# Using Commercial CAD Packages to Interface with 3D EM Modeling Codes

T. H. Hubing and G. K. Bhat
University of Missouri-Rolla
Rolla, MO 65401

**Abstract**

Generating input files for three-dimensional EM modeling codes can be very difficult without the aid of a graphical user interface. This paper describes how an inexpensive commercial CAD program for PC's was used to provide a graphical user interface for a finite element modeling code. The same principles could be applied to other CAD programs and the procedure would work equally well with other three-dimensional EM modeling software including moment-method and finite-difference time-domain codes.

## Introduction

Many useful three-dimensional electromagnetic modeling codes are distributed in the form of source code. A single Fortran or C source listing can be compiled to run on a wide range of systems from PCs to supercomputers. Hence, source code is perhaps the most convenient way to distribute a program to a wide variety of users.

Unfortunately graphical input/output procedures tend to vary considerably from one machine to another. Therefore, in order to compile a source code listing correctly on a variety of computers, input and output must generally be restricted to ASCII text. Simple three-dimensional configurations can usually be created using a text editor to build an input data file. Data files for large complex configurations however, can be difficult to build in this way and errors in such a data file are difficult to locate.

In order to combat this problem, many EM modelers have developed graphical front-ends for their source code programs. Graphical representations of the configuration being modeled help the user to build complex, error-free inputs. Graphical front-ends tend to be machine dependent and are rarely distributed with the rest of the code. They are usually complex programs that are difficult to develop, and relatively few non-commercial graphical front-ends are available.

This paper describes how an inexpensive, commercial CAD program (DesignCAD 3-D) is used to create input data files for a non-commercial finite element program. No modifications to the CAD program were necessary. Instead, rules were developed for identifying source parameters, material constants, and other non-geometric parameters within the CAD drawing. A simple utility converts the output from the CAD program into a suitable data input file for the EM modeling program.

### DesignCAD 3-D

DesignCAD 3-D is an inexpensive, relatively easy to learn, three-dimensional drawing program that runs on personal computers. It uses a mouse and/or the keyboard with built-in drawing tools for making lines, arcs, circles, rectangles, boxes, cones, cylinders, spheres, and hemispheres. All objects are drawn in color on a resizable grid. There is one main viewing window and three smaller views that are constantly displayed. The object can be easily rotated or shifted while it is being drawn. Text labels and shading are also features of the program.

### The Translation Code

DesignCAD 3-D saves all of the information describing a particular drawing in an ASCII text file. It was a simple matter to write a short *translation code* that reads the DesignCAD text file and converts it to an input file for our finite element program.

If a CAD drawing is going to be used to provide input to an EM modeling code, all of the necessary input information must be contained in the drawing. The geometry of the configuration is obtained from the dimensions of the drawing. Material properties, source fields, mesh densities, and other information can be stored in the drawing by taking advantage of color, special objects, text, or a combination of these features. The basic information required by our three-dimensional finite element code is

- the location of every mesh node
- the material coefficients at every mesh node
- whether a node is a boundary node and the orientation of the boundary
- the value of any source parameters at the node.

Since our code employs a uniform rectangular grid, it is a simple matter to designate the CAD program's grid points to be the mesh nodes. The *snap-to-grid* feature of the CAD program helps to ensure that nodes lie on material boundaries as much as possible.

A uniform rectangular grid is not a prerequisite for obtaining input from a CAD program. Finite element codes that employ more sophisticated mesh generating algorithms could still find all the input information they need in the CAD drawing. In this case, setting the grid points equal to mesh nodes was a matter of convenience.

Colors are used to define the material coefficients at every mesh node. For example, green is defined by our translation code to represent a lossless dielectric with a relative permittivity, $\varepsilon_r = 3$. Other colors represent other materials with different properties. White is different from other colors in that white objects have perfectly conducting walls and hollow interiors. This is a convenient way of defining the outer boundary of many problems. Although this particular finite element code

doesn't support infinite elements, representing infinite elements by a specific color would be a straight forward extension of the translation code.

