format. The second option—namely developing a custom importer into the animation system—is usually the one to be preferred since

- Commercial animation systems have an open software architecture which allows extending their functionality, without exposing the source code.
- Animation systems employ a complex scene file format.
- Animation systems have a well developed and documented Software Development Kits (SDKs), which provide good support for the development of custom modules.

Fig. 2 gives an overview of the approach of producing high-quality visualizations of CAD scenes within an animation system based on a custom importer. The importer pipeline takes as input the CAD scene stored in one or several files and creates the corresponding scene in the animation system. The importer pipeline is described in Section 3. Once the scene is imported into the animation software it is assigned graphics and visualization materials which enable a high-quality visualization. This step is described in Section 4. In many cases the eloquence of the visualization is enhanced by integrating the CAD scene into the surrounding scene which provides the necessary context without having to increase the complexity of the CAD scene unnecessarily. Approaches for modeling the surrounding scene using both tools provided by the animation system and external tools are described in Section 5. Once the CAD scene is registered in space and time with the surrounding scene, a high quality visualization is produced (output *a*). The animation system also serves as a platform from where simplified scenes suitable for interactive visualization are exported (output *b*). The various export options are discussed in Section 6.

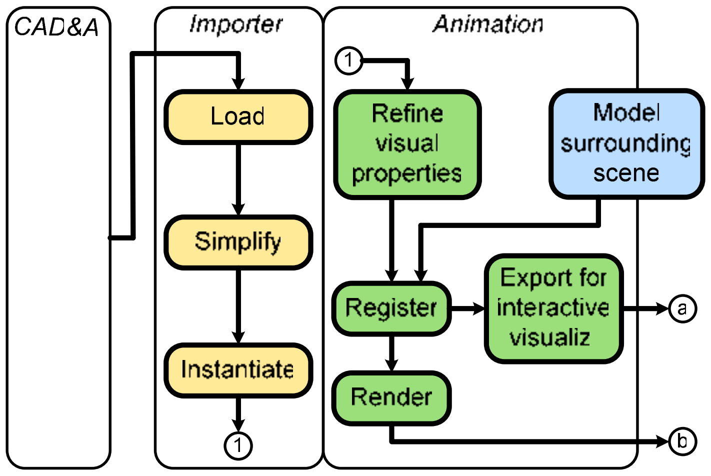

Fig. 2: Approach overview.

The approach was used to produce a high quality visualization of the September 11 Attack on the Pentagon [3, 11]. A large-scale FEA simulation of the aircraft impacting the Pentagon building was computed in LS-DYNA and then visualized in 3ds Max using a custom plug-in that imported the LS-DYNA output database into 3ds Max. The case study is used in the following sections as illustration.

3. IMPORTER PIPELINE AND IMPLEMENTATION

This section describes the stages of the importer pipeline and then gives a general description of the process of extending an open software architecture through a plug-in.

3.1 Pipeline Stages

The importer pipeline has 3 stages that convert the CAD scene into an animation system scene that is equivalent from the visualization standpoint (see Fig. 2 again). In the first stage the CAD scene files are parsed and loaded into intermediate data structures internal to the importer. Parsing the format of a well described CAD scene file is routine. Sometimes the file format is not fully specified so that this step is somewhat tedious. Not all data is loaded. For example values of material parameters that do not influence visual properties are skipped.

In the second stage all data that is irrelevant from the visualization standpoint is simplified. For example, intermediate node positions for states s_1, s_2, …, s_n can be discarded if the node trajectory is well approximated by the node positions at states s_0 and s_{n+1}. Deciding what trajectory simplification is acceptable depends on the application. A non-lossy simplification will remove only positions perfectly aligned with the segment defined by the first and last states. A lossy simplification could tolerate deviations up to a user chosen ε. Another example of data that is redundant from the standpoint of visualization are internal faces in a cell decomposition. If a face is never visible during the visualization, it can be safely removed to lower the geometric complexity of the scene. In the example of importing FEA data, removing the internal faces of a large structure composed of hexahedral elements reduces the face count considerably. Deciding which face is internal and which is not has to be done on the last state if eroding elements make the large structure break into pieces exposing originally internal faces.

