A Virtual Reality Framework to Optimize Design, Operation and Refueling of GEN-IV Reactors (DE-FG07-03ID14501)
Final Report, Year 1-3 (and no-cost-extension, year 4) (2003-2007)

Rizwan-uddin
Department of Nuclear, Plasma and Radiological Engineering
University of Illinois at Urbana-Champaign

Executive Summary

The project titled above has been completed according to the goals set out in the original proposal. Several papers have been presented at national and international meetings and have appeared in respective proceedings of these meetings. An international workshop co-sponsored by UIUC and EPRI and a follow-on meeting were also held on the campus of University of Illinois at Urbana-Champaign—attended by academia, research organizations, utility personnel—to discuss the potential of VR and best approaches to accomplish these goals. When the proposal was written in 2003, virtual reality was a relatively unknown term in the field of nuclear engineering. Since then, thanks to the work carried out under this grant—and a few other groups around the world—several papers are presented at the ANS annual meetings each year as well as at ANS topical meeting on Human Machine Interface and Instrumentation and Control conferences. Joint efforts with contributions from academia, industry, regulators and research laboratories have led to the identification of specific areas in which this technology can contribute toward improving nuclear energy production systems. Specifically, the work carried out under this grant has led to the development of an infrastructure that allows exploration of different potentials. Hardware have been identified and used. In addition, software tools have been identified and used to facilitate the development of appropriate models. Several modifications have been introduced in the software packages to tailor them for nuclear specific applications. For example, capability to display and *visualize* radiation in virtual environment has been added for ALARA related training purposes. The *health monitor* feature in a computer game development platform is successfully adapted to be used as a dosimeter. Tools identified and developed in this project have been used to develop and test nuclear specific virtual models for design optimization and training purposes. These were for example used to optimize the control room layout of a nuclear power plant which is studying the issues related to the transition from the analog to a digital control room. The follow-on meeting mentioned above was scheduled to test the capability of such VR models as well as to test the half-analog-half-digital control room in the virtual environment. Personnel from a utility participated in the exercise.

At the time the proposal was written (2002) it was hoped that within the time frame of this project a GEN-IV reactor would be selected, and hence some scenarios specific to that chosen design would also be simulated. However, due to the fact that a GEN-IV design has not yet been selected, specific exercises conducted were more generic than initially anticipated. These exercises can however be relatively easily tailored to suit the design ultimately selected, or for modification/optimization studies even in GEN-II, GEN-III and GEN-III+ reactors.

I. INTRODUCTION AND BACKGROUND

Many GEN-IV candidate designs are currently under investigation. Technical issues related to material, safety and economics are being addressed at research laboratories, industry and in academia. After safety, economic feasibility is likely to be the most important criterion in the success of GEN-IV design(s). Lessons learned from the designers and operators of GEN-II (and GEN-III) reactors must play a vital role in achieving both safety and economic feasibility goals. Though there are numerous lessons, this project focused on two of them.

It is well known that despite the fact that no new reactors have come online, the total electrical power generated by nuclear power plants (NPPs) in the US over the last 20 years has continued to increase. Longer core life has been the primary reason. Two other important factors that have contributed to an increase in total electricity generated by nuclear are: 1) significant reduction in refueling time over the last twenty years; and 2) better operator training that has led to reduction in planned and un-intended downtime of GEN-II reactors. Economic feasibility of GEN-II reactors would be much worse today if it was not for the improvements in the overall availability of NPPs due to these two factors. Without any doubt, reducing the number of refueling as well as reducing the downtime during refueling have played an extremely important role in making GEN-II reactors economically competitive in today's de-regulated environment. This is reflected in several proposed designs with a much longer core life (as much as 30-40 years) and in continued efforts to reduce refueling time.

In addition, reduction in safety and maintenance related shutdowns from the early days of PWR and BWR operations has also contributed to increased availability of NPPs. Realizing that the total cost associated with the downtime, in many cases in NPPs, can be much more than the cost of the repaired part, there is ample reason to believe that significant gains can be achieved by minimizing the downtime due to repair and replace operations. In addition to repair and maintenance, downtime also results due to accidents. Minimizing number of accidents that lead to operations at lower power or to complete shutdown, is obviously a desirable goal. [The "political" cost of accidents to nuclear power is probably much higher than the economic cost.] Most nuclear accidents have been attributed to human factors (errors). Hence, in addition to an optimized design for refueling, better personnel training for online and offline operation and maintenance, and better training to minimize both dose (ALARA) and accidents can also play an important role in the optimization of GEN-IV reactors. Other industries have also faced similar challenges. Corresponding issues in other industries have been addressed by taking better advantage of the latest technological developments.

Many technologies that have helped the automobile, aeronautics, civil and, to certain extent, chemical industries in becoming more efficient revolve around better use of computing power that has become available over the last 2-3 decades. Computing power is already playing a significant role in better simulation of the physical processes leading to lower conservatism, and hence improving the efficiency of NPPs. This is evident from the large body of work on coupled neutronics-thermalhydraulics simulations, much more detailed simulation of the core physics, shift toward transport and Monte Carlo methods instead of diffusion theory, etc.

However, other technologies that are now available due to increased computing power, such as computer aided design and manufacturing, 3D modeling, etc., have also allowed significant improvements in the design departments of the automobile, aeronautics, civil and chemical industries. Specifically, these technologies have played a significant role in planning layout and operator training. A case can be made of the compact automobile designs of the early days that led to cramped conditions under the hood, which in turn made repairs much more time-consuming and expensive. Extreme examples include necessity to remove several components before a $5 belt could be replaced. In comparison, newer designs, carried out with the help of CAD software, lead to much *cleaner* layout of the parts, and a "mechanic-friendly" environment under the hood.

In the case of nuclear power, in addition to layout planning, these technologies may also significantly help in operator training, and training of personnel in maintenance, dose reduction and repair departments. They may also offer better design optimization to more efficiently address regulatory issues such as plant safety and fire control.

While the field of 3D modeling has significantly matured, the next frontier is the full exploitation of *virtual reality* environments [1]. Efforts are already underway in conventional fields of engineering in this direction. For example, a project is sponsored by the Deere Co at Iowa State University to develop and evaluate a virtual environment for use in training, assembly, and maintenance methods development [2]. Such design-time optimization can also significantly benefit GEN-IV reactors.

At the start of this project there were only few research and development efforts in the nuclear arena in these areas. Virtual Reality (VR) systems were developed and applied at Argonne National Laboratory (ANL) to address issues from several different branches of science and engineering, including nuclear [3]. A detailed 3D model of an AP-600 reactor was developed. [Other applications of this technology at ANL include visualization of CAD data, crashworthiness experiments and computational combustion.] Work was carried out in France on developing computer-aided motion planning for nuclear maintenance. Researchers modified off-the-shelf 3D CAD modeling tools to suit the needs of nuclear industry. Specifically, fully functional CAD and computer aided motion (CAM) capabilities are now considered essential over the life of the NPP to "evaluate the feasibility of new maintenance tasks." It also is considered a useful tool for "communication and negotiation between all parties involved in a maintenance operation" [4]. Japanese and S. Korean researchers also worked along similar lines. Korea Electric Power Research Institute devoted significant resources to develop a virtual control room [5]. The fact remains that only relatively few groups in the nuclear field were taking advantage of these developments. In the US, beside the work at UIUC, Penn State University also has a program on virtual reality.

This project started with exploring different technologies suitable for nuclear applications. These are discussed below. Development work carried out under this project with selected technologies is discussed in the subsequent sections.

II. TECHNOLOGIES AVAILABLE

Several technologies can be utilized in the development of an effective virtual environment framework to optimize GEN-IV reactor designs. Some of these are discussed below.

II.A. GRAPHICAL USER INTERFACE

GUIs or Graphical user interfaces (at the 2D level) are now commonplace in nuclear software. However, there are still many applications that rely on old-style input/output interfaces. 2D GUIs were selectively used to simulate the perception of (2D) monitors and other 2D displays commonly used in the control rooms.

II.B. CAVE (CAVE AUTOMATIC VIRTUAL ENVIRONMENT)

The CAVE (so named after the “Allegory of the CAVE” in Plato’s Republic) was first developed at the Electronic Visualization Laboratory (EVL) at the University of Illinois in Chicago. A CAVE is operational at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign for many years. The CAVE (CAVE Automatic Virtual Environment) is a projection-based VR system that surrounds the viewer with 4 screens. The screens are arranged in a cube made up of three rear-projection screens for walls and a down-projection screen for the floor; that is, a projector overhead points to a mirror, which reflects the images onto the floor. Four basic components that comprise the CAVE are: the computers; the graphics systems; the tracking system; and the sound system [6].

The CAVE (see Fig. 1) works by reproducing many of the visual cues that brain uses to decipher the world around. Information such as the differing perspectives presented by the eyes, depth occlusion, and parallax (to name a few) are all combined into the single composite image that one is conscious of, while the rest is decoded by the brain to provide the depth cues. The CAVE must reproduce all of this information in real-time, as one moves about in the CAVE. The projected images are controlled by an SGI Onyx with two Infinite Reality graphics pipelines, each split into two channels. Sixty different images are displayed every second on each of the CAVE's four walls at very high resolution.

The CAVE's graphics system is broken into two parts: the images projected onto the CAVE's walls, and the shutter glasses that make the images appear three-dimensional (Fig. 2). The images projected onto the walls of the CAVE are provided by four Electrohome CRT projectors that surround the CAVE [6].

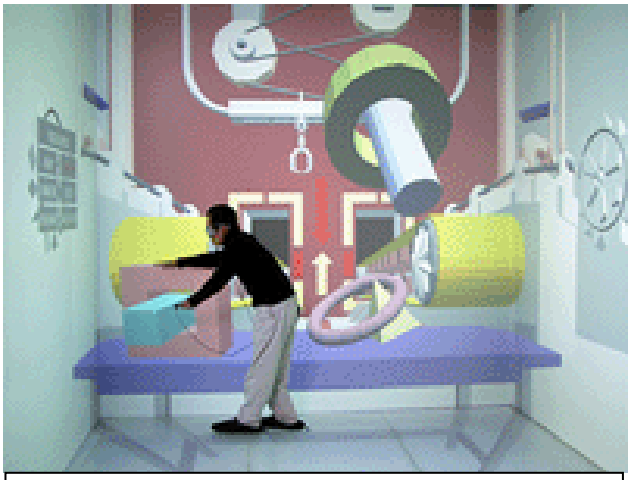


Fig. 1. A 3D virtual environment.

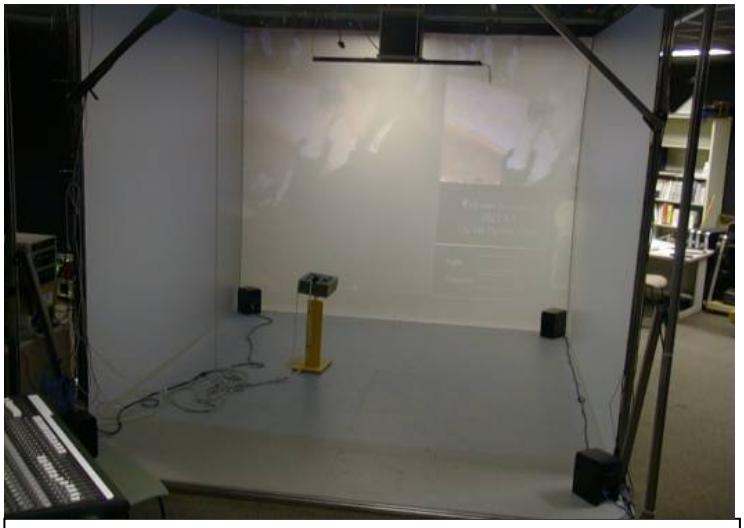


Fig. 2. The CAVE's visual and audio output hardware.

Two perspective-corrected images are drawn for each frame: one for the right eye and one for the left. Special Stereographics' CrystalEyes liquid crystal shutter glasses are worn that ensure that each eye sees only the image drawn for it. This creates a stereoscopic effect where the depth information encoded in the virtual scene is restored and conveyed to the eyes of those using it. The CAVE has an extremely advanced tracking system that enables it to constantly track the position and orientation of the special tracked glasses (Fig. 3). The six-degrees-of-freedom head-tracking device is mounted on the special glasses. The user stands inside the CAVE wearing these special (tracked) glasses. The person wearing the tracked glasses controls the viewpoint of the CAVE. They can look around the corner of an object, step behind it, look underneath it, or anything else that they could do in real life. A wand (a 3D mouse; also called a *CAVE-wand*) with buttons is the interactive input device (Fig. 4). The CAVE Wand is also attached to the Tracker Control Unit via a wire and allows the user to walk around (with the joystick in the middle) or interact with the virtual world through the push buttons. Currently, there are two types of wands. Both use the Ascension Flock of Birds tracking system [1], but have different control devices. The primary (new) wand has three buttons and a pressure-sensitive joystick. It is connected to the CAVE through a PC that is attached to one of the Onyx's serial ports. A server program on the PC reads data from the buttons and joystick and passes them to the Onyx. The older wand just has three buttons, and is attached to the mouse port of the Onyx. There are also "simulated" tracking options available, using either the keyboard and mouse or a spaceball. The use of one or another is transparent to the CAVE programmer, since it is defined

in the CAVE configuration file. Systems typically have two sensors, one for tracking the user's head, and another for the wand [6].

In addition to its extraordinary graphics capabilities, the CAVE also provides superb audio facilities. It has an eight channel audio system with state-of-the-art digital audio support that is controlled by an advanced sound mixing board [6].

For testing, one can run the CAVE using any number of walls simultaneously. The number of CAVE walls used does not affect the program. The CAVE library determines which walls are to be used and does the necessary setup when the program starts.

