

Airport Viz – A 3D Tool to Enhance Security Operations

Daniel B. Koch, Ph.D.

Oak Ridge National Laboratory; Oak Ridge, Tennessee, USA

Abstract

In the summer of 2000, the National Safe Skies Alliance (NSSA) awarded a project to the Applied Visualization Center (AVC) at the University of Tennessee, Knoxville (UTK) to develop a 3D computer tool to assist the Federal Aviation Administration security group, now the Transportation Security Administration (TSA), in evaluating new equipment and procedures to improve airport checkpoint security. A preliminary tool was demonstrated at the 2001 International Aviation Security Technology Symposium. Since then, the AVC went on to construct numerous detection equipment models as well as models of several airports. Airport Viz has been distributed by the NSSA to a number of airports around the country which are able to incorporate their own CAD models into the software due to its unique open architecture. It provides a checkpoint design and passenger flow simulation function, a layout design and simulation tool for checked baggage and cargo screening, and a means to assist in the vulnerability assessment of airport access points for pedestrians and vehicles.

Introduction

Creating a 3D application can be a daunting task even for experienced programmers. When the intended operator is only slightly familiar with using any form of visualization for running computer simulations, the task becomes even more formidable. Fortunately, there now exist development environments for creating 3D visual simulation tools with easy-to-understand user interfaces. This paper describes one such project where an open-ended visual simulation tool was developed for the NSSA [1] and the TSA to simulate airport security checkpoint layouts, checked baggage inspection systems, and other airport security operations. Visual simulation tools can be a valuable aid for exploring new technologies and procedures prior to fielding.

The term *virtual reality* often conjures up images of computer games or Hollywood films. Since the term is becoming somewhat overused, many professionals in the field now prefer the term *virtual environment* or *visual simulation* [2, 3]. Regardless of the terminology the aim remains the same - to use immersive software techniques to add a level of realism to an application. The benefit of such an approach to airport security and the methodology used to implement the tool are the topics of this paper.

Project Description

The main goals of the project included:

- Developing an extensive library of 3D computer models of various detection equipment.
- Constructing 3D computer models of selected airports including passenger security and hand baggage inspection areas, checked baggage areas, and pedestrian and vehicle access points.
- Writing a 3D visual flow simulation for the above models.

At the time of this writing, more than 100 computer models of various detection devices such as metal detectors, x-ray machines, and chemical analyzers have been developed. Custom models of the Knoxville, Atlanta, and Seattle airport checkpoint areas have also been completed. A software layout and passenger flow simulation tool was initially written to assist airport checkpoint security designers. Complementary tools were then developed for checked cargo/baggage screening and employee/vendor access to restricted areas of the airport. These tools were finally consolidated into a single program called *Airport Viz*.

System Design

A number of initial goals were defined for the software tool to increase its widespread adoption. First the tool had to be easy to use by someone with average computer skills. This had implications for the user interface design. Second, it had to be scalable, being able to run on a laptop at the low end and on a fast graphics workstation at the high end of the hardware spectrum. Cost considerations also played a part. It was decided that the runtime environment had to be inexpensive and not require a distribution license. Finally, the tool had to stand alone as a means for letting the end-user create new simulations so that they did not have to go back to the original developers for each new variation.

Figure 1 shows the screen of a typical simulation. The button bar may be seen at the lower right side of the window. All of the major commands may be executed from the button bar. Auxiliary keyboard commands may be executed during certain operations. Mouse movement controls the user's perspective while the simulation is running. This allows the security designer to interactively view the simulation from any angle, which may be useful in evaluating visual obstructions during screening operations.