A third source of redundant data is excessive tessellation. The tessellation level dictated by CAD constraints (e.g. manufacturability, uniformity of elements in FEA simulation) can be substantially higher than that needed to produce a smooth visualization. This is particularly true since graphics and visualization techniques such as Phong shading [9] have been developed to produce the appearance of a smooth surface even for coarse tessellations. To avoid wasting visualization resources, the importer should attempt to load higher order surfaces and then let the animation system set the tessellation level according to each rendered frame. When that is not possible, classic geometry simplification techniques can be used to replace adjacent coplanar or nearly coplanar faces with a larger face.

The two pipeline stages discussed so far load the relevant data into custom data structures internal to the importer, and are independent of the actual animation system used. The third stage instantiates animation system structures according to the importer data structures to create the animation system scene. For example, a set of connected FEA beam elements with a circular cross section is modeled as a spline with several segments and with a relatively high level of tessellation (e.g. 12). Beam elements with a square cross section are modeled by reducing the tessellation factor to 4 and by setting the shading model to flat (as opposed to smooth). If the beam elements have a more complicated cross section (i.e. an I or a T cross section), the elements are modeled with a triangle mesh. A double-sided mesh can model thin I-shaped beams at the cost of 6 vertices per node. Thick I-shaped beams require 12 vertices per node.

It is important that the importer defines its own data structures. Custom data structures tailored precisely for the type of data to be loaded make the import efficient, avoiding having to use the often heavy data structures of the animation system. The custom data structures also accelerate the development of the importer as the developer does not need to be intimately familiar with the animation system data structures to debug the implementation efficiently. Finally, the importer data structures provide a clear separation between the front end of the importer that implements the task of loading and simplifying the CAD scene and the back end of the importer which populates the animation system scene. The front end is reused when retargeting the importer to additional animation systems. For example developing an LS-DYNA to Maya importer based on the 3ds Max importer requires only a fraction of the effort of the initial development.

The output of the importer pipeline (see label 1 in Fig. 2) is a raw animation system scene created from the CAD scene. Fig. 3 shows a state of the Pentagon LS-DYNA simulation imported into 3ds Max. Although the import pipeline itself does not contribute directly to improving the visual qualities of the scene, it brings the CAD scene into the animation system where its visual properties are refined.

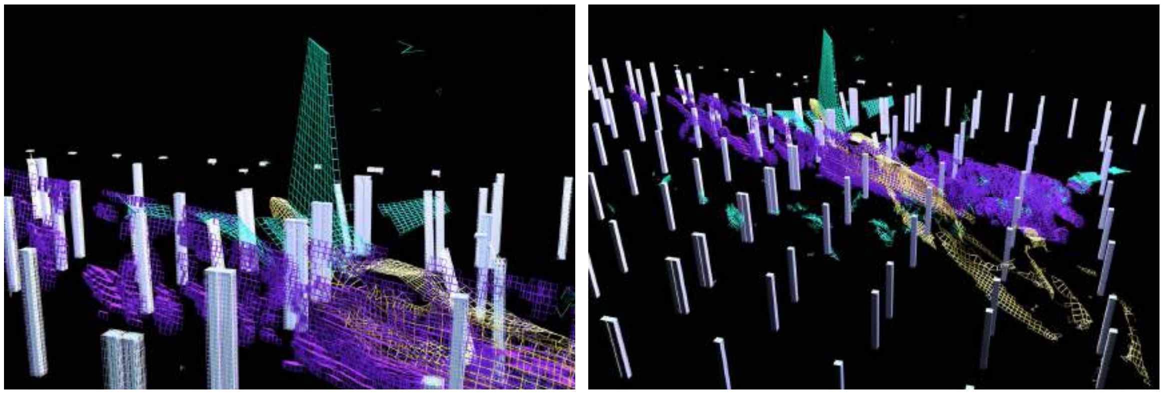

Fig. 3: Example of importer pipeline output—FEA data imported into animation system.

