Conf-920431--8

GRAPHICAL DEBUGGING OF COMBINATORIAL GEOMETRY*

CONF-920431--8

DE92 006415

Thomas J. Burns and Mark S. Smith
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory**

Paper to be Presented at the
American Nuclear Society
Topical Conference on
New Horizons in Radiaiton
Protection and Shielding
April 26-30, 1992
Pasco, Washington

*Research sponsored by the Defense Nuclear Agency
**Managed by Martin Marietta Energy Systems, Inc. under Contract No. DE-AC05-84OR21400 for the U. S. Department of Energy

GRAPHICAL DEBUGGING OF COMBINATIONAL GEOMETRY

Thomas J. Burns and Mark S. Smith
Oak Ridge National Laboratory
P. O. Box 2008, MS
Oak Ridge, Tennessee 37831-6363
Phone (615) 574-6101

ABSTRACT

A graphical debugger for combinatorial geometry being developed at Oak Ridge National Laboratory is described. The prototype debugger consists of two parts: a FORTRAN-based "view" generator and a Microsoft Windows application for displaying the geometry. Options and features of both modules are discussed. Examples illustrating the various options available are presented. The potential for utilizing the images produced using the debugger as a visualization tool for the output of the radiation transport codes is discussed as is the future direction of the development.

I. INTRODUCTION

A recurring difficulty in shielding and transport applications is the verification that the geometric model (and associated material assignments) accurately represent the physical configuration. Exacerbating this difficulty is the fact that for some problems, the number of objects (zones, regions, bodies etc.) used to describe the physical configuration can easily number in the hundreds or thousands. Additionally, the geometric descriptions in the stochastic codes are complicated by the Boolean operators AND, OR, and NOT used to combine the fundamental objects. Both these facts tend to mitigate against the successful verification and debugging of the geometric models directly from the input stream of the radiation transport codes. A clear need exists for a means of analyzing and debugging geometric models before consuming hours of CPU time on a supercomputer. Such a tool, a graphical debugger for combinatorial geometry, is currently being developed at Oak Ridge National Laboratory.

II. IMPLEMENTATION

The ultimate objective of this effort is the development of a XWindows application which can be executed on a high-end workstation. This application should be capable of accepting geometric input in a variety of formats (such as those currently used for the radiation transport codes as well as those generated by CAD/CAM packages). It is anticipated that the application will be capable of real-time generation of images representative of the combinatorial geometry input suitable for validating (and correcting if necessary) the model. Some of the proposed features of the final application are the capability of rotating the displayed object(s) arbitrarily, the capability to zoom in order to examine details of the object, and capacity for selective invisibility (i.e., to remove specified zones, regions, or materials from the image).

To facilitate initial development and testing, the prototype of the graphical debugger was constructed as two modules. The first half of the debugging system is a "view" generator designed to be executed on the same computer used to run the transport code.[1]. In order for the debugger to be truly useful, one of the principal design criteria used was that image produced by the "view" generator accurately represent the geometry as processed by the transport codes. That is, the view generator should "see" the model exactly as the transport codes do. To this end, the ray-tracing routines utilized in the Monte Carlo codes, MORSE[2] and MASH-GIFT[3] form the basis of the view generator.

An input processor was written to handle input data streams formatted for MORSE, MASH-GIFT, and TORT[4] and is being extended to handle the IGES (Interim Graphics Exchange Specification)[5] format. Options permit the user to select an arbitrary viewpoint in space, and the choice of either an isometric or perspective view of the geometry. The code also permits the user to select the degree of aggregation to be displayed in a particular view. The options include a single body, a single zone (Boolean combination), a single region (a group of Boolean combinations with the same region identifier), all zones comprised of a particular material, or the entire model. Two mechanisms for examining interior details of complex models have been implemented so far. The code permits selective invisibility, i.e., any combination of zones, materials, or regions can be flagged as invisible or transparent to the view generator. Alternatively, an arbitrary body (i.e., any body which can be described using MORSE or GIFT geometry) can be "cut away" from the geometry to facilitate debugging internal details. Both of these options can be employed simultaneously. Additional options to the view generator permit the selection of an appropriate aspect ratio for the view, as well as the specification of the desired resolution for the final output. The view generator is designed to produce a metafile, which consists of a compressed version of the "view" itself, which serves as input into the second half of the system.