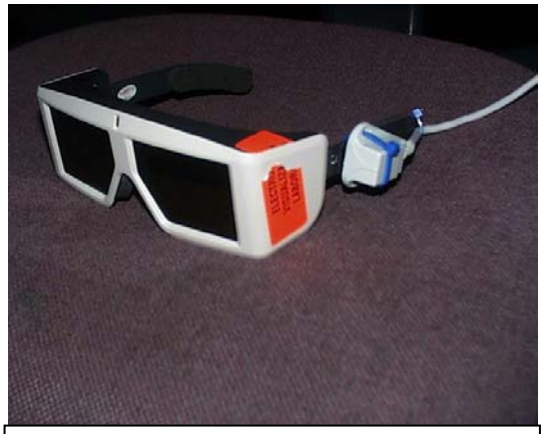


Fig. 3. These glasses enhance the user's perception of a 3D environment

Fig. 4: The CAVE's input device, CAVE Wand, is an advanced 3D mouse/joystick with more degrees of freedom than a conventional device for a similar purpose.

II.C. IMMERSADESK

The ImmersaDesk is a drafting-table format virtual prototyping device. The *original* ImmersaDesk had a 4 x 5-foot rear-projected screen tilted at an angle of 45 degrees. The size and position of the screen gives a wide-angle view and the ability to look down as well as forward. Except for the display device the same hardware is used as in the CAVE. User wears CrystalEyes stereo glasses. Stereo emitters are placed behind the screen. ImmersaDesk can use the same tracking systems that the CAVE uses. New models of ImmersaDesk are increasingly using flat screen monitors instead of projector-based screens [7].

II.D. VISBOX

Developed by some members of the CAVE user group at the NCSA, VISBOX (Fig. 5) is a poor-man's CAVE. It is a single wall system at a fraction of the cost of a CAVE. However, some of the features, such as the head tracking system, are superior to those available in the more expensive CAVE. User in the VISBOX cannot turn her face by 90 degree and still find immersed in the same environment, as is the case in the CAVE. However, applications are fully compatible with CAVE. Hence, development can be carried out on VISBOX, and then fine-tuned in the fully immersive environment of the CAVE. VISBOX runs on a single LINUX machine. However, Visbox is superior to CAVE in two ways. First, it uses a modified game controller commonly used for kids computer games. Operating using this game controller is significantly easier than the CAVE's wand. (Note that the game controller is modified with a tracker and hence it can be used to "point".) Second advantage of Visbox over CAVE is that the tracked glasses are wireless. The direction and orientation of the user is tracked by infra-red signals and two receptors mounted on the glasses. The glasses are also shown in Fig. 5. A VISBOX was acquired by the Department of Nuclear, Plasma and Radiological Engineering at UIUC with funds provided under the INIE grant (from DOE). This VISBOX is currently operational.

Fig. 5: The VISBOX is very similar to CAVE, but only has one display surface. Tracked glasses are wireless and standard game controller can be used instead of specialized wand.

II.E. STREAMLINING THE DEVELOP-INTEGRATE-TEST EFFORTS

Even though several different choices for a graphics library are available, OpenGL [8], seems to be the most widely supported one across different computer platforms and graphics cards, in addition to being one of the most advanced sets of routines for rendering graphics. OpenGL provides ample control of detail, and the ability to re-use code. In terms of the language for the actual computer code, or the source code, the most widely used languages for OpenGL are C and C++, both of which have been standardized and remain very popular. Since C++ is a superset of C, and its object-oriented structure can be very useful in implementing the hierarchical nature of the program, this is an ideal choice for the language.

However, it is realized that not all development will start from scratch. Hence, from the development of 3D CAD models to displaying them in the virtual environment, we also took advantage of other languages that are supported by CAD packages. Virtual Reality Modeling Language (VRML) [9] provided a common ground. CAD packages (at least some of them) do allow the users to save their models in *vrml* format, and given the toolbox mentioned above, these models can then be easily ported to the CAVE.

Fortunately, different visualization tools discussed above are somewhat "compatible." Development carried out on one can be ported to others as necessary. However, there was a need to streamline the transfer of 3D CAD models that can be developed in software packages like AutoCad, Pro-Engineer etc., into the 3D virtual environment.

III. TASKS ACCOMPLISHED

This being a very broad project involving a field that is still evolving and this being one of the first such applications in the field of nuclear engineering, with goals that ranged from exploration, to tool development, to model development, to testing for appropriateness; work carried out in this project presented below is grouped in the following categories.

- Model Development
  From Scratch (OpenGL library)
  Using packages such as AutoCAD, ProE, CREATE, etc
  Using game engines
- Interactivity
- Data display in virtual environment
- Visualizing radiation and ALARA training
- Design optimization
- Education and Training

III.A. MODEL DEVELOPMENT

There are several options available for model development for the purpose of rendition in a virtual environment. For example, a model of a nuclear reactor can be created by a proprietary drafting program *X* which runs on Windows machines. Here, the drafting program *X* might not

have an option to display models in a way compatible with 3D display technology. Moreover, being a product of a Windows-based application, the model could not run in the CAVE environment without first being converted to an appropriate file format. As another example, gaming engine *Y* might have superior capabilities permitting drafting, as well as the ability to add interactivity to the models created. Content developed using *Y* might even be compatible with 3D displays. However, the gaming engine *Y* might be platform dependent and only work on, say, Linux. These and several options have been explored. These are discussed below. Details are provided in appendices.

In addition, a conversion routine was written that allows *vrml* files to be displayed in the CAVE. Models created in CAD software can be saved in the *vrml* format, and then easily displayed in a virtual environment after passing through the above described code.

From Scratch (OpenGL based approach)

The approach chosen for this part of the work has been to develop virtual interactive models from scratch. The first decision to be made in this "from scratch" approach is the choice of programming language. C++ was selected, for being broadly supported and preferred among graphics programmers, in addition to being supported across many different platforms. The second decision was the selection of the graphics library – specifically, the set of APIs (Application Programming Interface) responsible for drawing objects on a computer screen. Here, the ideal candidate had to support 3D object rendering, and to be cross-platform compatible. This narrowed down the choice to OpenGL. OpenGL is a set of special purpose functions (and instructions) to draw objects (lines, polygons, etc) common to graphics display applications.

To develop an application executable on various platforms, the code developed was split into two parts. One part is strictly computer graphics, independent of computer platform. The other part contains all of the platform-dependent components of the code. For example, the platform-dependent portion of the code, when executed on a Windows machine, interacts with the Windows operating system and with DirectX (a set of programming instructions developed for the Windows platform), to obtain keyboard, mouse, and joystick data, and then passes this information along to the graphics portion of the code. In the CAVE, the platform-dependent portion of the code interacts with the CAVE library to obtain wand and keyboard data. It then passes this information along to the graphics portion of the code. The data obtained from platform-dependent components of the code are used to calculate user position, orientation, movement, object selection, etc. Calls to Windows API routines generate the window. They generate the frame, the minimize/maximize/close buttons, the title bar, the menu, and all the captions. Calls to Windows API routines also define and modify behaviors of the aforementioned items. They collect mouse/keyboard data as well as joystick data (specifically by using the Win32 API and DirectX API, respectively). Lastly, they define the large black area pictured. This area is set up to display objects drawn by OpenGL functions. This area is hence controlled exclusively by the OpenGL functions.

In the CAVE environment it is still necessary to communicate with the operating system that an area (full-screen sized in this case) needs to be created and controlled by OpenGL functions. This

communication with the operating system is done using the *CAVE library*. It is interesting to note that if desired, the same full-screen display could also be generated on a Windows machine, by using the appropriate Windows API functions.

Once various user-input data are gathered, such as which key has been pressed, or which way a user wants to move, it is then passed from platform-dependent routines to the OpenGL portion of the code. As can be seen, it is advantageous to have as much of the code as possible to be OpenGL based, and to minimize the platform-dependent portions, because these portions will need to be re-written for the various platforms available. Fortunately, generating an OpenGL display area and collecting user input is such a universal task that porting the platform-dependent portion of the code is almost trivial. The largest challenge of platform-dependent programming is to define the various menus and dialog boxes, which are discussed in Appendix A.

Using Packages such as AutoCAD, ProE and CREATE

While OpenGL based approach gives the greatest flexibility, it has the disadvantage of being labor intensive. Therefore, commercial modeling packages such as AutoCAD, ProEngineer and CREATE were also explored for virtual model development. Some basic models were developed using ProE but that effort was soon abandoned in favor of CREATE; a special purpose package developed at Halden Reactor Project. The model of the control room originally developed using CREATE (at Halden) and modified at UIUC is described in Appendix B and Appendix C.

Using Game Engine Technology

The most efficient approach to develop 3D models was found to be using the game engines. There are several available with their own pros and cons. Two were tried and finally the Unreal Tournament was selected due to its superior features. The details are given in Appendix D.

III.B. INTERACTIVITY

Interactivity and dynamic interactions with or within the virtual models is an important part of this project. The interactivity level and mode changes depending upon the platform and model development paradigm. For example, for models developed from scratch (OpenGL paradigm), all interactive features were developed in-house. Whereas for other modes of model development, numerous interactive features are built-in and have been fruitfully exploited.

Interactivity in OpenGL models

Interactivity is tied in to the operating system of a host computer. For example, on a Windows machine, the motion of a mouse device is interpreted by using Win32 API. Here, Windows provides the 2D location of the mouse device on a screen, and then OpenGL can be used to translate this to 3D coordinates in application space. In a CAVE, the CAVE library is used to read the 3D position of the wand device, which is translated to application-coordinates by using the CAVE library again. This data can be used for collision detection, navigation, or determination of the user's view.

Collision detection is based on defining an arbitrary, small volume around the user-selected coordinate. This volume is usually a cube, for simplicity. It is tested with every object in the scene for overlapping volume in space by calling the appropriate OpenGL functions. If there is an intersection between the arbitrary volume, and some enumerated object, this is defined as a collision event. If no objects intersect this arbitrary volume, then there are no collision events. A collision between a user and an object such as a wall can be programmed to prevent any movement in the forward direction; this prevents a user from "walking through walls and other objects." A collision between a user's hand and another object is defined as object selection. Based on the selected object's properties, the programmer defines whether the object is to be pushed, picked up, carried, moved, dropped, pressed, and so on.

The process of navigation is described below in detail.

Four variables – *position*, *speed*, *direction*, and *stepsize* – are used to navigate a scene, as follows. Navigation is accomplished either by pressing the g, h, n, b, f, and j keys to move respectively in the forward, backward, right, left, up and down directions, or by using a joystick or a wand in the CAVE.

In Windows, a user presses the 'g' key to move forward. *MainWndProc* function determines that a key has been pressed, and passes this information on to the *myApp::keyboard* function. Here, this keystroke is interpreted to have no meaning, and it is passed on to the *K_Render_Main::keyboard* function. At this point, the keystroke is interpreted, and the *K_glPerspective::MoveForward* function is called to adjust *position* by *stepsize* in the direction the user is facing, *direction*.

Instead of movement using the 'g' key, suppose that the joystick is used to navigate and it is moved forward. After a call to *UpdateInputState*, the program learns that the *y* position of the joystick is not zero. Immediately, the new position of the joystick is dispatched to *myApp::joystick*, which sends this information to *K_Render_Main::joystick*. After recognizing that *y* is *significantly* different from zero, this function calculates the time elapsed since it was last executed. This time step is multiplied by *speed, direction*, and by the value of *y* as well. Lastly, the function calls *K_glPerspective::MoveForward* to update the user's position. A few details on this algorithm follow. First, the value of *y* ranges from -1 to 1, with 0 meaning that the joystick is in its center position and not being moved. Since the values of *y* are real numbers, there is an infinite number of them (actually, in this implementation there are only 2001 discrete values, but for all practical purposes they are treated as being continuous in the real domain from -1 to 1). This means that the probability of the variable *y* being identically zero is, zero. Therefore, a range of real numbers around zero is defined to signify that the joystick is effectively not being moved. It is also important to note that the variable *speed* in the case of joystick actually defines the *maximum* speed of travel, i.e. when *y* = 1. Therefore, to obtain the user-selected speed, *speed* and *y* must be multiplied together.
As a final example, suppose a CAVE user points the wand in a certain direction, and pushes the joystick forward. Now, the variable *CAVE_JOYSTICK_Y* is evaluated to be non-zero indicating a desire to move forward. This leads to the evaluation of the time interval since the last function execution, and this time step is multiplied by *speed* and the value of *CAVE_JOYSTICK_Y*. A

function call, *CAVEGetVector (CAVE_WAND_FRONT, wandDir)* obtains the direction in which the wand was pointed. Multiplying this with the previously calculated scalar, and passing the result to *CAVENavTranslate*, automatically updates the position of the user in the virtual environment. Uses of these features to display radiation or dose etc and for other practical uses are discussed in the paper at the end of Appendix A.

Interactivity: Modeling Packages

Because of the nature of these packages, primarily developed for static display of models and features, it is relatively inefficient to introduce significant levels of dynamic interactive features in these models. Features that are built-in in the CAVE library are of course available for these models as well. The CREATE software, specifically built for interactive applications is however one exception, and does allow relatively easy manipulation of scenes and rendering in the virtual environment. For example, as discussed in the section on use of VR for ALARA, it was relatively easy to display visual radiation levels at various cut-planes in a model created in CREATE.

Interactivity: Game Engines

Model developed in game engines have several interactive features that have also been fruitfully exploited. These include automobiles, visual radiation levels, health meter adapted as a Geiger counter or a dose display meter, etc. Details are described in Appendix D.