3.2 Software Development Considerations

The importer is implemented as an animation system plug-in. Modern animation systems such as 3ds Max and Maya have an open software architecture extendable through plug-ins. A plug-in is a custom module of the animation system, developed with the SDK that is made available to developers. The plug-in is usually written in a widely used high level programming language such as C++. The plug-in source code is built into a dynamically linked library file which is loaded into the animation system at startup. There are several types of plug-ins, according to the type of functionality they provide. The plug-in type best suited for extending the importing capabilities of the animation system is selected. The plug-in is implemented by a class that inherits from the generic class for the plug-in type and specializes the virtual methods according to the CAD scene imported.

For example, 3ds Max SDK defines 13 plug-in types, 12 of which are listed in Tab. 1. *File Import* is a plug-in type specifically designed for importing from novel formats. There are 12 virtual methods for the *File Import* plug-in class. A constructor and a destructor allow building and releasing data structures internal to the plug-in. A set of 9 methods allow gathering data about the plug-in such as file extensions supported, author name, copyright message, short and long text descriptions, and version. One method defines 3ds Max commands to be performed after the plug-in concludes such as choosing appropriate views for each viewport. The actual import pipeline is implemented by a method suggestively dubbed *DoImport*, which implements the import pipeline.

	Plug-in type		**Plug-in type**		**Plug-in type**
1	*Atmospheric*	5	*Image processing*	9	*Renderer*
2	*Controllers*	6	*Materials*	10	*Space Warps*
3	*File Import/ File Export*	7	*Object Modifiers*	11	*System*
4	*Image loading and saving*	8	*Procedural Objects*	12	*Texture Maps*

Tab. 1: Plug-in types defined by the 3ds Max SDK.

In order to develop the plug-in efficiently the front-end, which is independent of the animation system, should be tested in a separate executable. Once the front-end performs as expected, the entire plug-in is tested within a much slower debugging loop which includes launching the animation system and invoking the newly added functionality by importing a CAD file type from the animation system user interface.

4. VISUALIZATION REFINEMENT

Animation systems have reached a high-level sophistication and allow synthesizing images of stunning quality and realism. A detailed animation system tutorial is beyond the scope of this paper. The essential ingredients for producing a visualization are geometry, animation, materials, lights, and cameras.

The animated geometry has already been created by the importer. Animation systems offer powerful tools such as tools for computing normals, for further simplifying geometry to meet face count budgets, and for smoothing faces and normals. However, any modification to the geometry or its animation alters the imported CAD scene. Therefore modifications can be performed only within the limits of the desired level of fidelity.

Modern animation systems offer sophisticated material editor tools which allow creating materials whose appearance matches that of virtually any real world scene. Many materials needed to visualize the CAD scene are already available in comprehensive material databases. Materials models can incorporate physical measurements such as photographs or bidirectional reflectance distribution function (BRDF) data. Some materials can be rendered at interactive rates with the feed forward graphics pipeline implemented by graphics hardware (e.g. materials based on simple direct illumination shading models, texture mapping, approximate reflections and refractions, or hard shadows), and some materials require more expensive rendering techniques such as ray tracing (e.g. materials requiring accurate reflection and refraction, soft shadows, complex BRDFs).

Animation systems also offer comprehensive support for light design and placement. Omnidirectional, directional, spot, area, skylight, free, target and combinations allow building lights with the desired general characteristics which are then refined using numerous parameters to obtain precise lighting effects.

Cameras are essential since they ultimately capture the images of the virtual world to produce video sequences that expose the characteristics of the scene and help make the points for which the CAD scene was generated in the first place. As in the case of lights, perspective, orthographic, free, target and combinations define the desired type of camera which is then further described by choosing appropriate parameter values. The animation systems provide convenient methods for animating cameras such that they smoothly travel through the scene as to reveal all important aspects. One general approach is to define key frames by choosing a few relevant viewpoints and then let the system create intermediate views by interpolation. A second general approach is to define the camera trajectory using a one dimensional geometric object such as a spline. A third possibility is to capture the camera motion in the real world using motion capture devices or trackers.

Fig. 4 shows snapshots of 3ds Max visualizations of the Pentagon simulation data. Most of the kinetic energy of the plane was concentrated in its fuel, so the visualization effort focused on liquid visualization.