The second part of the CG debugging system consists of a Microsoft Windows application, ORGBUG[6], that executes on an IBM compatible personal computer, which accepts the metafiles produced by CGVIEW. The choice of Microsoft Windows as the graphical user interface (GUI) for the prototype debugger was made primarily as a function of convenience and expediency. The use of Microsoft Windows tends to minimize hardware dependencies since it supports a variety of display and output devices. Additionally, the basic design philosophy of Windows and Xwindows is similar (they are both event-driven) which will simplify the eventual porting to the workstation.

In addition to reading the metafiles generated by CGVIEW, ORGBUG is structured to display the image contained in the file. As a Windows application it permits (via scroll bars) the image to be larger (in terms of resolution) than the window itself. As a consequence of the decision to split the debugger into two modules however, the image displayed is a fixed one, i.e., no rotation or zoom functions are implemented. Either a monochrome (wireframe) or color image can be selected. If a color representation is selected, "colorization" can be done on a zone, material, or region basis using either a default palette or user-assigned colors.

As a consequence of being linked to the metafile (which contains the zone, body, and surface identifiers for each pixel), ORGBUG implements a identification option. Using a pointing device, the user can request the identifiers for any visible pixel. This feature has proved to be invaluable for debugging complicated geometries.

Eventually hardcopy output will be possible via devices supported by Microsoft Windows including dot matrix printers, laser printers, and film recorders. However, at this point the development, ORGBUG has only limited output options. The debugger can perform file output to either the Windows bitmap format (BMP) or the PCX format. Additionally, the debugger supports the Windows Clipboard. This latter feature permits the transfer of the image to other Windows applications which have hardcopy output features. Similarly the BMP and PCX file output options permit image transfer to other PC programs for modification and/or output.

III. ILLUSTRATIONS

The best way to depict the utility of the geometry debugger is to show the output produced. Views produced using three different combinatorial models are discussed in this section. As noted above, the system can produce color output. This is particular useful if the various colors are mapped to the model materials. However, due to publication limitations, the various figures used in this section were generated using the wireframe option (black wires on a white background).

To illustrate some of the capabilities of the graphical debugging system, Figures 1a-1d were generated based on a geometric model formatted for the MORSE Monte Carlo radiation transport code. The geometric model itself was originally created as part of a shielding and neutronic analysis for the TFTR (Tokamak Fusion Test Reactor)[7]. Figures 1a-1d were generated by selecting only those objects which were assigned the same material identification, corresponding to the toroidal field coils, the poloidal field coils, the torus itself, and the support structure. Although the default scaling option of the geometry debugger is to scale the selected objects to fill the viewing frame, this particular option was disabled so that the four figures would be scaled identically. This allows the relative positions and sizes of the various objects to be maintained across multiple frames. It should be noted that, at the time this model was created, MORSE did not have a toroidal body. Hence, the torus depicted in Figure 1c is modeled as a set of short cylindrical annuli ORed together. Note that, in this particular model, one of the annuli is missing.

Figure 2 represents an isometric view of the entire geometry of the TFTR, i.e., the aggregation of the pieces depicted in Figures 1a-1d. Additionally, the view was generated by specifying that a 90 degree wedge was to be "cut away" from the geometry. Use of this option is one way of maintaining a perspective relative to the entire geometry, while simultaneously permitting the analyst to examine internal structures of relatively complex groups of objects.

Although discrete ordinates codes typically do not employ combinatorial geometry, the fact that TORT uses RPPs (i.e., a MORSE rectangular parallelepiped) to define regions, and then overlays those regions onto a space mesh allows the geometry debugger to be employed. The overlay scheme utilized implies a set of Boolean operations which the geometry debugger constructs. Figure 3 illustrates the use of the geometry debugger based on a TORT input deck. The model is that of the Chinzei school which was part of previous radiation transport study[8]. Figure 3a depicts a perspective view of the building with the right front quarter removed in order to expose the interior structures such as floors, walls, and ceilings.