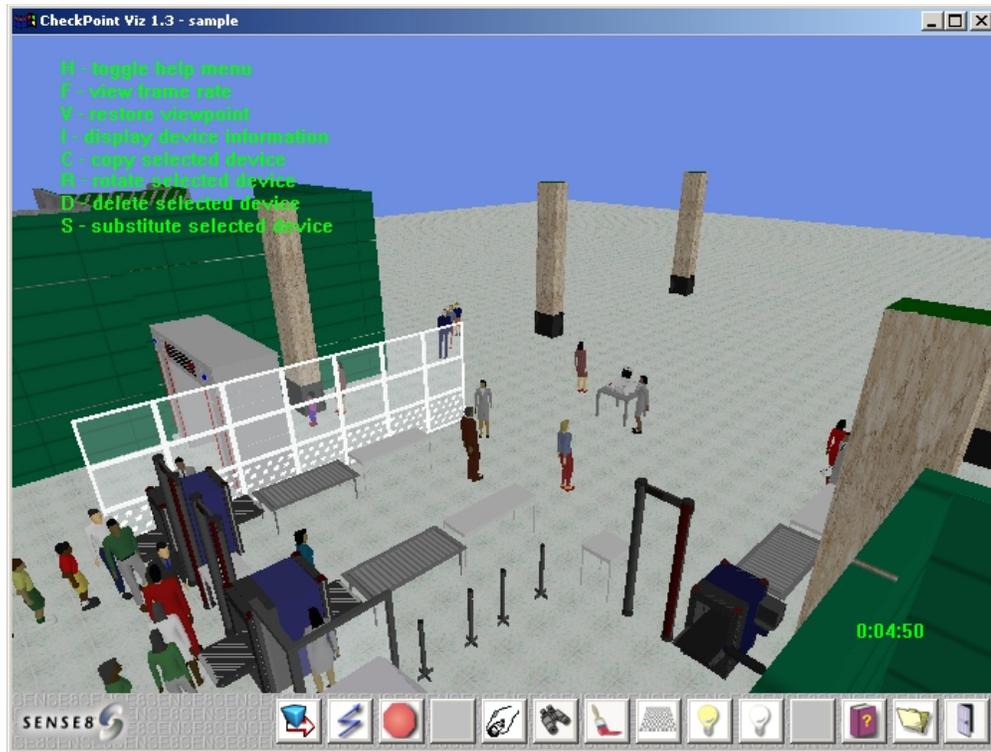


Figure 1. CheckPoint Viz Simulation Screen

Major operations required by the user include importing different detection devices into the scene, placing them at various locations, defining a path or paths to be taken by passengers, and specifying the passenger traffic loads. Once the simulation starts, the user may navigate around the scene to view passenger traffic from any perspective. Simulation speed is controllable. Passenger generation and device operation follow probabilistic models. A layout may be saved and recalled for later use. The application has been tested on laptop computers and dedicated graphics workstations with good results.

Software Development

With projects of this size and complexity, it was imperative that a structured approach be taken for software development. This included using tools both for the management of the project as well as for the actual code writing. To accomplish this, software engineering tools were selected and modified to work within the graphical *WorldUp* environment. Of particular utility were those methods employed by the so-called *Cleanroom Software Engineering* process [4-7]. *Spiral development* or *incremental development* are two similar terms used to describe this process. The methodology has as its goal defect-free code.

The project spanned nearly 36 months and generated approximately 60 megabytes of code. From the beginning, staff changes were seen as inevitable and raised concerns over ongoing support of the software produced. Therefore, the development team set goals for the project, including the need for a

self-documenting design, embedded training aids, and most of all, low software maintenance requirements.

Program Operation

Referring back to Figure 1 it can be seen that the user interface is very simple. User control is restricted to the use of a button bar for basic operations and keyboard commands for certain options. All of the available commands are always visible unlike drop-down menu systems. Pop-up tool tips help the user associate button operations with their respective icons.

Initially a user starts off with a blank workspace and imports an airport model. It is then populated with various detectors and architectural elements using a menu system organized according to categories. In this respect the tool behaves similarly to a CAD program. Using the mouse and arrow keys, the user lays out the checkpoint design. Figure 2 shows a screen capture of information displayed while importing a typical device model. An end user may add to the model database if they have the capability of building their own 3D models.

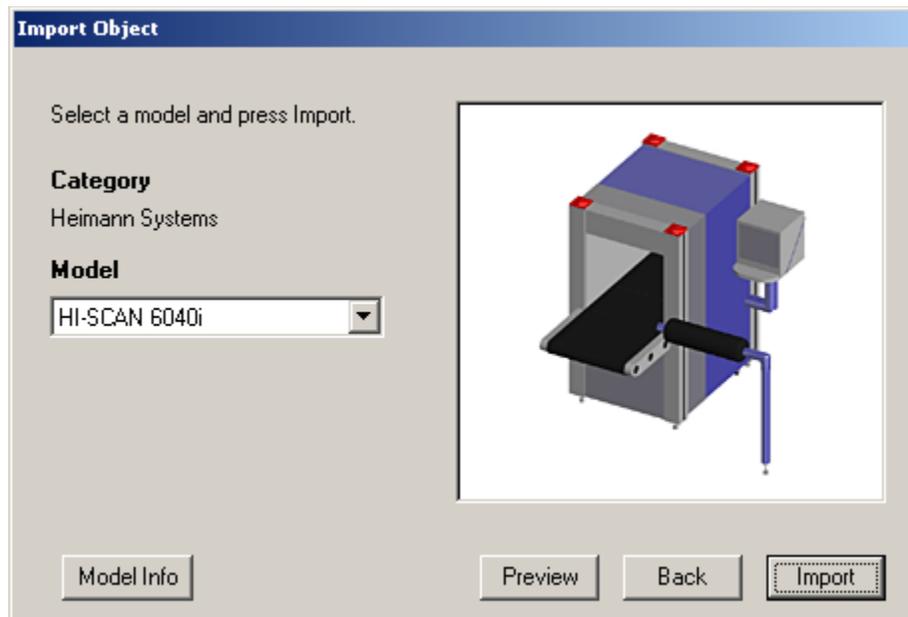


Figure 2. Model Import Screen

The next step involves defining paths that the simulated passengers take through the checkpoint layout. Users may define several groups of paths for the same physical layout to experiment with different passenger processing schemes. Alarm handling paths may also be defined to cover the situation where additional passenger screening is needed such as after triggering a metal detector for example. Figure 3 illustrates what the user sees during path definition.