5. MODELING THE SURROUNDING SCENE

It is often the case that the impact of the visualization is magnified if the CAD scene is integrated into the surrounding scene. For example, a new building should be placed in the context of the neighboring buildings, streets, and parks. Illustrating how to service a power plant by replacing a large component should also show the neighboring components and the access paths to and within the plant in order ensure accessibility and to devise the optimal approach for installing the new component.

Modeling 3D scenes is a challenging open research problem approached from the directions of computer graphics, computer vision, physics, and optical engineering. Animation systems offer support for 3D modeling based on the traditional computer graphics approach of manually instantiating, placing, and decorating geometric primitives. The approach is inefficient due to the manual work involved and relies on 3D content creators with a rare combination of artistic and technical skills.

An alternative approach is to model from acquired color and depth data. Color acquisition is a solved problem as high-resolution digital cameras are now ubiquitous. Satellite and aerial imaging provide detailed visual descriptions of large real-world scenes. Photographs can be applied to geometry by texture mapping or projective texture mapping which are classic computer graphics techniques for orthographically or perspectively mapping color data to surfaces. Depth acquisition is more challenging. A series of depth acquisition technologies have been developed, including depth from stereo [10], depth from structured light [5], and time-of-flight laser rangefinding [2, 4], each with its strengths and weaknesses (see Tab. 2.).

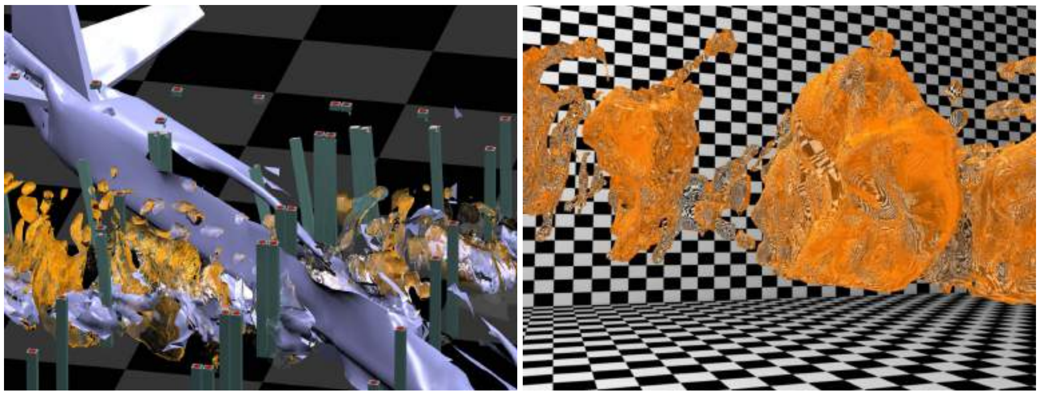

Fig. 4: High-quality visualization of FEA data, produced within animation system.

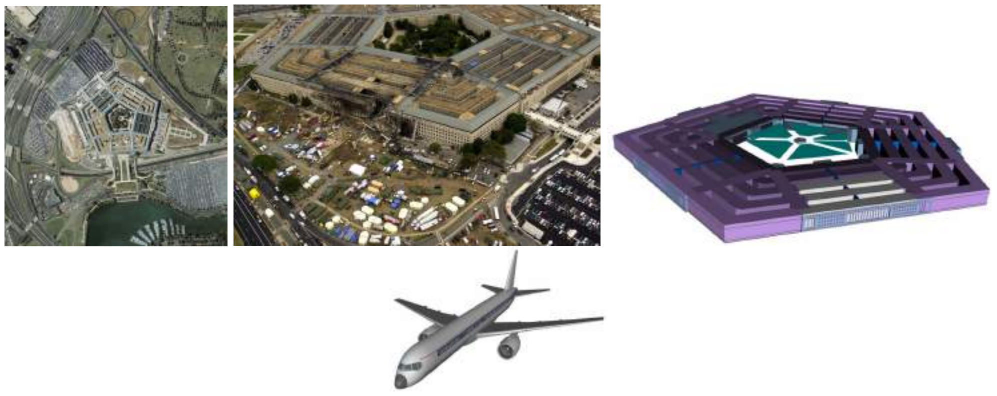

Fig. 5: Satellite (courtesy Space Imaging) and aerial (courtesy ASCE) photographs (top), and Boeing 757 (courtesy Amazing Graphics) and Pentagon building models (bottom). The images and models were used to describe the surrounding scene for the September 11 Attack on the Pentagon simulation.