By judicious selection of options, certain extremely useful graphical descriptions of a particular model can be produced. For example, Figure 4 is the result of requesting an isometric view of the Chinzei school model, setting the viewpoint to be directly overhead, and cutting away a half space. The result is a depiction of the first floor plan of the building.

A second mechanism for displaying the internal details of a complex set of objects is to use the selective invisibility feature of the geometry debugger. Figure 5 depicts an isometric of a Soviet BMP (armored personnel carrier)[9]. The model is described in GIFT4 geometry for use with MASH

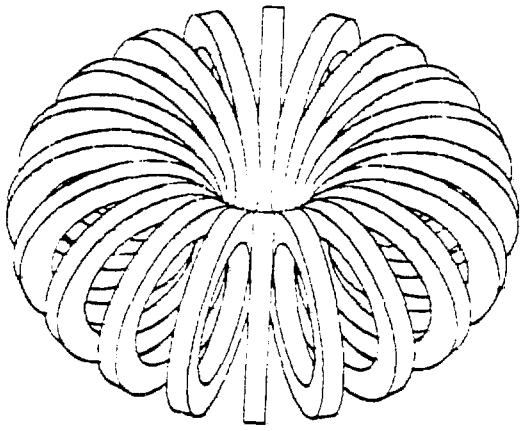

Figure 1a. TFTR Toroidal Field Coils.

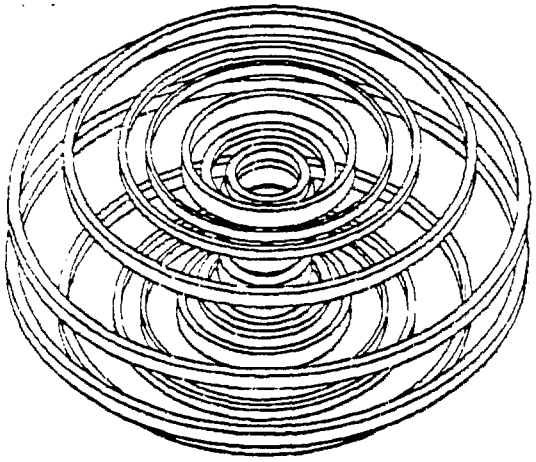

Figure 1b. TFTR Poloidal Field Coils.

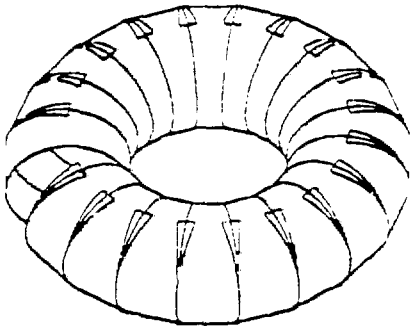

Figure 1c. TFTR Torus.

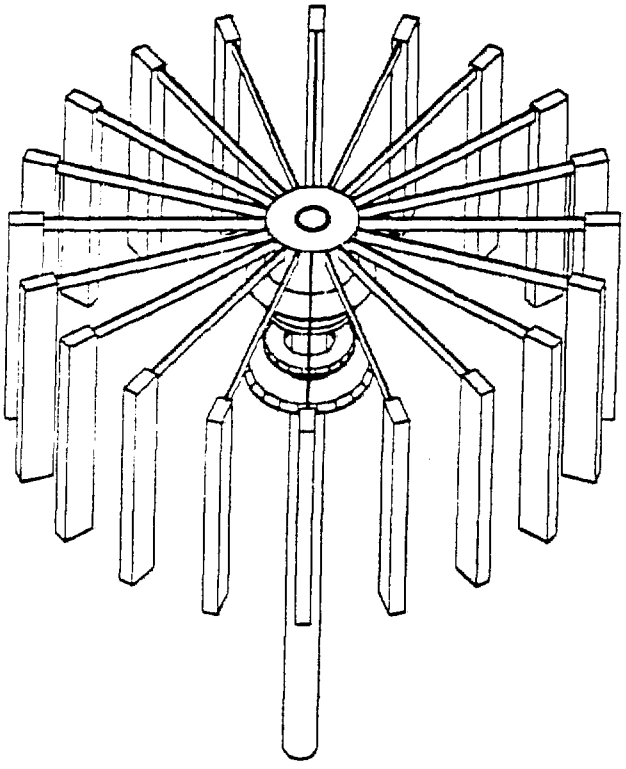

Figure 1d. TFTR Support Structure.