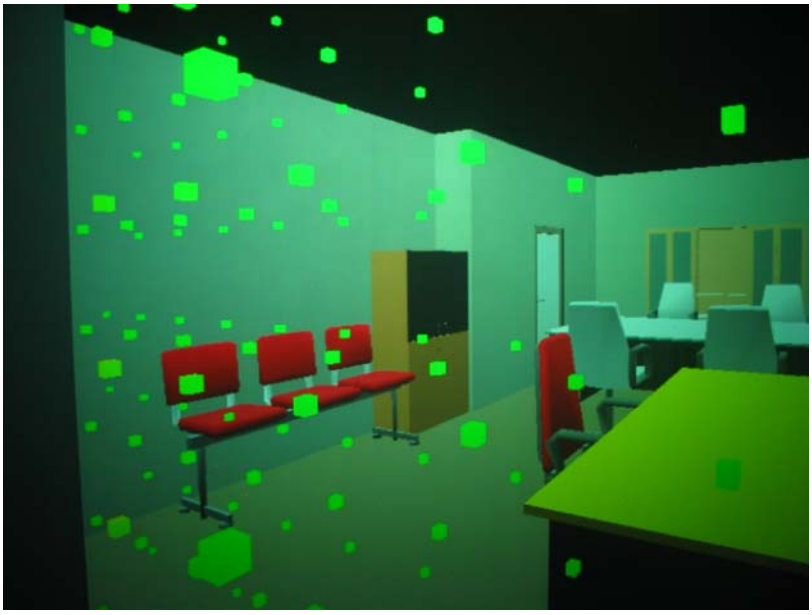
III.C. DATA DISPLAY IN VIRTUAL ENVIRONMENT

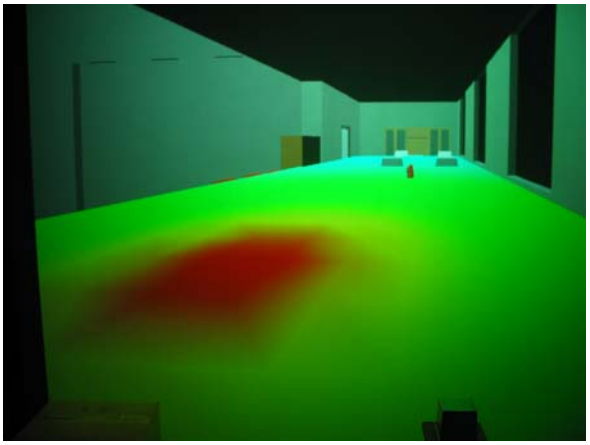
Display of neutron flux, temperature, pressure etc in virtual environment required the development and exploitation of standards developed at NCSA. Among other options, we have relied on Hierarchical Data Format (HDF) project of NCSA [10]. It involves the development and support of software and file formats for scientific data management. Codes have been written that allow a user to display numerical data from 3D simulations in the CAVE. A generic capability has been developed that allows data written in a class of HDF-4 standard files to be easily displayed in the CAVE. It is possible to rotate or translate, and show data on cutting planes.

III.D. VISUALIZING RADIATION AND ALARA TRAINING

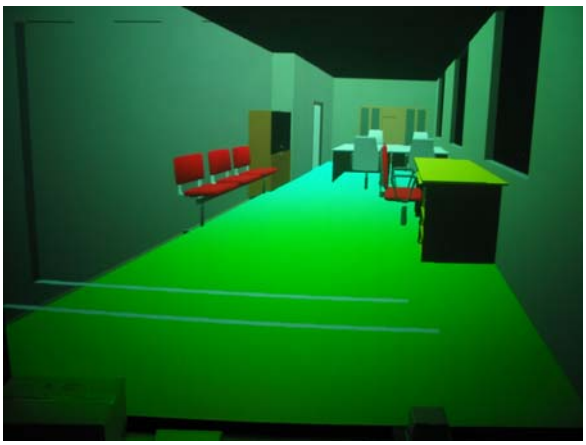
Display of radiation levels in a 3D virtual model was accomplished though different techniques depending upon the model being used. For example, in the case of the model developed in CREATE or one of the 3D modeling packages, a script was written to display radiation levels in the coordinate system of the model. A text file with radiation levels and corresponding coordinates is read and the data is displayed in either color contours or other formats. For the applications developed from scratch for the CAVE, the script also “integrated” the local radiation level and the amount of time spent at that site to calculate a running dose, which was displayed on a meter. Similar approach can also be easily developed for the models developed using the game engine.

Pictures of “visualization” of radiation field data are shown below. Different ways of displaying radiation level has been tested. In one approach, color and size coded glyphs can be displayed indicating radiation levels in the 3D virtual environment. In another approach, the user can see color-coded planes at different elevations in the virtual environment. These planes can be selected by simply pressing a button on the controller. Models shown here are those of the virtual reality laboratory in NPRE at UIUC. Red indicates high radiation levels. It should be, at the risk of repetition, pointed out that these are not just 3D models with radiation field displayed. The real strength of VR is that the user can immerse herself and “walk” around in these simulated radiation fields—an experience that cannot be conveyed on paper.





(radiation plane at z = 3 ft)



(radiation plane at z = 4 inches)

III.E. DESIGN OPTIMIZATION

Numerous tools developed under this project are designed with the goal to be used for design optimization from operational point of view. Due to limited progress at the national level on specific details of a next generation nuclear power plant, only generic studies were carried out to test these tools for design optimization. These included the use of the virtual control room model for analog to digital conversion studies. As a result of the workshop conducted at UIUC on the use of VR in nuclear engineering applications, a specific case study was conducted with a utility. This utility is studying the conversion of its control room from analog to digital. A virtual

model of the existing (analog) control room was developed. Next, part of the room was replaced by a digital set up. Personnel from the utility visited UIUC to explore the utility and functionality of the "mixed" control room. [Note that it is expected that the transition from analog to digital control room will go through a similar step-by-step conversion in which for some time both systems will have to work simultaneously.] Exercise was very fruitful and established the utility of virtual models for such studies. The model was studied for such features as line-of-sight clearance, reach, visibility and readability, mobility, etc.

III.F. EDUCATION, TRAINING AND OUTREACH

Though not proposed in the original proposal, one of the beneficial by-product of this project was the demonstration that these tools are extremely useful for education, training and outreach. Virtual models were a huge draw for high school students and their parents. In addition, we participated in an outreach exercise at a major museum in Chicago, which drew several thousand visitors.

IV. INTERNATIONAL AND INDUSTRY COLLABORATION

We collaborated with Halden Reactor Project (Norway) on development, display and evaluation of nuclear specific VR applications. Specifically, UIUC acquired Halden's CREATE software. This was used to quickly and easily develop some of the applications.

KAERI has also been active in this area. UIUC hosted Dr. Kyung-Doo Kim of KAERI who worked on integrating virtual reality with some of the integral system codes like RELAP. This was a very complex problem and required a good understanding of both nuclear system codes as well as virtual reality.

UIUC also forged close relationship with EPRI and utilities. A UIUC-EPRI sponsored international workshop on VR in nuclear engineering was held on the Urbana campus in April of 2005. Several participants from nuclear utilities also attended the workshop and evaluated the use of VR for nuclear specific applications.

REFERENCES

1. W. Sherman and A. Craig, Understanding Virtual Reality, Morgan and Kaufmann, 2002.

2. Argonne National Laboratory web site: http://www.re.anl.gov:80/vr/virtualreality.html

3. T.S. Shaw, "Generation IV Nuclear Energy System Construction Cost Reductions Through the Use of Virtual Environments," NERI Project, 2001.

4. E. Schmitzberger and J.L. Bouchet, A Test Case of Computer Aided Motion Planning for Nuclear Maintenance Operation," ICONE, 2001.

5. The Avant-Garde Simulation Technologies in Korea Electric Power Research Institute, Simulator Team, http://www.kepri.re.kr.

6. NCSA web site: www.archive.ncsa.uiuc.edu/VR/VR and http://cave.ncsa.uiuc.edu/

7. Electronic Visualization Laboratory, UIC, website: http://www.evl.uic.edu/home.html

8. Iowa State University's Virtual Reality Application Center's website http://www.vrac.iastate.edu/

9. NCSA's HDF group: http://hdf.ncsa.uiuc.edu/

10. J. Winters, "Standards and Guidelines for Cost Effective Layout and Modularization of Nuclear Reactor Plants," NERI Project, 1999.

APPENDIX A

Details of OpenGL-Based Approach for 3D, VR Models

A program was developed to develop 3D models from scratch which will allow interactivity in the context of nuclear specific applications. The program provides a natural or instinctive way of changing various parameters; for example, moving the control rods by moving a lever. The basic method to achieve this goal includes creating a simulated control room, with customizable panels to hold all the instrumentation, and panel holders to hold the panels.

The program is divided into three major sections: the dynamically linked library (*dll*), the main program (*exe*), and the configuration data (*txt*). The purpose of the *dll* is to enumerate and evaluate a given function. Individual functions can be added to it for the purpose of reading a file, getting a value from a different program, or calculating a value from any given mathematical function. The *exe* draws what is actually displayed on the screen. It contains the functions to draw such items as a gauge, a panel with gauges, a structure containing panels, or an entire control room containing customizable components. Each of the components being drawn is defined in the *txt* by such parameters as dimensions, and the index of the function in the *dll* that should be used to evaluate, for example, the position of the meter needle. Also, each component that can be drawn by the *exe* includes one or more default pre-defined values for testing purposes.

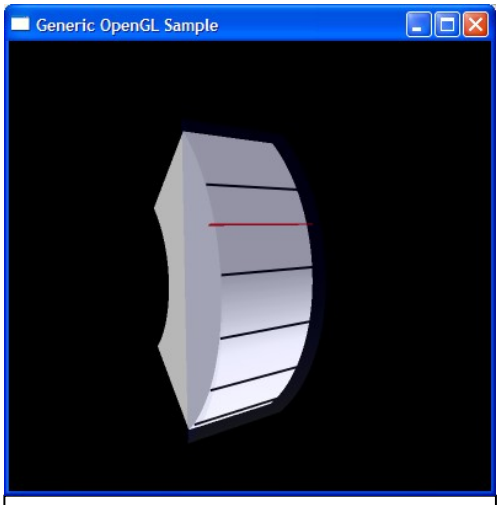


Fig. A-1: A virtual cylindrical gauge

The following is a more detailed description of the program. Design of a basic component to interact with the *dll*—in this case, a cylindrical gauge—is described first (Fig. A-1). The gauge includes pre-defined values for such customizable items as the radius to the glass, or the needle, or the face. However, as of now the particular values of those parameters being used have been hard-coded, and for convenience defined as set A. The user would specify in the *txt* file that a cylindrical gauge A is needed at a certain position, with a given length, width, face, and index of the function to be used for evaluating the needle's position.

The gauge is a sample component; others such as a knob, lever, CRT, etc. are being developed.

Next, a panel is created that holds different components. To test the link between the panel and individual components, several default sets of parameters were defined to respectively display 1, 2, 4, or 8 of the aforementioned gauges. A picture of the panel is shown in Fig. A-2.

Naturally, the next component to be built is a place to install the panel. This could be a table (control panel) or a wall. These are created using the general category of "cylinder" (two parallel plane surfaces and walls connecting them). Pictures of a 3D control panel and a segment of a wall

holding various instrumentations are shown in Figs. A-3 and A-4, respectively. An actual control room with instrumentation panels that can be arranged by the user, is shown in Fig. A-5.

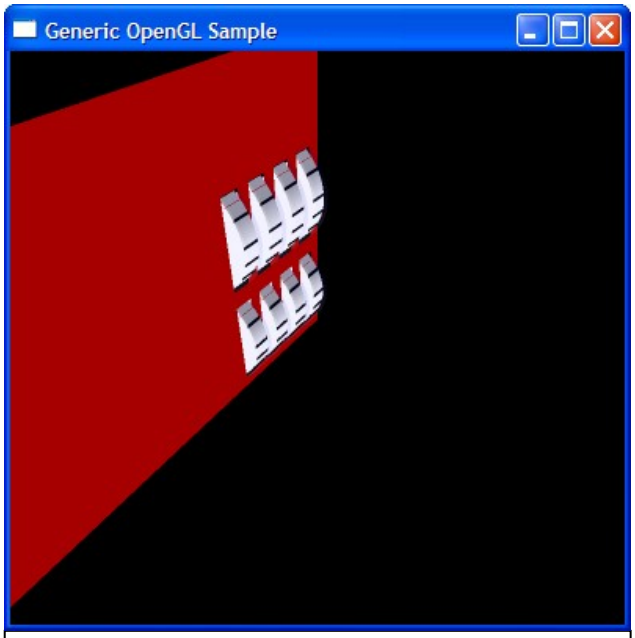


Fig. A-2: The panel used to hold various instrumentation components being tested with 8 gauges.

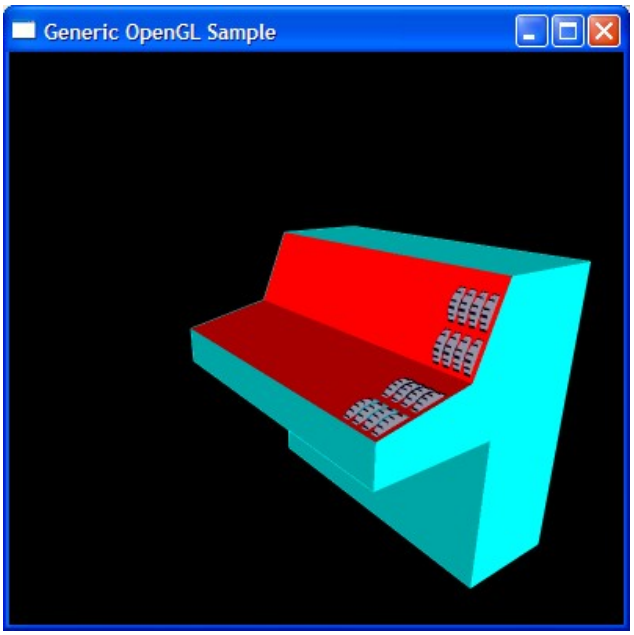


Fig. A-3: A sample control panel with instrumentation.

The universal potential of the simple design of a cylinder with any user-defined polygon as the base is exploited. For example, it is used to create a more universal code, and to draw a room—the ceiling and the floor are the base of the cylinder, and the sides of the cylinder are the walls of a room. The code, previously used to draw a single panel, is extended to draw any given side of a cylinder, and the routine to draw the control panel, as seen in Figs. A-3 and A-4, is extended to a more general code to draw any given cylinder.

The complexity of the program necessitates that the parameters used to draw the graphics (number of gauges, dimensions, etc.) be separated from the actual rendering code. The next major step was to create a platform-independent implementation of a user-friendly interface to change those parameters. This step helps the development and implementation of the *txt*. The ability to use pre-selected images to emulate material textures, and to provide more control over what is displayed on gauges, has been incorporated.