Depth acquisition method	**Robustness**	**Cost**
Depth from stereo	Problems with uniformly colored surfaces or on surfaces with view dependent appearance (e.g. specular, transparent).	Fast, inexpensive, intuitive acquisition (photographs) Slow search for correspondences
Depth from structured light	Handles uniformly colored surfaces. Problems with ambient light (light pattern becomes invisible), with view dependent appearance, with dynamic scenes (lengthy sequential scanning).	Slow acquisition and modeling (registration, 3D triangulation) Expensive controllable light sources (projectors, lasers)
Time-of-flight laser rangefinding	Does not require controlling the scene lighting. Problems with view dependent appearance and with dynamic scenes.	Slow acquisition and modeling. Expensive laser rangefinders.

Tab. 2: Strengths and weaknesses of various depth acquisition technologies.

In the case of the Pentagon Attack, the surrounding scene was modeled using satellite and aerial digital photographs, as well as surface models for the plane and the Pentagon (Fig 5). It was important to model not only the scene spatially adjacent to the impact, but also to extend the timeline to the moments preceding and succeeding the impact. Fig. 6, 7, and 8 show snapshots of the pre-impact, impact, and post-impact visualizations.

6. EXPORT FOR INTERACTIVE VISUALIZATION

Animation systems produce visualizations off-line by running complex algorithms distributed across rendering farms. However, animation systems are excellent platforms from which to export simplified versions of the scene that are suitable for interactive visualization. Most animation systems allow exporting in VRML and other formats that are readily handled by interactive renderers. VRML standalone and web browser plug-in viewers allow visualizing complex scenes at interactive rates, leveraging the great power of PC graphics accelerators. Materials that require ray tracing in 3ds Max are replaced by simpler materials that can be handled by the feed-forward graphics pipeline. For example liquid is visualized interactively by alpha-blending, an approximate but fast method for rendering transparency (Fig. 9).

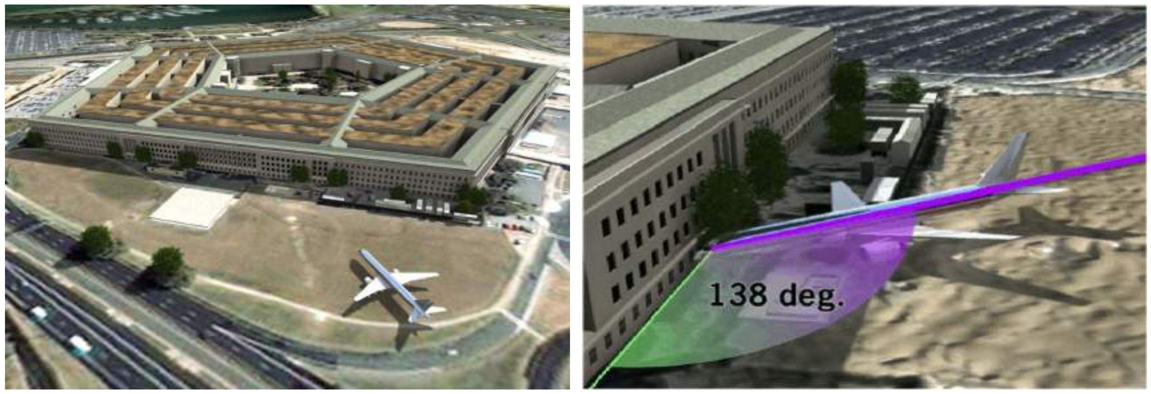

Fig. 6: Snapshots from the pre-impact visualization.