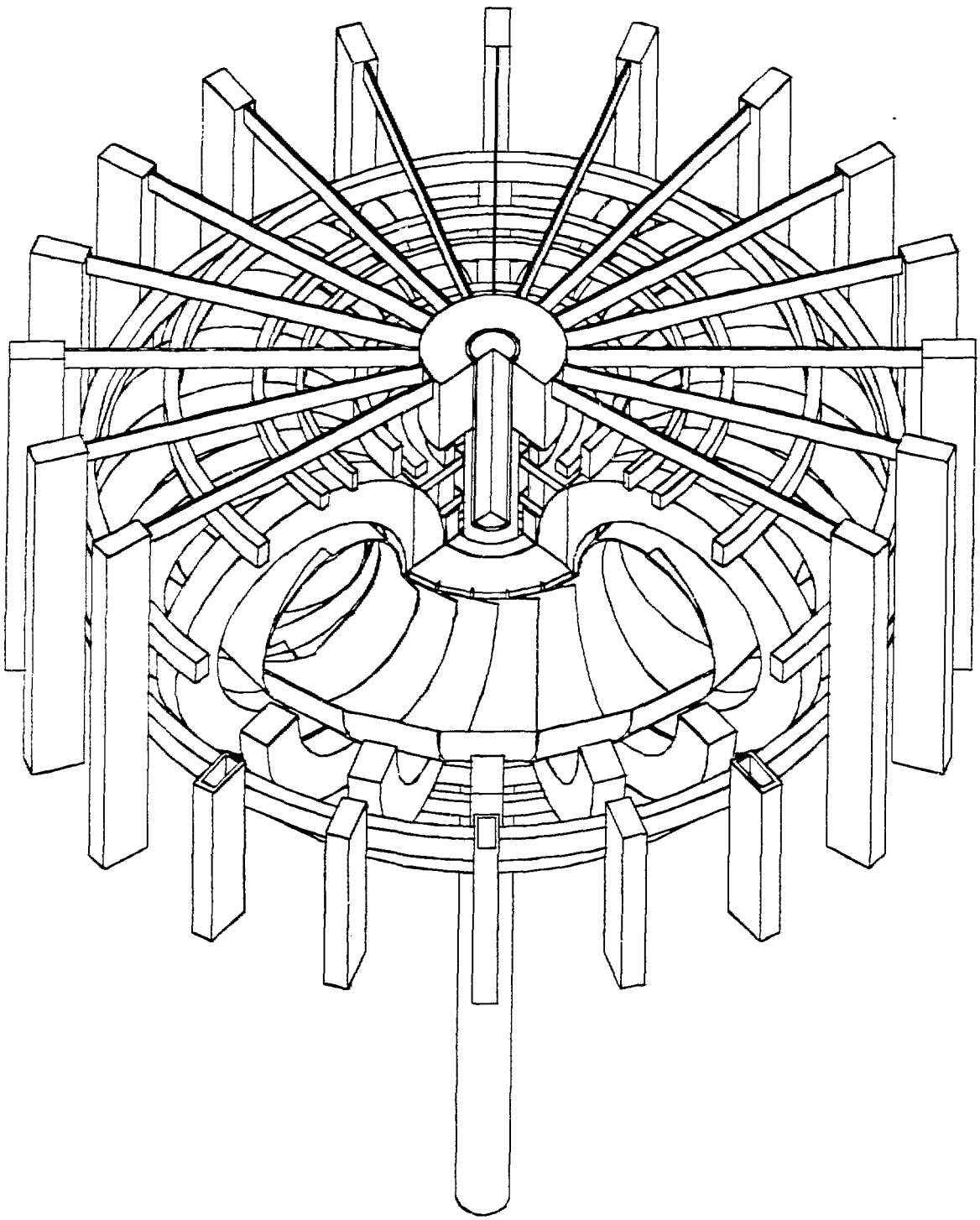

Figure 2. TFTR Model with Cutaway.