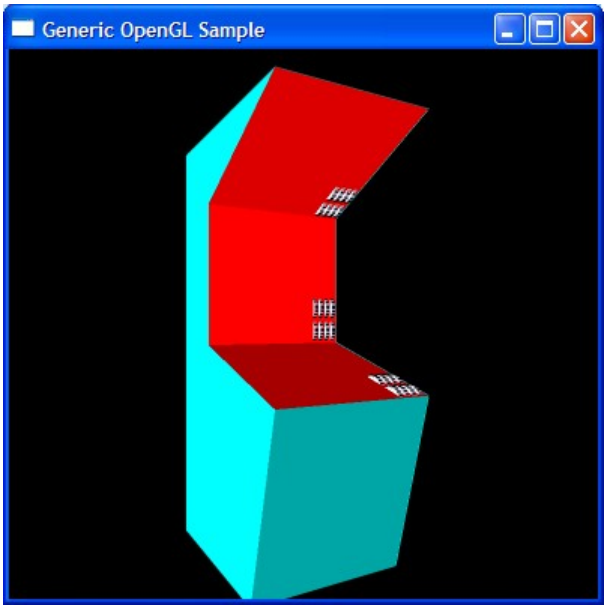


Fig. A-4: Another sample control panel with instrumentation.

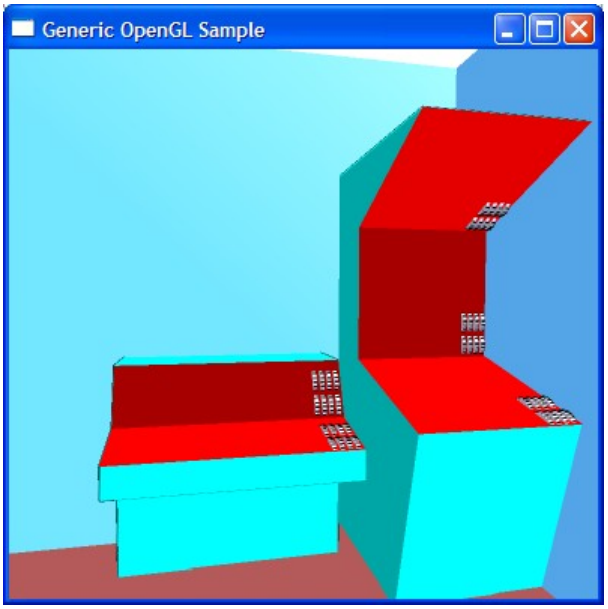


Fig. A-5: A sample control room in early stages of development.

Additional details are given in the following paper presented at the 2004 ANS International Topical Meeting on Nuclear Power Plant Instrumentation, Controls and Human-Machine Interface Technologies.

Fourth American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Controls and Human-Machine Interface Technologies (NPIC&HMIT 2004), Columbus, Ohio, September, 2004

THE VIRTUAL NUCLEAR LABORATORY

Nick Karancevic and Rizwan-uddin
Department of Nuclear, Plasma and Radiological Engineering
University of Illinois, Urbana, IL 61801, USA
nick@karancevic.com rizwan@uiuc.edu

Keywords: virtual reality, simulations, visualization

ABSTRACT

In a world where people are overwhelmed by visual information, from television and billboard ads to computer video games, it should come as no surprise that industries are increasingly relying on advanced visualization technology to better handle the increased flow of information. Advanced visualization tools are also being used to improve human-machine interface by providing much more realistic simulated environments for design, training, planning and "practice" purposes. For nuclear engineers, technology to simulate everything from simple half-life measurement experiments to complete control rooms, is readily available and can be used on platforms as accessible as personal computers – or as sophisticated as three-dimensional virtual reality systems. Presented herein is what to an outside observer might look like a typical computer video game, yet to a nuclear engineer it would more closely resemble a simulated nuclear environment, such as a radiation lab, or a control room of a research reactor. It is hoped that modeling tools developed in this project will help large-scale exploitation of virtual reality technology by nuclear engineers.

1. INTRODUCTION

Virtual reality (VR) is an excellent tool for education, outreach, training, planning and research (Sherman 2003). Its use by nuclear scientists and engineers is also on the rise. Reviews of visualization technology and applications in the field of nuclear engineering have recently appeared (Rizwan-uddin, 2003; Karancevic 2003; Whisker 2003a and 2003b). Research groups in France, Korea, Japan and Scandinavian countries are actively pursuing the use of VR in the field of nuclear engineering. In these proceedings, Hanes and Naser (2004) review various visualization tools, including VR, and their use in medicine and different branches of engineering, including nuclear. For a more detailed review of visualization tools available as well as for recent applications in nuclear engineering, reader should refer to these references.

We are developing basic tools to facilitate the use of virtual reality in the field of nuclear engineering. Intended applications range from simple virtual tour of nuclear facilities for outreach purposes, conducting virtual radiation related experiments, virtual facilities for improved human-machine interfacing, virtual facilities for optimum design to minimize maintenance and replacement time for parts, virtual dose calculations, etc. We are also working on a virtual model of a next generation research reactor that will

become a test case for the tools being developed by our group. We here report recent progress made towards creation of virtual nuclear facilities including simple radiation experiments and a reactor control room. We have added enhanced capabilities to the basic computer code. These include standard graphics routines, hierarchical and organizational data structures, as well as an increased level of realism and more sophisticated models than reported earlier.

2. HARDWARE AND SOFTWARE

As with most other applications, it is quite likely that PCs will eventually be the computing platform of choice for this application as well. Display technology is likely to grow independently, most likely influenced by the flat panel technology, and eventually moving toward wider use of large size curved displays. Consumer-level graphics cards are capable of supporting dual monitor displays, which can be creatively exploited to display stereoscopic images on one monitor instead. 3D display monitors have already been developed, and are likely to become economically feasible over the next several years. Software to display 3D virtual systems will most likely be influenced by the video game industry. Hardware and software currently being used in this project are briefly discussed below.

2.1 Hardware

The CAVE (so named after the "Allegory of the CAVE" in Plato's Republic) was first developed at the Electronic Visualization Laboratory (EVL) at the University of Illinois in Chicago. A CAVE has now been operational at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign for many years. The CAVE (CAVE Automatic Virtual Environment) is a projection-based VR system that surrounds the viewer with 4 (or more) screens. The screens in NCSA CAVE are arranged in a cube made up of three rear-projection screens for walls, and a down-projection screen for the floor (a projector points to a mirror overhead, which reflects the images onto the floor). Four basic components that comprise the CAVE are: the computers; the graphics systems; the tracking system; and the sound system (NCSA CAVE website).

In addition to its extraordinary graphics capabilities, the CAVE also provides superb audio facilities. It has an eight channel audio system with state-of-the-art digital audio support that is controlled by an advanced sound mixing board. Details can be found at the CAVE website.

Fig. 1. A Picture of the UIUC CAVE.

Developed by some members of the CAVE user group at the NCSA, VISBOX (Fig. 2) is a poor-man's CAVE. It is a single wall system at a fraction of the cost of a CAVE. However, some of the features, such as the head tracking system, are superior to those available in the more expensive CAVE. VISBOX user cannot turn her face by 90 degrees and still find immersed in the same environment, as is the case in the CAVE. However, applications are fully compatible with CAVE. Hence, development can be carried out on VISBOX, and then fine-tuned in the fully immersive environment of the CAVE. VISBOX runs on a single LINUX machine. A VISBOX will soon be operational in the Department of Nuclear, Plasma and Radiological Engineering at the University of Illinois at Urbana-Champaign.

Fig. 2. A picture of a VISBOX and a user.

2.2 Software

Even though several different choices for a graphics library are available, OpenGL (OpenGL website), seems to be the most widely supported one across different computer platforms and graphics cards, in addition to being one of the most advanced sets of routines for rendering graphics. OpenGL provides ample control of detail, and the ability to re-use code. In terms of the language for the actual computer code, or the source code, the most widely used languages for OpenGL are C and C++, both of which have been standardized and very popular. Since C++ is a superset of C, and its object-oriented structure can be very useful in implementing the hierarchical nature of the program, this is an ideal choice. The simplicity and versatility of OpenGL with C and C++ allow for complete flexibility, great power, and the ability to easily incorporate any needs specific to nuclear simulations.

However, it is realized that not all developments will start from scratch. Hence, from the development of 3D CAD models to displaying them in the virtual environment, we are also taking advantage of other languages that are supported by CAD packages. Currently there are relatively few standards. Virtual Reality Modeling Language (VRML) seems to provide a good starting point (VRML website at web3d.org). Some CAD packages do allow the users to save their models in *vrml* format, which can then be easily ported to the CAVE (Rizwan-uddin 2003).

In addition to modeling and representation of the physical systems, standards must also be developed to display the results of numerical simulations (like system variables such as temperature, neutron flux, heat generation rate, etc.) as part of the 3D CAD model. In this regard, past work carried out at NCSA toward the development of standards for file formats will be very useful. Specifically, we plan to rely on Hierarchical Data Format (HDF) project of NCSA (HDF website at NCSA). HDF project provides a standard for the development and support of software and file formats for scientific data management.

3. THE CODE AND GUIs

A general-purpose program is being developed in C++/OpenGL to create virtual models of interest. The program is modular and allows development of components and their assembly. For example, models of meters, gauges, levers, etc., can be assembled on a control panel; several control panels and furniture can be assembled into a control room; and several rooms can be assembled to form a building. Models can be divided into four categories: static; dynamic; interactive; and interactive/simulators. Static models are the simplest with no moving parts. One can walk or fly through these static virtual models. Dynamic models have moving parts, with predetermined motion that cannot be altered by the viewer. Interactive models allow the viewer to move (some or all) components, such as a chair, by "grabbing" them. Interactive/simulator models are the most detailed, and allow the user to interact with the environment and then observe the result or consequence of her action. A simple example would be a worker in a radiation field with a Geiger counter. As the worker walks around the radiation field, the counter displays the exposure/dose, based on a pre-calculated radiation field or as determined in real time. A more complicated example is that of a control room in which the operator can press buttons and turn knobs, and the meters and dials then, based on pre-calculated responses or real time simulations, display the reactor response.

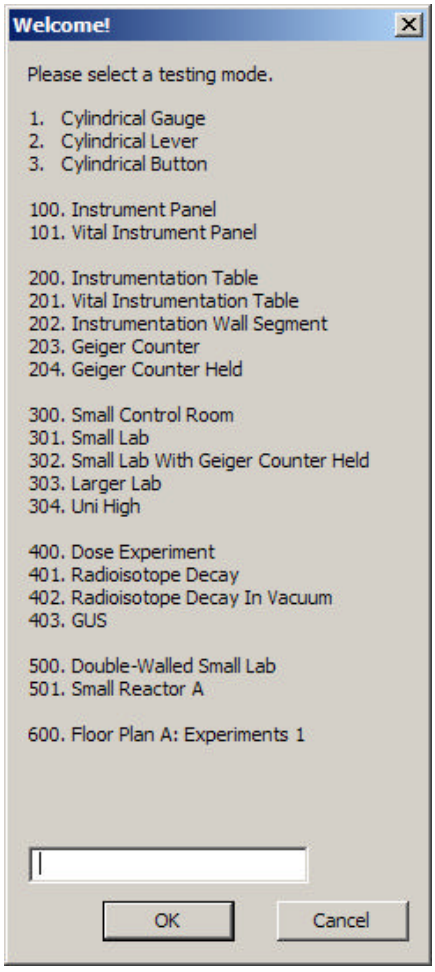


Fig. 3 Startup dialog box

The importance of hiding intricate details of programming is as crucial in the development of these virtual models as with any engineering process. An interactive capability, utilizing standard Windows dialog boxes, has thus been developed to shield

the last layer of model developers from the "messy" details. Three dialog boxes of this "developer's tool" are shown in Figs. 3-5. Figure 3 shows various models developed thus far, and allows the user to select one by entering the corresponding code. The dialog box shown in Fig. 4 allows for on-the-fly changes to any component of the simulation, as well as for saving and loading any particular model. The dialog box shown in Fig. 5 contains various options and debugging parameters, including the use of a 3D anaglyphic display.