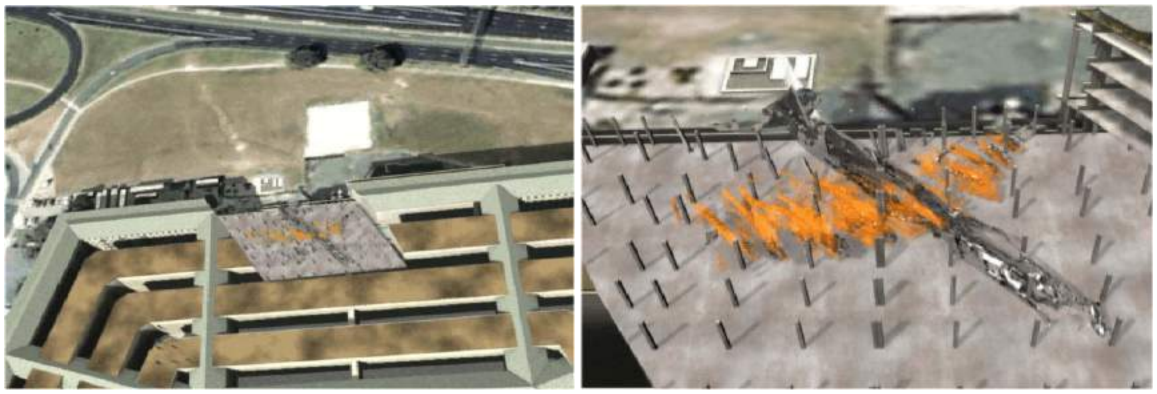

Fig. 7: Snapshots from the impact visualization. At the region of impact the top floors of the Pentagon building were removed for improved visualization.

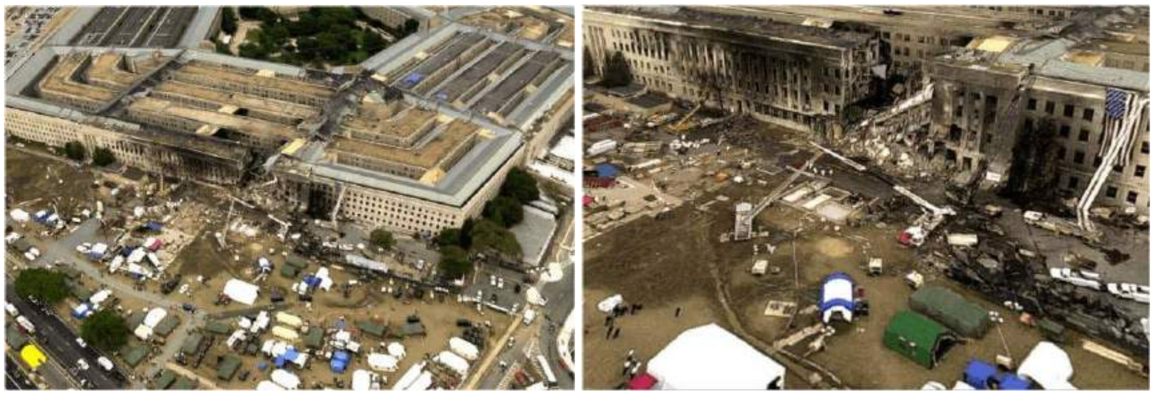

Fig. 8: Snapshots from the post-impact visualization.

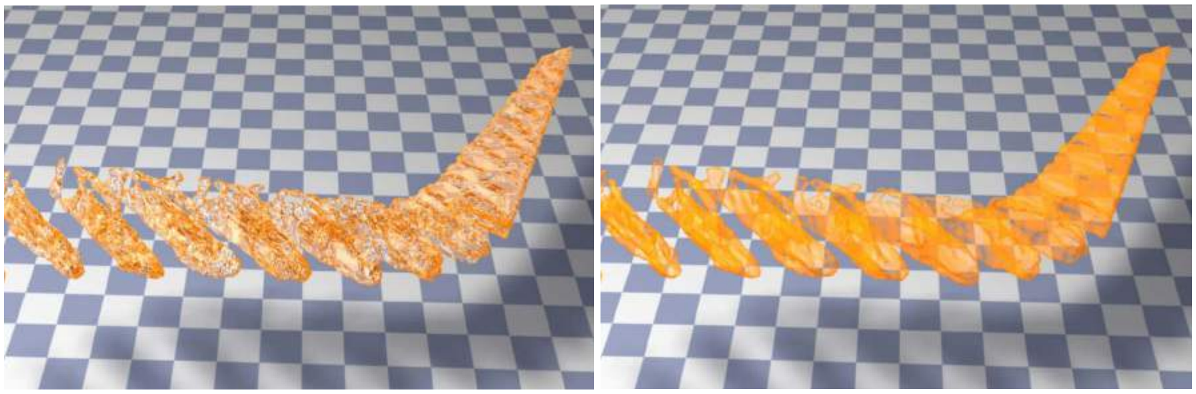

Fig. 9: Ray traced (off-line) and alpha-blended (interactive) liquid visualization.