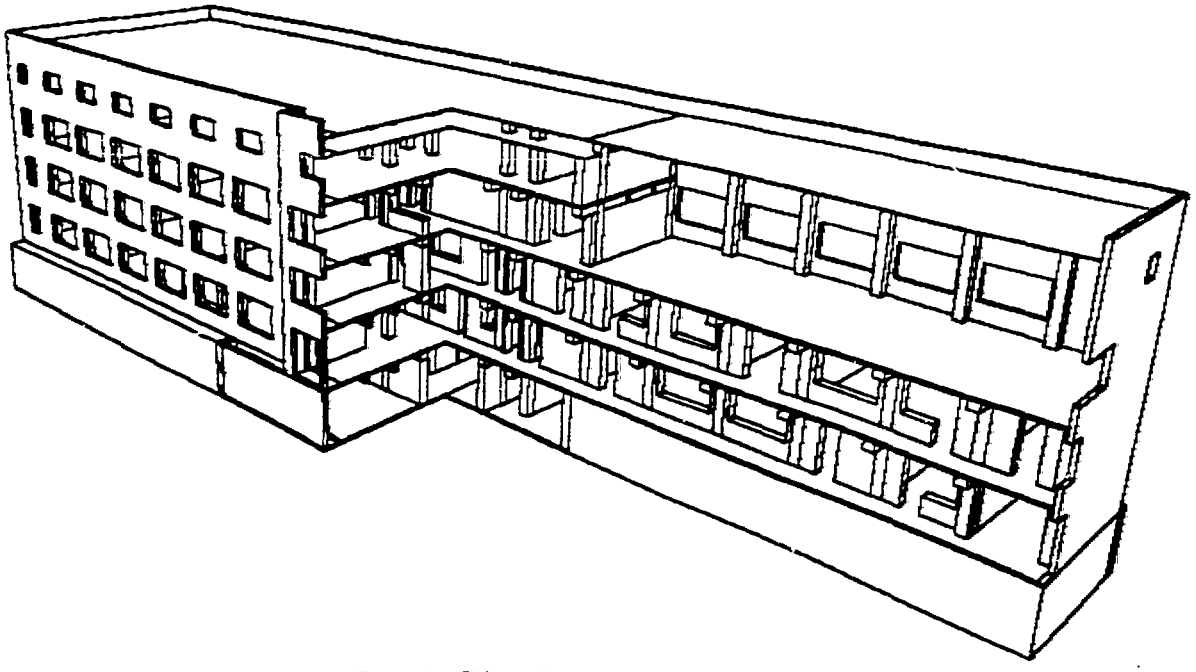

Figure 3. Chinzei School Building with Cutaway.

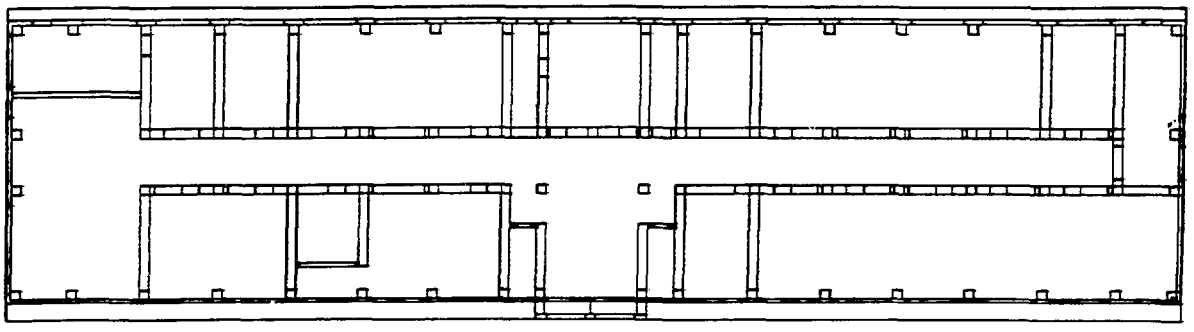

Figure 4. Chinzei School Building - First Floor Plan.

code. By selectively flagging specific zones regions, and materials as invisible, Figure 5 can be converted to Figure 6, i.e., the gun turret and top armor can be removed to show the internal details of the model. Using the selective invisibility feature for multiple views permits views such as Figure 7 to be constructed. The invisibility flags utilized in Figure 6 were reversed for a second view using the same viewpoint and scaling parameters. The second view was then overlaid (and displaced slightly) on Figure 6. The result is an "exploded" view of the Soviet BMP.