Figure 3. Path Definition Screen

Finally, the simulation is ready to run. The user begins by specifying various passenger parameters such as traffic load and walking speed. These values feed the underlying probabilistic model. The frame time can also be specified allowing the user to run the simulation in real time or faster/slower as desired. Once a simulation has run its course a window showing statistics for the passenger flow is displayed. The data may then be saved to a file for further analysis. Figure 4 shows the results of one such simulation run.

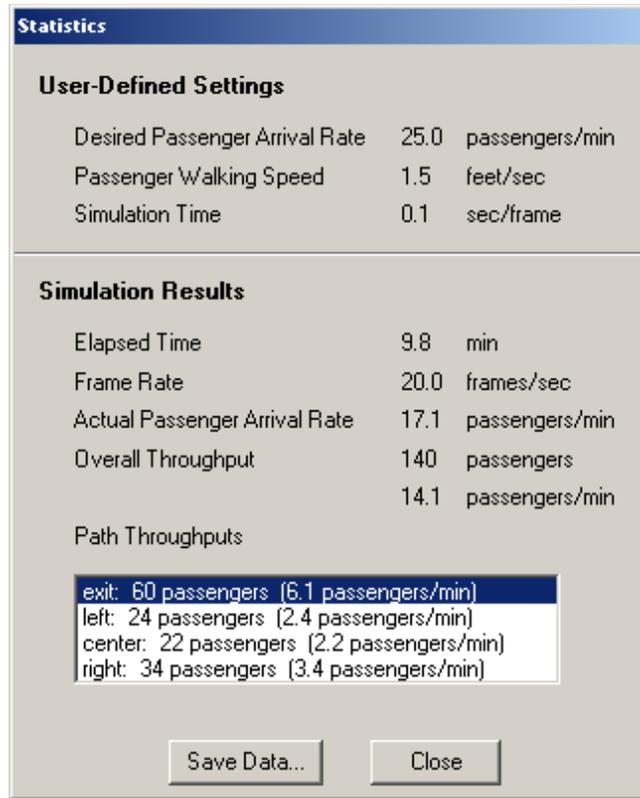


Figure 4. Statistics Screen

While the example above highlights a passenger checkpoint design, the software allows any sort of flow simulation which includes checked baggage operations and passenger/vehicle airport access.

Conclusions

The underlying 3D visualization technology used in developing the software tool described here may be extended to other complementary areas. Now that the placement and flow simulation has been developed, it is possible to interact with it in an immersive setting. This would involve the use of peripherals such as a head mounted display and pinch gloves with head and hand trackers. An operator or security person could experience the visualization as though they were inside it. The equipment and airport settings would appear life-size thus giving the user many more cues as to the desirability of a particular layout. In addition, checkpoint operators could actually be given realistic detection equipment training in this manner.

References

- [1] National Safe Skies Alliance, <http://www.sskies.org>
- [2] J. Vince, *Virtual Reality Systems*, Harlow, UK: Addison-Wesley, 1995.
- [3] L. Davis, et. al., "Enabling a Continuum of Virtual Environment Experiences", *IEEE Computer Graphics and Applications*, Vol. 23, No. 2, pp. 10-12, March/April 2003.
- [4] H.D. Mills, "Stepwise Refinement and Verification in Box-Structured Systems", *IEEE Computer*, Vol. 21, No. 6, pp. 44-54, June 1988.
- [5] A. Hausler, R.C. Linger and C.J. Trammell, "Adopting Cleanroom Software Engineering with a Phased Approach", *IBM Systems Journal*, Vol. 33, No. 1, 1994.
- [6] R.C. Linger, "Cleanroom Process Model", *IEEE Software*, Vol. 11, pp. 50-58, March 1994.
- [7] S.J. Prowell, C.J. Trammell, R.C. Linger, J.H. Poore, *Cleanroom Software Engineering, Technology and Process*, Reading, MA: Addison-Wesley, 1999.

Acknowledgments

The author would like to acknowledge the contributions of several students and colleagues to the development of Airport Viz including (in alphabetical order): Susan Beck, Marcus Dutton, Qi Lin, Neena Nambiar, Kim Nylander, Matthew Washer, Andrew Wilson, and Dayu Yang. Dr. Koch is currently a senior R&D staff member at the Oak Ridge National Laboratory where he is creating a software framework for combining discrete-time simulation with geospatial data in a virtual environment for homeland security applications. He also serves as Deputy Director of the joint *UT/ORNL Center for Homeland Security* and is an Adjunct Associate Professor at UTK.