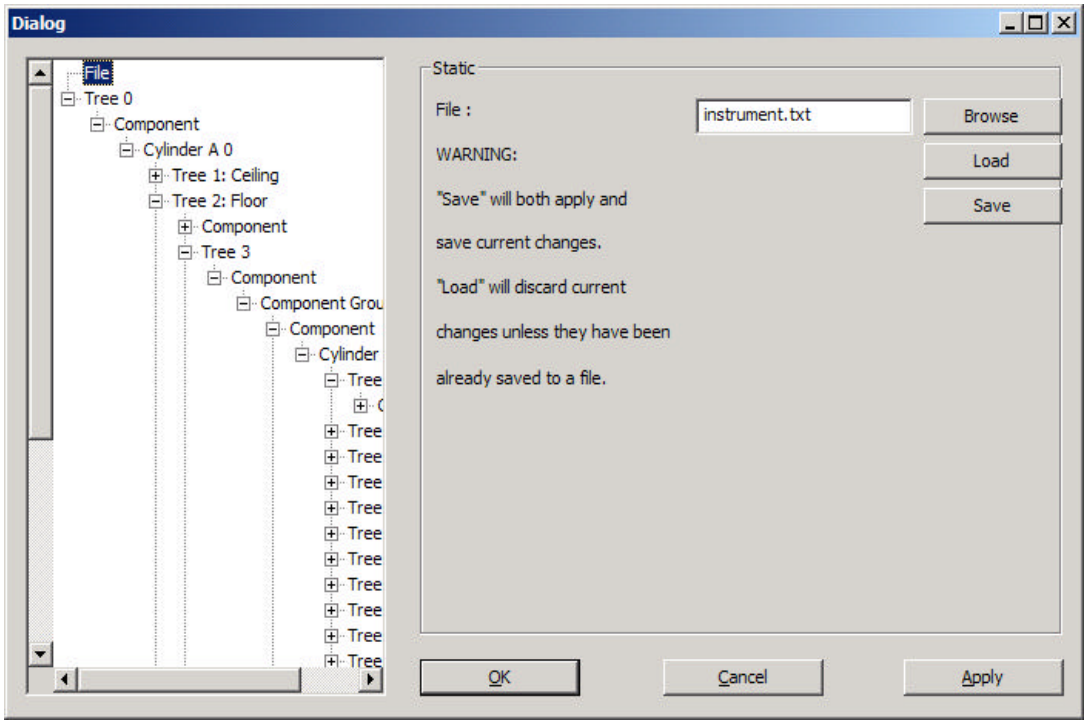


Fig. 4 VR model editor

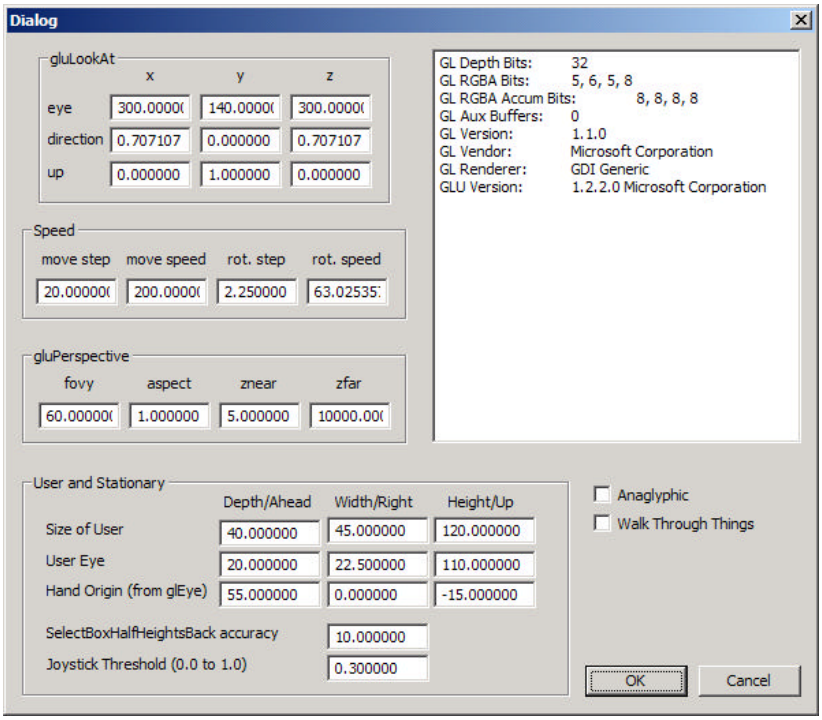


Fig. 5
Various options for model editing

Briefly, the data structure revolves around a tree structure. Each "tree node" contains a "root component" and an arbitrary number of branches. Each branch is no more than another tree node. The data structure allows great flexibility in modeling. Any item to be modeled can be the aforementioned "component." One particular component is named a "cylinder" because it is composed of a floor, a ceiling, and an arbitrary number of walls, each of which is represented by a tree node. As an example, the entire scene of Fig. 6 starts with a "cylinder." This "cylinder" contains four tree nodes with quadrilaterals and their properties (walls), a tree node containing a polygon for the ceiling, and a tree node for the floor. The tree node for the floor contains a polygon describing the floor surface, and several branches, each of which represents a component, or group of components placed on the floor. For example, one of the branches contains a "cylinder" representing the brick wall. The end user will however be able to *click* through the GUI, and see descriptive labels and options. Hence, these "messy" details will not be very important.

4. VIRTUAL MODELS DEVELOPED

Several components, such as gauges, meters, levers, and more detailed models have been developed. Four of these models are described here.

The first, and simplest, of these is a virtual experiment to measure the half-life of a radioactive substance. As the shield between the radioactive substance and counter is removed, the virtual detector displays the number of counts over short time intervals. These counts are automatically updated. The data displayed by the detector can be either pre-computed and stored in a database, or calculated in real-time (Fig. 7).

The second of these models is a virtual laboratory for radiation related experiments. It consists of a room, a shielding wall, and a workbench. It also includes some virtual instruments like a Geiger counter. The radiation field is due to a point source placed behind the wall shown in Fig. 6. The user, holding the virtual Geiger counter in her hand, can move around the room, either by using the joystick or a keyboard on a PC, or by physically walking around in the CAVE. The Geiger counter displays the radiation level (from a pre-calculated database) at the current location of the user. Model is currently being extended to calculate the total dose received by the worker from the time she enters the room till her exit (Whisker 2003b).

The third interactive simulation is a model of the GUS (graphite uranium sub-critical [assembly]) facility at the University of Illinois. Driven by a source, this sub-critical assembly is used for undergraduate laboratories involving measurements of flux distribution, buckling, neutron leakage, and so on. A virtual model of this facility provides a tour of the laboratory, and will have the capability to conduct simple virtual experiments in the future (Fig. 8). The GUS assembly is modeled by "pasting" digital pictures of GUS on a cube. Figure 9 shows a picture of the virtual radiation lab with some control panels, multi-channel analyzers, and the GUS assembly in the back. Certain components, like chairs, in this model can be "grabbed" and moved by the user.

Figure 6 Shielding experiment

Figure 7 Decay experiment

Figure 8 GUS Facility

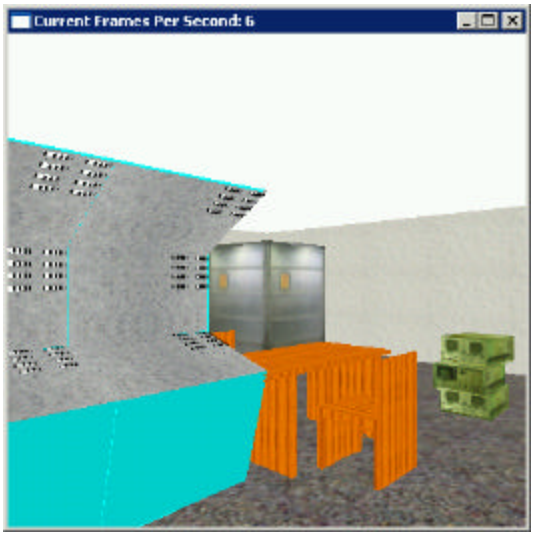


Figure 9 A laboratory, with control panels, a desk and some chairs, GUS, and some multi-channel analyzers

The last, and the most detailed, model is that of a research reactor (Figs. 10 and 11). Components in this model include desks and chairs, generic control panels, and a workbench, displaying reactor power (%) and bulk temperature (°C). Also modeled is the emergency "scram" switch. Once again, power and temperature, after an operator's virtual intervention, can be displayed on the virtual gauges, either from a pre-calculated database or from real time simulations.

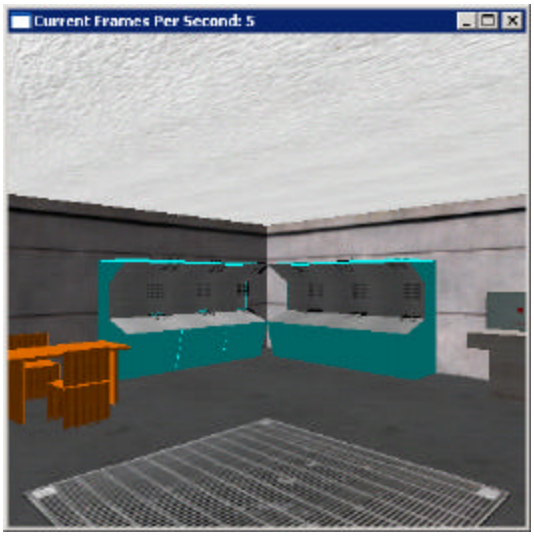


Fig. 10 Prototype research reactor

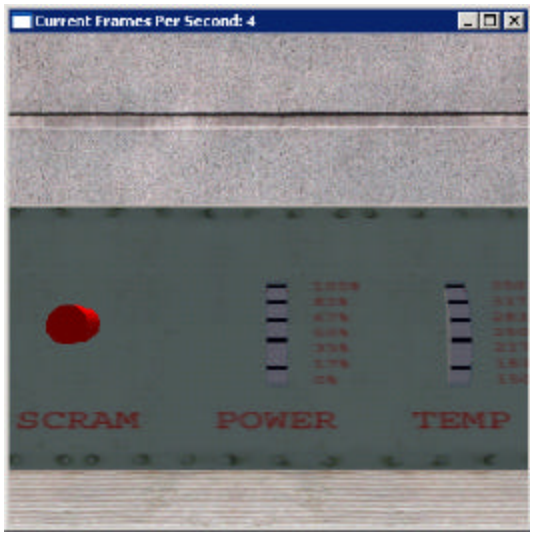


Fig. 11 Details of a control panel

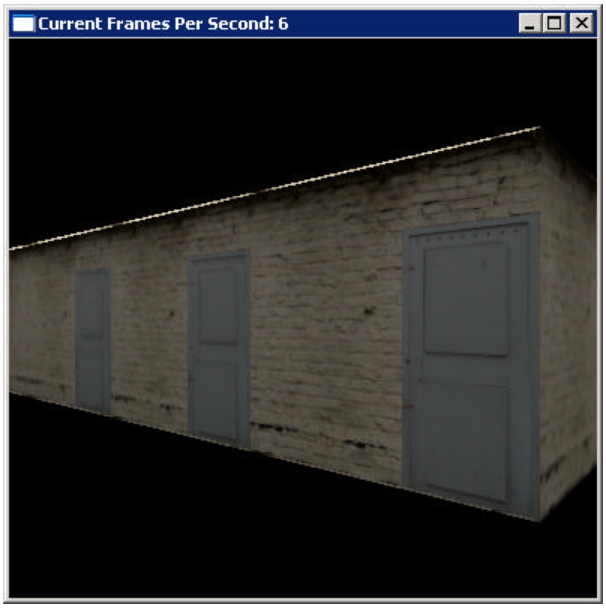


Fig. 12 Several models assembled in a building

Finally, to demonstrate the ability to assemble "components" into larger models, models 2-4 are assembled into a building with three rooms, each housing one model. One can walk through a door in Fig. 12 and find himself in either Fig. 6, 9 or 10. The potential of this approach is only limited by a designer's imagination. Tools developed here can be easily expanded to develop the model of a complete commercial power reactor site.

5. CONCLUSIONS

Safety and ability to minimize cost and exposure risk will inevitably remain two of the goals of any nuclear enterprise. Several recent technological innovations, including virtual reality, can be helpful in achieving these goals. Virtual reality cannot only be used to improve human-machine interfacing and operator training; it may also be very useful in achieving educational and outreach goals of the discipline. Nuclear engineers have only recently started to exploit the potential of virtual reality. While only a few simple models have been presented here, several more detailed and complex models, ranging from more sophisticated experiments, to parts of nuclear power sites, are currently being developed, and will be presented elsewhere.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the NCSA staff for their extensive support and opportunity to use the CAVE facility. Work was supported in part by a DOE NEER grant, number DE-FG07-03ID14501.

REFERENCES

Baratta, A.J., Whisker, V.E., Mouli, S.C., Shaulis, S.A. Shaw, T.S., Warren, M.E., Winters, J.W., and Clelland, J.A., 2002. Generation IV Construction cost Reductions Using Virtual Environments. *Trans. ANS*, **86**, p. 40.

CAVE website at NCSA, 2004. http://cave.ncsa.uiuc.edu/

Hanes, L.F. and Naser J., 2004. *Use of Visualization Technology to Improve Human Decision-Making*, in Proc. Fourth American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Controls and Human-Machine Interface Technologies (NPIC&HMIT 2004), Columbus, Ohio, September, 2004.

HDF website at NCSA, 2004. http://hdf.ncsa.uiuc.edu/

Karancevic, N., Rizwan-uddin, 2003. *Virtual Systems for Understanding and Advancement of Nuclear Power*, Trans. ANS/ENS International Winter *Meeting*, New Orleans, LA.

National Center for Supercomputing Applications, 2004. http://www.ncsa.uiuc.edu

OpenGL official website, 2004. http://www.opengl.org

Rizwan-uddin, Karancevic, N., Tikves, S., 2003. Virtual Reality: At the Service of GEN-IV, and V, and ...., Proceedings of *ICAPP-2003*, ANS, Cordoba, Spain.

Sherman, W.R., Craig, A.B., 2003. Understanding Virtual Reality, Elsevier Science, San Francisco, CA.

Visbox, inc., 2004. http://www.visbox.com

VRML website, 2004. http://www.web3d.org/vrml/vrml.htm.

Whisker, V.E., Warren, M.E., Baratta, A.J., Shaw, T.S., 2003a. Using Immersive Virtual Environments to Develop and Visualize Construction Schedules for Advanced Nuclear Power Plants, Proceedings of *ICAPP-2003*, ANS, Cordoba, Spain.

Whisker, V.E., Warren, M.E., Baratta, A.J., Shaw, T.S., 2003b. Development of a Radiation Dose Model for Immersive Virtual Environments, *Trans. ANS/ENS International Winter Meeting*, New Orleans, LA.

APPENDIX B

Visbox and Some Virtual Models

Screen shots of some of the virtual models (displayed on the large VISBOX screen) are shown here. The control room model was originally developed at Halden. It was modified at UIUC.