Arrows are used to set the electric field at a specific node to a fixed non-zero value. The base of the arrow is located on the node associated with that field value. The direction of the arrow indicates the direction of the field. The magnitude of the field is represented by the color of the arrow. The length of the arrow and the size of the arrow head can be used to convey additional information about the field.

The translation code is relatively simple. The boundary of the configuration is specified in the first line of the CAD program's text file. Beginning with the first node (i.e. grid point), the code sets all electric field values to *free* and the material coefficients to free space values. Then the code checks to see if the node is on the boundary of a white object. If it is, the tangential components of electric field are set to zero. Next, the code checks to see if the node is within or on the boundary of a colored object. If so, the material coefficients are set to the values corresponding to that color. Finally, the code checks for the presence of an arrow at that node. If there is an arrow, the components of electric field corresponding to the arrow's direction are set to a fixed value based on the color of the arrow. The translation code cycles through every node in this manner until every node in the object has been described. The output of the translation code is an ASCII text file that can be read directly by the finite element code.

## Example

The shorted rectangular waveguide illustrated in Figure 1 took only a few minutes to draw using the CAD program. The 4×11×16 node waveguide was drawn in white with the *box* tool. The dielectric block in the middle of the waveguide was drawn using the same tool in green. The arrows

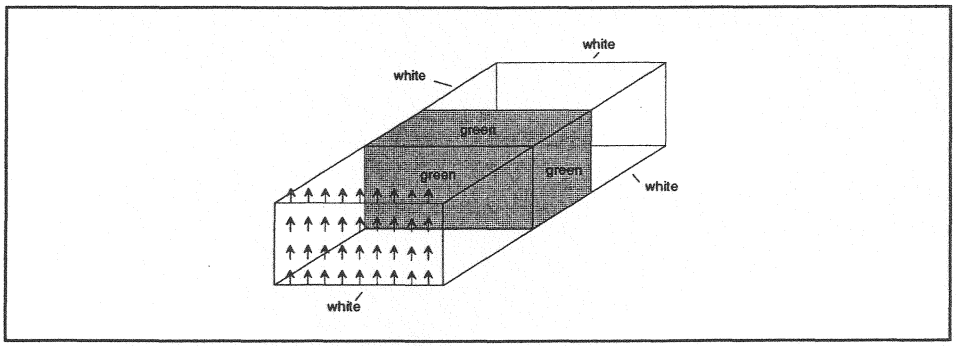


Figure 1: Shorted waveguide with dielectric loading

on the front end represent the source field. In this case, the source was a field consistent with $TE_{10}$ mode propagation. The arrow colors change from red to blue moving from the innermost to outermost positions horizontally.

The text file generated by the CAD program is 203 lines long. There are four types of lines in the file; a header line, command lines, point lines and system parameters. The header line gives the overall dimensions of the drawing in screen units. Command lines are used to describe individual parts of the drawing such as lines, planes, arrows, etc. Point lines locate individual points using screen coordinates. System parameter lines serve a number of purposes the most important of which is to state the user drawing size. The user drawing size is used to convert between screen coordinates and grid coordinates.

The file generated by the translation code for input to the finite element program has a short header plus one line for each of the 704 nodes. Prior to developing the CAD interface, this input was generated by a number of subroutines. These subroutines had to be modified for each new configuration and there was no graphical feedback to alert the code user of an error in the input. The CAD interface has made it considerably easier to model new objects with fewer opportunities for mistakes.

**Summary**

By adopting an inexpensive commercial CAD program as our graphical user interface, we have greatly simplified the process of defining input for a three-dimensional finite element code. Writing a translation code to convert a DesignCAD 3-D output file into a finite element input file was a fairly straight forward task.

A similar procedure could be used to translate DesignCAD 3-D files to input suitable for other three-dimensional electromagnetic modeling codes such as NEC or TSAR. We chose this CAD program because of its price, performance, ease of use, and because it runs on a PC. Any CAD program that creates a readable description file could have been used.

In general, CAD programs are easier to use than code-specific graphical front-ends. CAD programs have a wider user base and there are usually a number to choose from for any given platform. The use of commercial CAD programs saves code developers from having to write their own graphical front-ends. This saves time and allows them to write truly portable codes. It also gives code users a professional graphical interface well suited to their needs and their particular platform.