IV. VISUALIZATION

Although the initial purpose of the geometry debugger was to permit the analyst to validate and correct the INPUT for the radiation transport codes, it readily became apparent that the images produced could also play a significant role in displaying the output of the radiation transport codes as well. Both the discrete ordinates codes (TORT) and the stochastic codes (MORSE and MASH) can produce significant quantities of output data. The images which can be generated via the

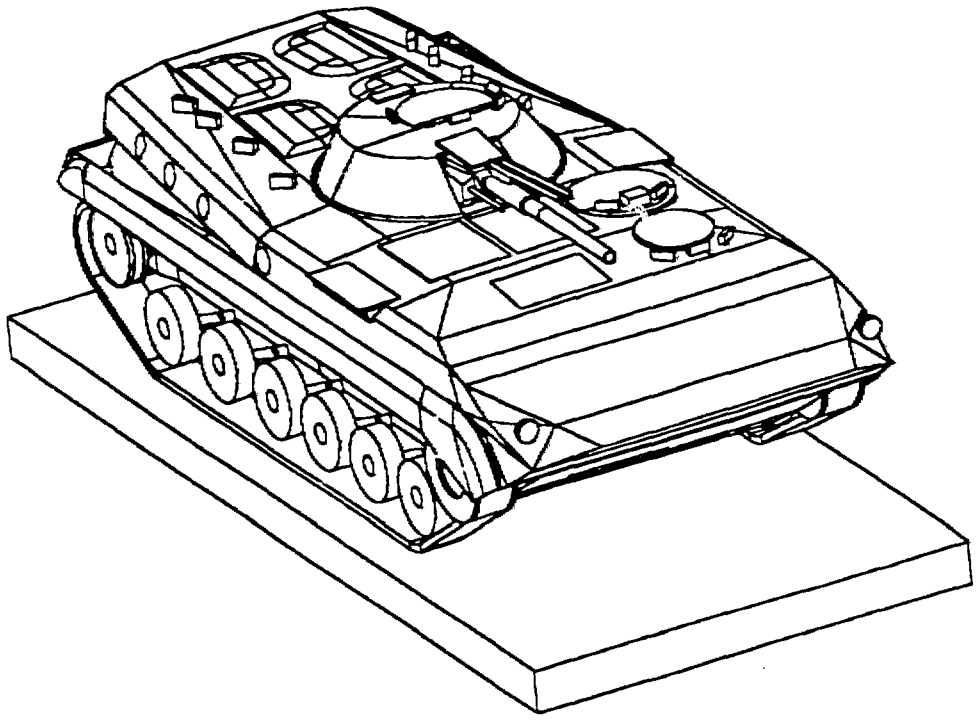

Figure 5. Soviet BMP Model.