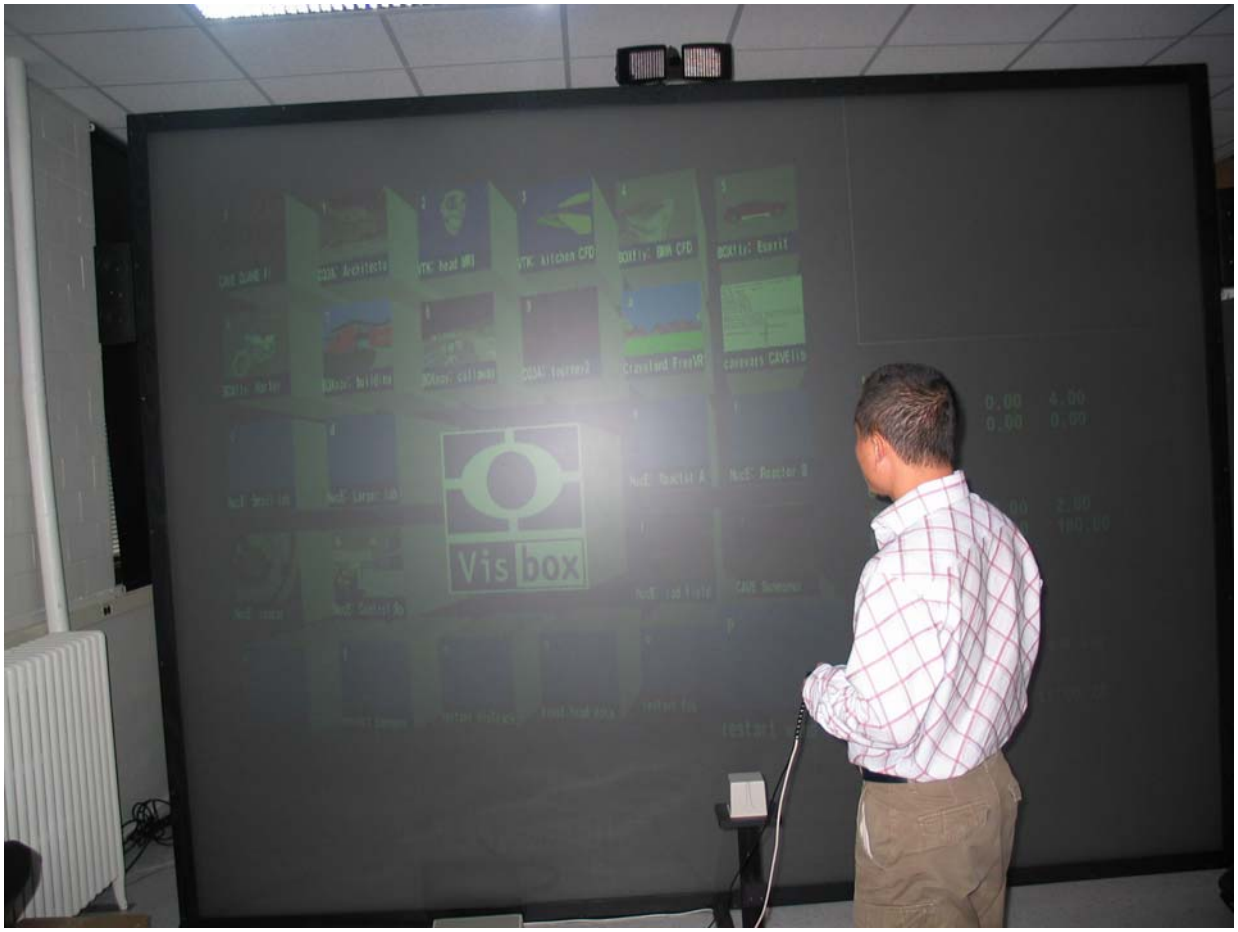


Figure B-1. A picture of the VisBox screen displaying the menu for several applications is shown. Any application can be selected using the game controller.

Figure B-2. A modern control room model being displayed—with an operator in the foreground (barely) shows that one can get a fairly realistic virtual experience in a VisBox.

While standard VR allows navigation in the virtual environment, introducing interactivity with objects such as control panels and touch screen monitors in such environment has been implemented. For this purpose, among other products we have used the front end of the BWR simulator and displayed it in the virtual environment of a reactor control room. Figure B-3 shows a virtual control room with the front end of the BWR simulator “pasted” on a control panel of the virtual model. [The red button near the center of the picture is the “scram” button.]

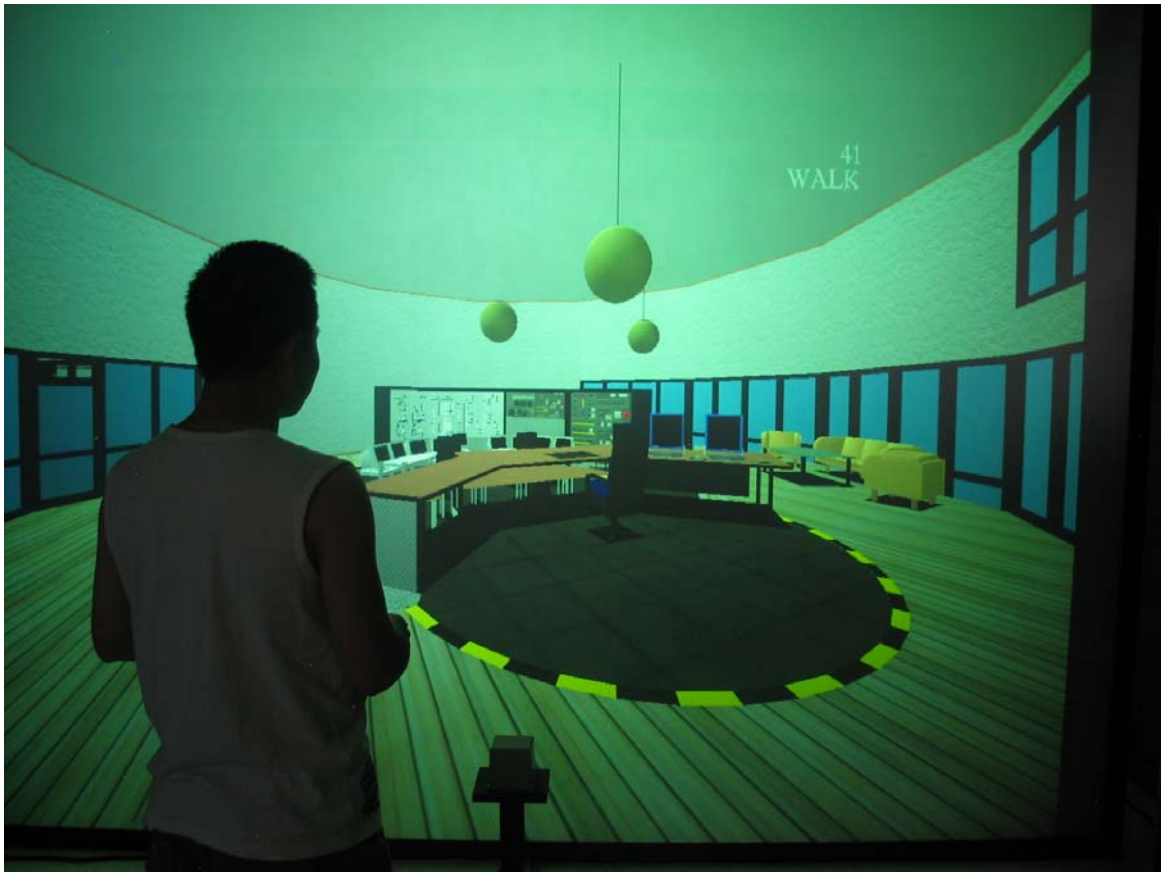


Figure B-3. A BWR simulator Figure B-3. A BWR simulator in a virtual control room.

Figure B-4. Another view of the virtual control room model

The control room optimization and virtual control room with a simulator have been carried out for a generic control room design. However, fundamental work is complete and ready to be implemented for any specific design that is chosen for the next generation reactor.

APPENDIX C

A Virtual Control Room with an Embedded, Interactive Nuclear Reactor Simulator

Stefano Markidis and Rizwan-uddin

Department of Nuclear, Plasma, and Radiological Engineering
University of Illinois, Urbana, IL 61801, USA
s.markidis@gmail.com, rizwan@uiuc.edu

Abstract – *The use of virtual nuclear control room can be an effective and powerful tool for training personnel working in the nuclear power plants. Operators could experience and simulate the functioning of the plant, even in critical situations, without being in a real power plant or running any risk. 3D models can be exported to Virtual Reality formats and then displayed in the Virtual Reality environment providing an immersive 3D experience. However, two major limitations of this approach are that 3D models exhibit static textures, and they are not fully interactive and therefore cannot be used effectively in training personnel. In this paper we first describe a possible solution for embedding the output of a computer application in a 3D virtual scene, coupling real-world applications and VR systems. The VR system reported here grabs the output of an application running on an X server; creates a texture with the output and then displays it on a screen or a wall in the virtual reality environment. We then propose a simple model for providing interaction between the user in the VR system and the running simulator. This approach is based on the use of internet-based application that can be commanded by a laptop or tablet-pc added to the virtual environment.*

I. INTRODUCTION

Virtual Reality (VR) systems may be used to help design control rooms and to train personnel in (virtual) nuclear power plants. Using dedicated software, it is possible to create realistic virtual control rooms and other parts of a nuclear power plant. Software packages, like CREATE developed by Halden Virtual Reality Center (HVRC), allow rapid prototyping of 3D virtual models [1]. These models can be visualized and *experienced* in immersive environments of virtual reality systems like CAVE [2], CUBE [3] and VisBOX [4], and used for evaluation, testing and training purposes. Hence, realistic interactions with 3D models can be simulated to optimize design choices, and to make ergonomic estimations [5].

It is important to make a distinction between the functionalities of the "developer's tools" (like CREATE) that display their results on 2D monitors, and the functionalities available in the 3D immersive environment of a virtual reality system like CAVE, CUBE and VisBOX. It is currently possible to select objects and move them interactively in the developer's tools like CREATE. With some effort this capability can also be translated to the virtual reality platforms like VisBOX. This functionality in the VR environment will help speed up the prototyping and testing procedures. A second limitation is that equipment indications and controls on the panels and computer monitors of the virtual control room in the developer's tools as well as in virtual reality immersive environment are static textures, and interactivity is not often supported. It is this second limitation that has been addressed in this work. Thus, a nuclear reactor simulator has been embedded in the virtual control room. The virtual environment can therefore now display the (live) results of a nuclear reactor simulator as well as allow interactivity to operate that simulator.

In this paper we describe the preliminary integration of a generic application, like a nuclear reactor simulator, in a VR system. Moreover, implementation of interactive control of the application in the VR environment using a WEB-based interface is also described. The hardware equipment and software used in the development of this project is given. Finally the results of a nuclear reactor simulator embedded in a virtual nuclear reactor control room are shown.

II. VIRTUAL NUCLEAR CONTROL ROOM

Given a static virtual control room, goal of an interactive, virtual simulator in the VR environment can be broken down into two sub-goals: display of contents of a live and dynamically evolving computer window on a flat surface in the virtual model; and allowing equivalent of mouse-actions in the virtual environment providing interaction with the virtual environment and triggering events.

II.A. Embedded Application in Nuclear Control Room

To display the results of the reactor simulator in the virtual environment, a generic capability has been developed to display the content of any window of choice in a computer monitor on a flat surface of choice in the virtual environment. Window of choice may be displaying the content of a WEB browser, graphical output of a simulator, or a movie. Realizing that results of almost any application (a simulator output, a movie, etc) can be displayed in a WEB browser, and to take advantage of the standardization and portability that accompanies WEB-based tools, specific application developed here is focused on displaying the content of a web browser window in the virtual environment. One can open any page of choice in the web browser and hence display it in the virtual environment. Hence, the machine running the VR application must also be connected to the web and must have a WEB browser.

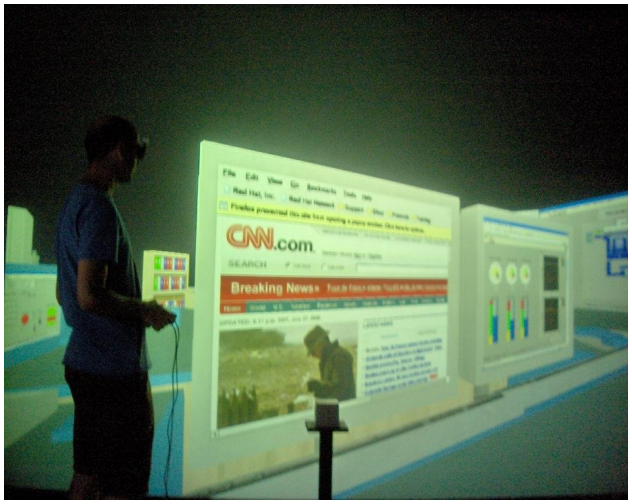


Fig. 1. The CNN WEB-page displayed in the virtual control room, as an example of output of a generic application running in the virtual reality environment.

The WEB browser runs on an X server, called Xvfb, that instead of rendering the image to the standard display renders off screen to a memory area. The VR application captures the graphical output of the WEB browser running on the X virtual server as a bitmap and converts it to an OpenGL texture. The texture is then placed on a specified polygon and displayed in the virtual control room. At each graphical callback, the image of the WEB browser on the X virtual server is captured, converted to texture and mapped to the selected flat surface in the virtual control room. This idea of employing an X virtual server and creating a texture of an application for the virtual system was originally formulated by Belleman, who developed the *XiVE* library [6].

Two different computers—one to run the simulator and the other for the virtual reality application—are used. Figure 2 shows a schematic diagram outlining the interaction between the two computers and the software applications. The simulator runs on a dedicated machine and a WEB-server on the same machine broadcasts the results to the world-wide-web. Two applications run on the virtual reality (second) computer: a WEB browser, that receives and displays the results of the simulator via the internet, and the VR application that renders the virtual control room and tracks the position of the VR user.

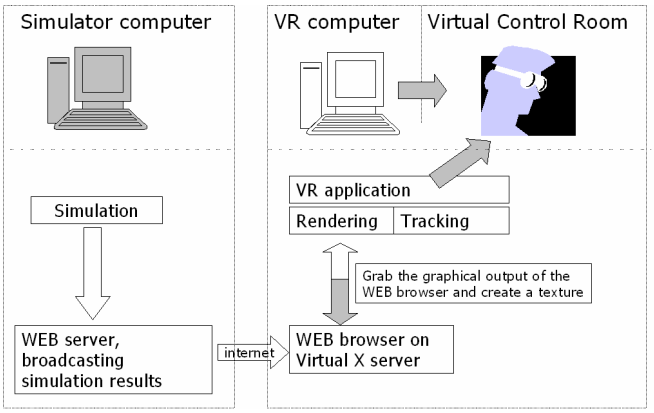


Fig. 2. Schematic diagram of a nuclear reactor simulator embedded in a virtual control room.