Another example of simplification needed to achieve interactive rendering rates is replacing accurate ray traced reflections with approximate environment mapped reflections. The animation system can be used to produce quality environment maps to be used subsequently in interactive visualizations.

Several methods have been developed to improve the quality of the interactively rendered images. One method is to "bake" the high-quality appearance of surfaces into textures. Another method is to save snapshots of the scene enhanced with depth [7], which are then used as rendering primitives in interactive visualizations. Both methods convey successfully an "off-line" quality to interactively rendered images, but they limit the changes that can occur in the scene during visualization. For example one cannot change the lighting or move objects that cast shadows since that would imply re-baking the appearance textures or re-rendering the reference images.

Graphics hardware continues to progress at a rapid pace hence the quality gap between images rendered off- and on-line shrinks continually. As graphics processing units (GPUs) and the languages that program them grow in sophistication, vertex and pixel programs will grow in length and complexity. We foresee that the animation system will serve as an interface between the user of complex materials and the programmer that designs the shaders that render the material at interactive rates on the GPUs.

7. DISCUSSION

This paper proposes to produce high-quality visualizations of CAD scenes in state-of-the-art animation systems. Developing an importer is a relatively simple task which unlocks the many benefits of animation systems. The CAD specialist will not obtain the same results using an animation system as would an artistically gifted proficient user of the animation system. However, the CAD specialist will produce visualizations superior to those produced in visualization modules of CAD tools after a minimal learning effort. Moreover, the importer gives the CAD specialist the option of recruiting the help of a professional animator, which is not possible when the scene is only available in the native CAD format. The approach of outsourcing visualization has potential in domains beyond CAD including science and engineering in general. Importers could be developed in open source within a community that shares front and back ends of the importers to allow interconnecting N science and engineering software systems to M computer animation software systems at a cost proportional to $N+M$.

8. ACKNOWLEDGMENTS

The authors wish to thank Mete Sozen, Sami Kilic, Scott Meador, Jason Doty, Hendry Lim, Amit Chourasia for their help, and the Tellabs Foundation, the Computer Science Department of Purdue University, the Computing Research Institute of Purdue University, the Information Technology Department at Purdue, the Envision Center of Purdue University, the National Science Foundation, IBM, Intel, and Microsoft for their support.

9. REFERENCES

[1] 3ds Max, http://www.autodesk.com/3dsmax, Autodesk.
[2] DeltaSphere 3000, http://www.deltasphere.com/, DeltaSphere Inc.
[3] Hoffmann, C.; Popescu, V.; Kilic, S.; Sozen, M.: The Pentagon on September 11th, IEEE Computing in Science and Engineering, January/February, 2004, 52-60.
[4] Leica ScanStation, http://www.leica-geosystems.com/, Leica Geosystems.
[5] Levoy, M, et al.: The Digital Michelangelo project: 3D scanning large statures, ACM SIGGRAPH, 2000, 131-144.
[6] LS-DYNA, http://www.lstc.com/, Livermore Software Technology Corporation.
[7] McMillan, L.; Bishop, G.: Plenoptic Modeling: An Image-Based Rendering System, ACM SIGGRAPH, 1995, 39-46.
[8] Maya, http://www.autodesk.com/maya, Autodesk.
[9] Phong, B.T.: Illumination for computer generated images, Ph.D. thesis, University of Utah, Salt Lake, UT, 1973.
[10] Pollefeys, M.; Van Gool, L.: From images to 3D models, Communications of the ACM, (45):7, 2002, 50-55.
[11] Popescu, V.; Hoffmann, C.: Fidelity in Visualizing Large-Scale Simulations, Computer Aided Design, (37), 2005, 99-107.
[12] VRML, http://www.web3d.org/x3d/specifications/vrml/, Web 3D Consortium.