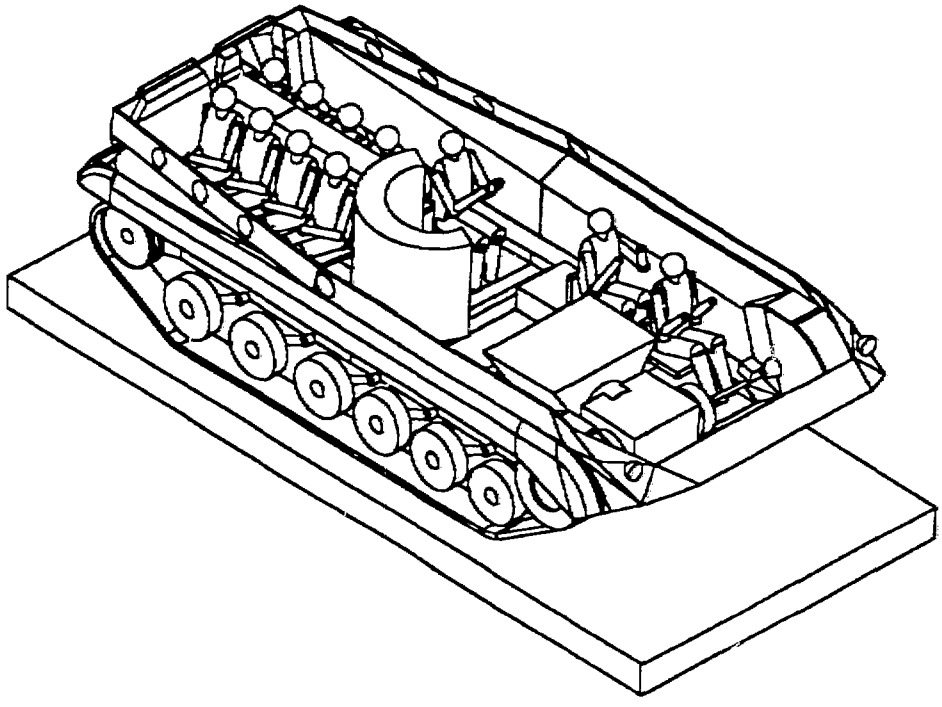

Figure 6. Soviet BMP Model - Turret and Top Armor Removed.