II.B. Interactive Nuclear Control Room

The fact that the output of the simulator is webcasted facilitates the interaction between the user in the virtual environment and the window with the dynamic output, and provides a simple mechanism for interactivity. Instead of direct interaction between a user device such as joystick, specialized glove, etc, and the simulator window, it is possible to introduce in the virtual environment one or more laptops, or tablet PCs connected to the WEB server through a wireless Internet connection [7,8]. The tablet PC can display the same front end of the simulator in a WEB browser –with interactive control buttons- as shown in the virtual control room, and hence can be used to trigger events. In this approach the *mouse action* is easily simulated in the VR environment by using a laptop computer or a tablet-pc connected to the Internet. This approach is much more natural than alternatives in which either a joystick is used in conjunction with a ray piercing the virtual button, or approaches in which a specialized glove is used by the operator to manipulate virtual buttons. Besides, it removes the complexity of programming the direct interaction between the user and the window.

II.C. Hardware and Software

The Virtual Reality Lab in the Department of Nuclear, Plasma and Radiological Engineering at the University of Illinois at Urbana-Champaign has a VisBOX immersive projection system [4]. The VR system employs one back projection 12'x 9' display screen (single wall), stereoscopic rendering, specialized audio, a magnetically tracked joystick and a wireless head tracker. See Figures 3 and 4. The computer running the VR system has an Intel Xeon 3.06 GHZ processor with 2 GByte of RAM. Large RAM is necessary in this project because the X virtual server, used in this project for the output of the application, writes to RAM instead of the memory of the graphics card.

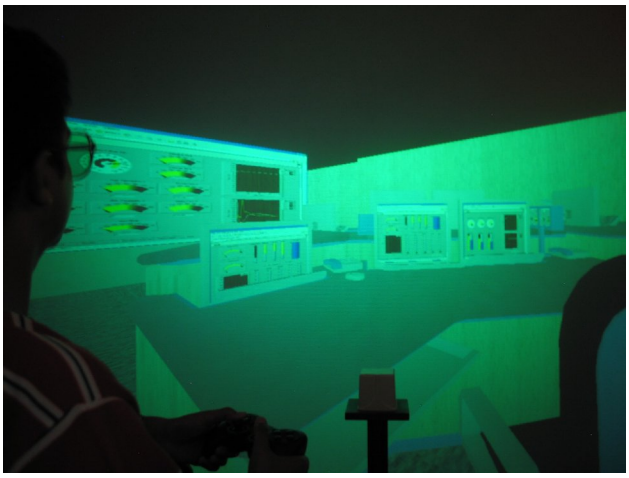


Figure 3. Virtual Reality Laboratory in the Department of Nuclear, Plasma, Radiological Engineering at the University of Illinois in Urbana-Champaign showing a virtual model of a nuclear reactor control room.

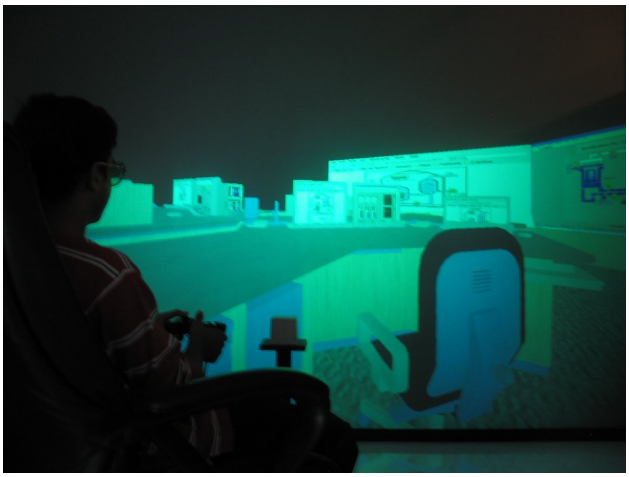


Figure 4. Another view of the virtual control room.

The operating system running VisBOX is Linux Red Hat Fedora Core. We modified a 3D, virtual model of a nuclear control room developed using CREATE, by HVRC to host the output of our nuclear simulator. The 3D model is saved in the VRML format with CREATE and then converted to an OpenGL/C file using *PolyTrans* [9]. The software for capturing the graphical output of the application running on the X virtual server is written in C (gcc compiler) using X11 and OpenGL libraries and is largely inspired by the *XiVE* library and *xwd* application [6, 10]. The source codes of both applications are available on the WEB. Moreover the *XiVE* code also allows a mechanism to use the tracked joystick to interact with the dynamic window. However, in our VR system the interaction with the simulator is via the tablet PC and an internet connection.

The choice of the language *C* is primarily dictated by the fact that all the libraries for virtual reality applications in our group have been developed in *C*. Moreover, the X11 libraries are easily accessible from *C/C++*.. However, we recognize that *Java* coupled with *Java3D* is likely to be a better programming platform for a project of this kind. CREATE and other advanced simulation packages, like *CartaBlanca*, are increasingly been written in Java. [1,11] Besides, the development in Java of WEB-servers and graphical interfaces is faster than in other programming languages

III. RESULTS

To demonstrate the capability to embed a nuclear simulator in a virtual control room, a WEB-based nuclear simulator was chosen. This would, for example, allow the display of results of the simulator running anywhere in the world as long as it is capable of webcasting its results over the World Wide Web. In a parallel development at the University of Illinois [12], a simulator-like front end has been added to the widely used RELAP5 code [13]. Taking advantage of the capabilities of LabVIEW, the RELAP results are graphically displayed and, more importantly, can be viewed in a web browser. If permitted, the remote viewer of RELAP5 output can also interact by, for example, initiating a scram by clicking on the scram button in the WEB browser [12]. It is this recently added capability to RELAP5 that is used to test the framework to embed a simulator in the virtual control room.

In Figures 5 and 6 two views of a virtual control room with the embedded nuclear reactor simulator are shown. Color-coded void fraction distributiondisplayed on the virtual screen can be seen in Fig. 5. The monitor on the right is showing the pressure level in the steam generators

and the user's laptop is also showing the same results. The large screen on the right in Fig.6 is the main control window in the LabVIEW-based RELAP5 nuclear reactor simulator. It shows the red scram button, as well as other features to select different display modes, and interactive control features. The WEB browser in the PC in the front of the user in the virtual environment is also showing the same "page". Hence the user can mimic, for example, a scram by simply tapping (on a tablet PC) or clicking on the scram button on his pc. A click on the scram button initiates the scram sequence in the simulator, and consequently all the virtual screens showing the results of the simulator, which are being continuously updated, also display the results of the scram.

Currently, the performance measured by the speed with which the simulator results are displayed in the virtual control room, is acceptable. However, large amount of data that needs to be frequently updated in the virtual control room will push the limits of the current procedure. We are currently working on improving the performance by employing techniques to faster capture the graphical output of the X virtual server. However, it is also possible to improve the performance simply by reducing the size of the window displayed in the virtual environment and therefore the memory used for bitmapping the output of the applications.

IV. CONCLUSIONS

We proposed a general methodology to embed an application output in a 3D immersive environment and a simple model to provide interactive control of the application using a WEB-based GUI. An implementation and the results of these two basic strategies for coupling a nuclear reactor simulator and a 3D VR model have been shown. The future improvements for this project could be a new design for the software package and a better organization of the code.

We believe that the embedding of a interactive nuclear reactor simulator in a VR environment is another step toward the use of virtual experience of a control room to not only test the control room design features, but also to train reactor operators on a virtual simulator.

ACKNOWLEDGMENTS

The authors would like to acknowledge Paul Rajlich of VisBOX for the important and continuous help during the development of this project. This work was supported in part by a DOE INIE grant and a DOE NEER grant.

REFERENCES

1. HALDEN VIRTUAL REALITY CENTER Web page, http://www.ife.no/laboratories/hvrc1/, (2006).
2. C. CRUZ-NEIRA, D. SANDIN, T.DEFANTI, *Virtual Reality: The Design and Implementation of the CAVE®*. Proceedings of SIGGRAPH 93 Computer Graphics Conference, ACM SIGGRAPH, (1993).
3. "THE CUBE" WEB page, http://www.isl.uiuc.edu/Labs/room_b650.htm, (2006).
4. VISBOX INC. WEB page, http://www.visbox.com/, (2006).
5. RIZWAN-UDDIN, N. KARENCEVIC, S. TIKVES, *Virtual Reality at the service of GEN-IV, and V, and ...*, Proceedings of ICAPP-2003, (2003).
6. XiVE WEB page, http://staff.science.uva.nl/~robbel/XiVE/, (2006).
7. V.E. WHISKER, A.J. BARATTA, T.S. SHAW, J. W. WINTERS, J.A. CLELLAND, F.T. JOHNSON, *Simulating Nuclear Power Plant Maintenance Activities Using Immersive Virtual Environments*, Proceedings of ICAPP-2004, (2004)
8. IOWA STATE UNIVERSITY VESUITE WEB page, http://www.vesuite.org/, 2006.
9. POLYTRANS WEB page, http://www.okino.com/conv/conv.htm, (2006).
10. xwd.c code WEB page, http://stuff.mit.edu/afs/sipb/user/qjb/source/hacks/xwd/xwd.c, (2006).
11. W.B. VANDERHEYDEN, E. D. DENDY, N.T. PADIAL-COLLINS, *CartaBlanca— a pure-Java, component-based systems simulation tool for coupled non-linear physics on unstructured grids*, Proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande, (2001).
12. K.D. KIM, RIZWAN-UDDIN, *A web-based nuclear simulator using RELAP5 and LabVIEW*, manuscript in preparation
13. IDAHO NATIONAL ENGINEERING & ENVIRONMENTAL LABORATORY (INEEL), *RELAP5 manual revisions 2.2 and 2.3*, http://www.inel.gov/relap5/r5manuals.htm, (2006).

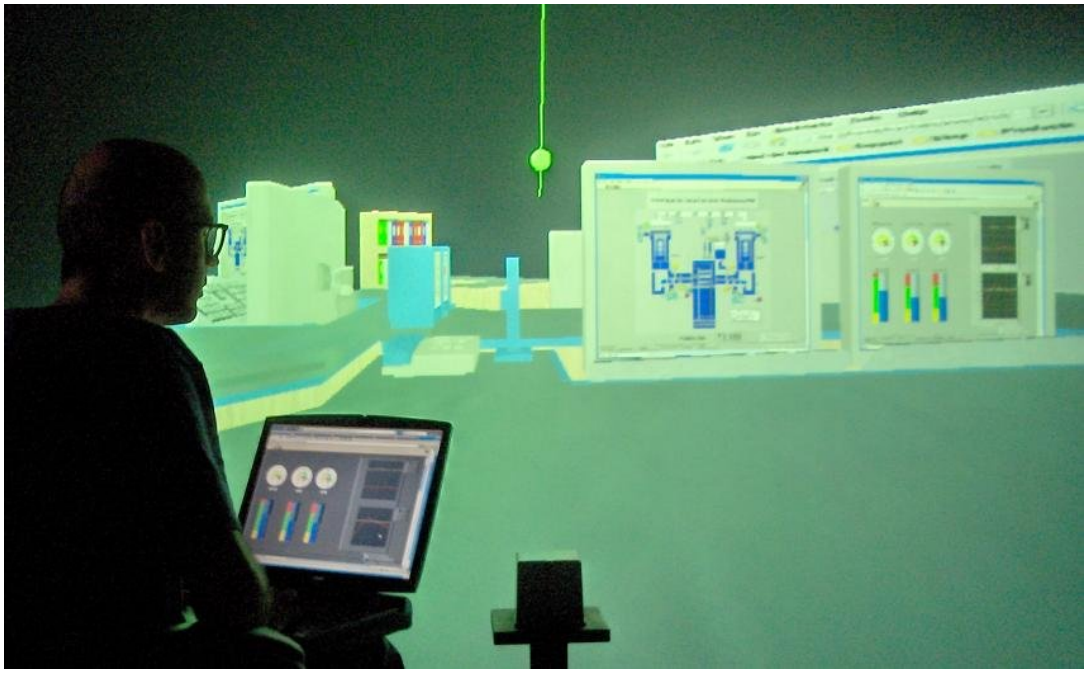


Figure 5. Picture of the virtual control room. One of the screen displays color-coded void fraction distribution in the reactor

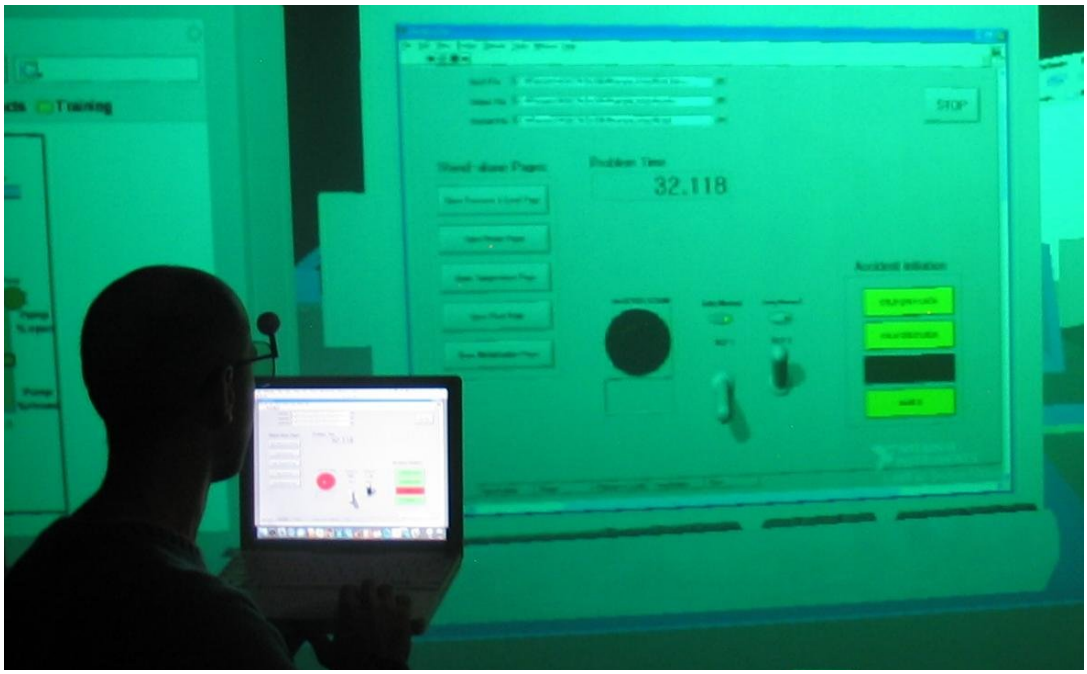


Figure 6. Another view of the virtual control room. The red button on the virtual monitor if (virtually) pressed will scram the reactor.

APPENDIX D

Three Dimensional, Virtual, Game-Like Environments for Education and Training

Joel Dixon, Stefano Markidis, Cheng Luo, Jared Reynolds, Rizwan-uddin

University of Illinois at Urbana-Champaign
103 S. Goodwin Ave, Urbana, IL, 61801, and jdixon2@uiuc.edu

INTRODUCTION

Training emergency response personnel to handle a hazardous radiation spill is a challenging task. Conducting repeated training sessions in dangerous or restricted areas in and around research reactors or NPP sites is not a realistic option. Computer games provide an alternative training mode [1]. Recent developments in 3D, virtual environments can be inexpensively and effectively utilized to train in a virtual environment, to communicate the dangers, and train on procedures to be used when dealing with a radiation emergency. While currently adequate, education and training to enhance safety and security at NPP sites can also be much more effectively and efficiently carried out using 3D, virtual animation technology [1-6]. Such models have also been used to train the decommissioning personnel. Different options are available to quickly and efficiently develop the desired 3D models. One of them is the Unreal Engine—a platform in which 3D, game-like environments can be developed quickly and easily with sophisticated details and interactivity. We here report the development of a fairly realistic model of the University of Illinois TRIGA reactor, which is now destined for decommissioning, using the Unreal Engine. The model can be used to train first responders, decommissioning personnel, and even to effectively communicate reactor safety and security issues to students and nuclear personnel.

MODELING THE ENVIRONMENT IN UNREAL

The editor for the Unreal Engine, *UnrealEd* (UED), is designed to be simple and easy to use so that even beginners with no programming or modeling experience can use to create their own *levels* in the game with very little training. (See glossary at the end of this paper for definitions of terms like *level*, etc.)

When developing a *level* in UED, the computer screen is divided into four windows as shown in Fig. 1a. The upper-left, upper-right, and lower-right windows respectively show the standard top, front and side views of the model, respectively. These are flat 2-D views of the model. The lower-left window is the view port, which shows the view from a camera that can be moved around the level while editing. There are thousands of pre-built items that can be selected from a list and simply inserted in the map. For example, to make a room with a table, one needs to select a pre-built table and copy-paste it onto the floor. The object can then be moved, rotated, or scaled. The editing of the map can be broken down and distributed among workers to work on different sections that can be combined later into a single final map.

There is a large community of fans and programmers that maintain several internet sites dedicated to helping people with UED problems [7, 8]. Questions posted at these sites are quickly answered by experts. Codes and modifications made by experts are also available.

In addition to having a simple interface, UED also has the programming complexity to support many advanced features for the simulation. Some of these which are directly relevant to the nuclear education and training related applications are described below.

Using Pictures for Realistic Effects

UED allows pasting of pictures captured by ordinary digital cameras onto surfaces in the model. This feature is very useful in giving a realistic impression of the models. Figure 1b shows screen shot of the reactor control room with pictures of the actual control panels pasted on the model control panel surfaces. The reactor bay is visible through the glass window.

Lava Zones

Lava zones are regions set in the map that are hazardous. If a player (worker) stays in the *lava zone*, his health index drops by a set amount each second. This property, called DamagePerSec, can be used to mimic the radiation level [1]. An increasing value of DamagePerSec indicates a higher radiation level in the *lava zone*.

Health Index

Usually used in computer games to show the number of punches or bullets a player has received, this feature, when combined with *lava zones*, can be used as a dosimeter showing dose received while working in a radiation field. The player's health at the beginning of a simulation is at "full," say around 100. As the worker walks into the calibrated radiation field (*lava zones)*, his health index will drop in proportion to the level of radiation in the area. At the end of the simulation the total loss in health index can be related to a dose received.

Visual Representation of Radiation Effects

To help first responders, maintenance, and decommissioning personnel visualize radiation in the facility, the map floor can be color-coded to show the level of radiation. Several such fields can be turned on or off during the training exercise to show the radiation level, temperature, etc.

RESULTS

A rather detailed model of the UIUC TRIGA has been developed. Additional details are still being added. Inside the reactor building, different floor levels, staircases, control room, octagon shaped structure with the pool and the core, and the bay area have been modeled. The model is roughly to scale. See Fig. 1c. The model includes the surrounding buildings, streets and parking lots. Figure 1d shows the reactor building environment. Only the reactor building has the inside details. A player can start from outside and walk into the reactor building. Digital pictures of the actual facility are being used to make the facility as realistic as possible. *Lava zones* and the health index have been tested. Different segments of the floor can be colored to "show" the level of radiation in that area. Horizontal surfaces in Fig. 1c are color coded showing a fake radiation field emanating from the reactor core. Radiation decreases as with increasing distance from the core. Several details have been added to the model of the water pool including the realistic looking water surface effects. Currently a single player (worker, avatar) can walk through the model.

Training Model

One of the first training modules developed, is to train a group of firefighters in a simulation of a radiation spill at their local research reactor. They (or rather, their avatars) first get the briefing from their leader as they arrive in their vehicles. As they arrive on the scene, they pick up a Geiger counter and begin measuring the radiation level. Upon entering the plant they "see" a spill and visualize the levels of radiation around it. They reach the controls needed to secure the area using a safe path that is free from radiation. After securing the site, they leave the building and go through decontamination.

FUTURE WORK

Several features of the Unreal engine have not been tapped yet. For example, it allows multiple players (workers) to "play" the game simultaneously. It allows access and simultaneous gaming capability via the net with voice communication. Work is currently underway to allow simultaneous and interactive training of multiple people with access to the internet from anywhere around the world. In addition, we will also explore the possibility to integrate this model into a stereographic system to provide true 3D immersive experience.

GLOSSARY OF TERMS

Level:	The model of the reactor building and surroundings
Map:	Used interchangeably with level.
Health Index:	A number that represents the health level of a player. It decreases as health is damaged. The default (a healthy person) has a health index of 100 in Unreal.
Camping:	An inactive player

REFERENCES

1. ROBERT L. SANDERS, JOSEPH E. LAKE, *Training First Responders to Nuclear Facilities Using 3-D Visualization Technology*, Proceedings of the 2005 Winter Simulation Conference, pp. 914-918 (2005).
2. JOHN GRIFFITH, JIANWEI HU, NICK KARANCEVIC, FEDERICO E. TERUEL, YIZHOU YAN, JAMES SUTBBINS and RIZWAN UDDIN, *Research and Virtual Reactor Design at UIUC, Trans. ANS*, 93 pp. 873-874 (2005).
3. STEFANO MARKIDIS, RIZWAN-UDDIN, A *Virtual Control Room with an Embedded, Interactive Nuclear Reactor Simulator*, Proc. HMIT, ANS (2006).
4. NICK KARANCEVIC, RIZWAN-UDDIN, *The Virtual Nuclear Laboratory*, Proc. Fourth American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Controls and Human-Machine Interface Technologies, Columbus, Ohio, (2004)
5. NICK KARANCEVIC, RIZWAN-UDDIN, *Towards Virtual GEN-IV Reactors*, Transactions of the American Nuclear Society, Washington, D.C., Vol. 91, pp. 912-914, (2004).
6. NICK KARANCEVIC, RIZWAN-UDDIN, Virtual System for Understanding and Advancement of Nuclear Power, *Trans. ANS*, 88 (2003).
7. Beyond Unreal (http://forums.beyondunreal.com/)
8. Epic Games Forums (http://forums.epicgames.com/ index.php)

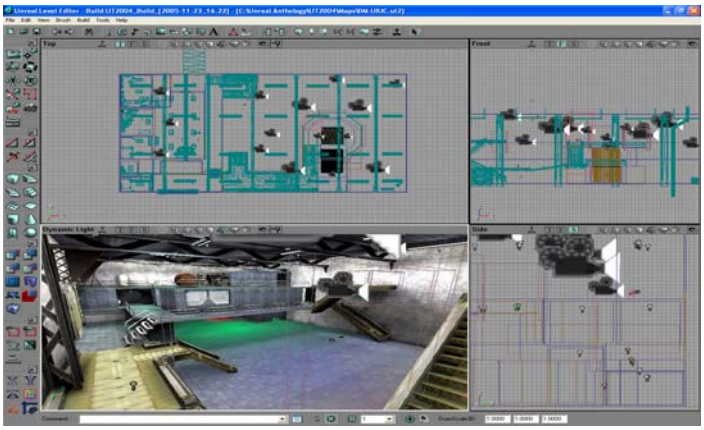


Fig. 1a. A view of the Unreal editor's windows showing the model.

Fig. 1b. A view of the control panel.

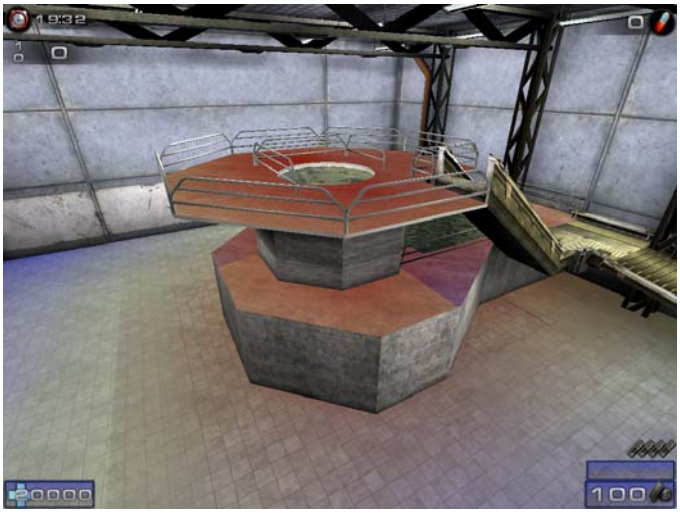


Fig. 1c. Model of the reactor bay. Flat surfaces are color-coded with a fake radiation field with highest levels at the core.

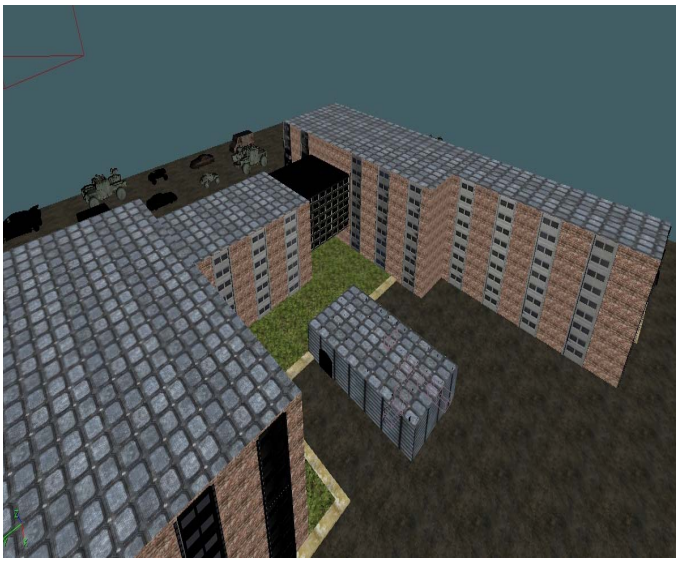


Fig. 1d. An overview of the entire reactor structure and surrounding buildings and streets. The reactor facility is the small building in the center.

List of publications based on work at least partially supported by this DOE NEER grant. (Additional publications are expected)

Nick Karancevic and Rizwan-uddin, **Virtual Systems for Understanding and Development of Nuclear Power**, in *Trans. Amer. Nucl. Soc*., **88** (2003).

Nick Karancevic, J. Stubbins and Rizwan-uddin, **A Virtual Laboratory and Control Room**, in *Trans. Amer. Nucl. Soc*., **90** (2004).

Nick Karancevic, Rizwan-uddin, **Towards Virtual GEN-IV Reactors**, in *Trans. Amer. Nucl. Soc*., **90** (2004).

Nick Karancevic and Rizwan-uddin**, The Virtual Nuclear Laboratory,** in *Proc. Fourth American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Controls and Human-Machine Interface Technologies (NPIC&HMIT 2004)*, Columbus, Ohio, September, 2004.

Koji Sugawara, Rizwan-uddin, **Automated Conversion of Keno Input Files to MCNP Input Files and Comparison of Keno and MCNP Results,** in Proc. *The Monte Carlo Method: Versatility Unbounded In A Dynamic Computing World*, Chattanooga, Tennessee, April 17–21, 2005, American Nuclear Society, LaGrange Park, IL (2005).

John Griffith, Jianwei Hu, Nick Karancevic, Federico E. Teruel, Yizhou Yan, James Sutbbins and Rizwan Uddin, **Research and Virtual Reactor Design at UIUC**, in *Trans. Amer. Nucl. Soc*., **93** (2005).

K.D. Kim, Prashant Jain and Rizwan-uddin, **Web- and System-code Based, Interactive, Nuclear Power Plant Simulators**, in Proc. 5th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies, Albuquerque, USA, Nov 12-16 (2006).

Joel Dixon, Stefano Markidis, Cheng Luo, Jared Reynolds, Rizwan-uddin, **Three Dimensional, Virtual, Game-Like Environments for Education and Training,** *Trans. ANS*, **97** (2007).

K.D Kim and Rizwan-uddin, **A Web-based Nuclear Simulator Using RELAP5 and LabVIEW**, *Nucl. Engg. and Design*, **237**, 1185-1194 (2007).