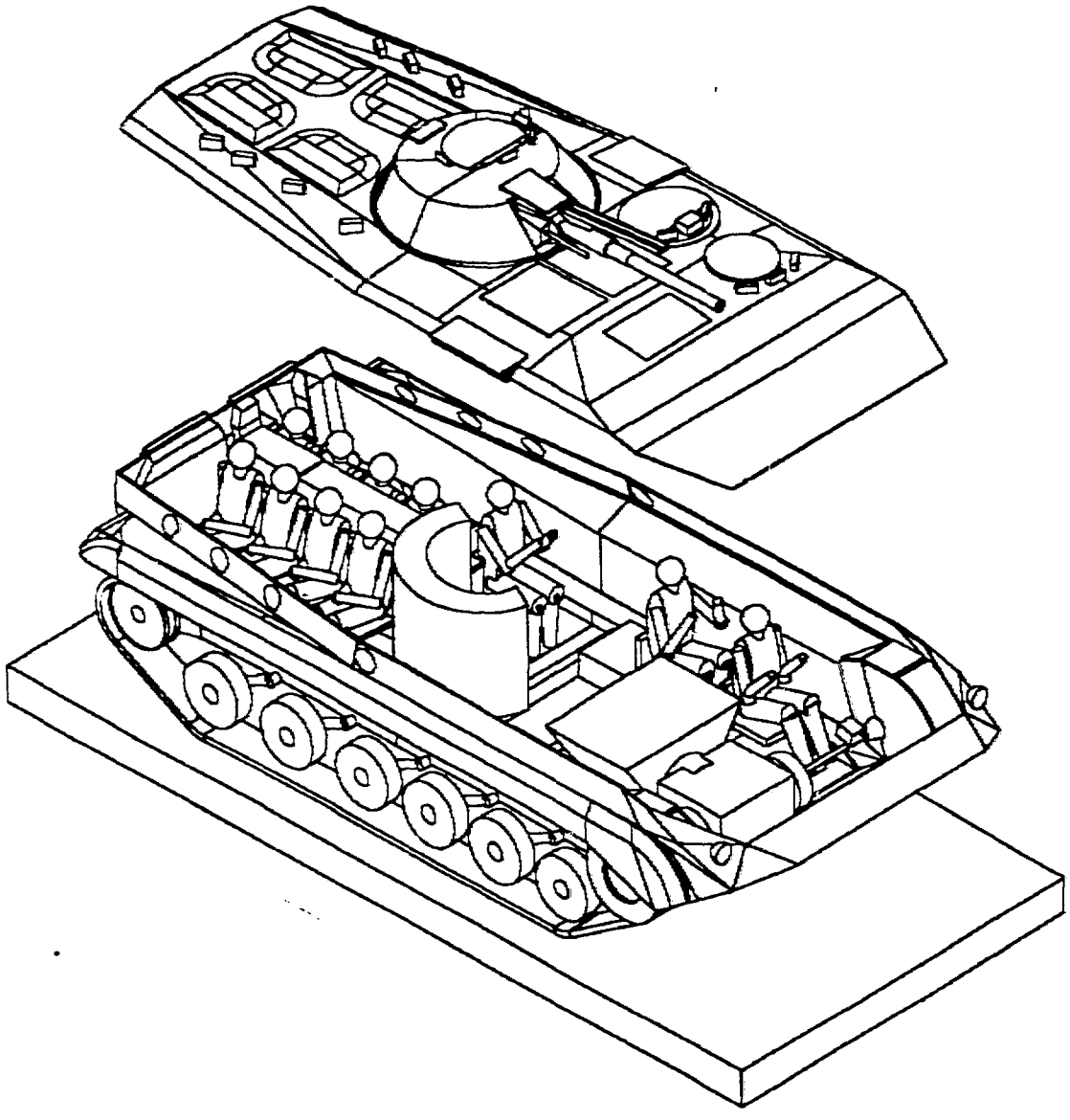

Figure 7. Soviet BMP Model - Exploded View.

geometry debugger can provide a mechanism for visualizing what can be an enormous amount of data. For example, combining views such as Figure 4 with existing output manipulation codes like the DOGS[10] system would permit the overlay of flux contours directly on the floor plan of the building. Moreover, it should also be possible to map color coded response values on view such as Figure 3, in effect, mapping the response values onto the exterior of the building as well as onto the floors, walls, and ceilings of model. In a similar vein, the output of MASH (typically a leakage flux as a function of energy and position) could be color coded and mapped onto an image such as Figure 5 to display the geometry vulnerability to radiation effects.

V. FUTURE DIRECTION

The ultimate goal of this project is to combine the two parts into a single Xwindows application running on a workstation. This will permit real-time display and debugging of the geometric models used in radiation transport applications.

VI. REFERENCES

1. T. J. BURNS, "CGVIEW - A Program to Generate Isometric and Perspective View of Combinatorial Geometries", ORNL/TM-12019, Oak Ridge National Laboratory, (To be published).

2. M. B. EMMETT, "The MORSE Monte Carlo Radiation Transport Code System," ORNL-4972 (1975, ORNL-4972/R1 (1983), ORNL-4972/R2 (1984), Oak Ridge National Laboratory.

3. J. O. JOHNSON et al., "A User's Manual for MASH 1.0 - A Monte Carlo Adjoint Shielding Code System," ORNL/TM-11778, Oak Ridge National Laboratory, (to be published 1991).

4. W. A. RHOADES and R. L. CHILDS, "The TORT Three-Dimensional Discrete Ordinates Neutron/Photon Transport Code," ORNL/TM-6268, Oak Ridge National Laboratory, (November 1987).

5. BRADFORD SMITH et al., "Initial Graphics Exchange Specification (IGES), Version 4.0," SP-767, Society of Automotive Engineers,(June 1988).

6. T. J. BURNS, "ORGBUG - A Windows-based Combinatorial Geometry Debugger", ORNL/TM-12020, Oak Ridge National Laboratory, (To be published).

7. R. T. SANTORO et al., "Comparisons of Calculated and Measured Spectral Distributions From a 14-MeV Neutron Source Inside the Tokamak Fusion Test Reactor", ORNL /TM-9888, Oak Ridge National Laboratory, (December 1985).

8. W. A. RHOADES, R. L. CHILDS and D. T. INGERSOLL, "Radiation Exposure Inside Concrete Buildings at Nagasaki", ORNL /TM-10569, Oak Ridge National Laboratory, (May 1989).

9. C. M. WARD, Personal Communication, Foreign Science and Technology Center, November 15,1989.

10. D. T. INGERSOLL and C. O. SLATER, "DOGS - A Collection of Graphics for Support of Discrete Ordinates Codes", ORNL/TM-7188, Oak Ridge National Laboratory, (March 1980).