Modification to the Monte Carlo N-Particle (MCNP) Visual Editor (MCNPVised) to Read in Computer Aided Design (CAD) Files

Final Report
Instrument No: DE-FG02-03ER83844
June 27, 2003 through June 26, 2005

Randolph Schwarz of Visual Editor Consultants
Leland L. Carter of Carter Monte Carlo Analysis, Inc.
Alysia Schwarz of Visual Editor Consultants

Summary of Work

The purpose of this work was to modify the MCNP Visual Editor to include the capability of importing 2D Drawing Interface Format (DXF) files and 3D CAD Standard ACIS Text (SAT) files and converting these files to create an MCNP input file.

As a result of this research, it was determined that CAD can be used to efficiently create an MCNP geometry using the geometry capabilities available in CAD, including the efficient creation of 3D objects, the ability to create copies of CAD bodies, and the 3D display of CAD geometries.

In addition, the conversion program can convert currently existing geometries using the import for CAD solid models. Many CAD geometries created for manufacturing contain complexities that are not important to the MCNP model; these CAD files should be modified to conform to the more restrictive constraints of an MCNP geometry. In particular, all space must be defined and unions and intersections should not be too complicated for efficiency of the Monte Carlo simulation. Once these constraints are satisfied, the conversion from the CAD geometry to the MCNP geometry will be an exact conversion.

The conversion program will convert cones, cylinders, polyhedra, tori, and spheres to the corresponding MCNP geometry. CAD geometries consisting of more than 1,000 bodies have been converted and geometries created by users have been successfully converted to create the corresponding MCNP input file.

It is expected that the CAD conversion program can be effectively used by MCNP users developing new MCNP geometries from scratch and by users who have a currently existing CAD model that was not initially developed for export to MCNP.

Table of Contents

Table of Contents (con't.)

Page

List of Figures

1.0 Abstract

Monte Carlo N-Particle Transport Code (MCNP) (Reference 1) is the code of choice for doing complex neutron/photon/electron transport calculations for the nuclear industry and research institutions. The Visual Editor for Monte Carlo N-Particle (References 2 to 11) is internationally recognized as the best code for visually creating and graphically displaying input files for MCNP. The work performed in this grant was used to enhance the capabilities of the MCNP Visual Editor to allow it to read in both 2D and 3D Computer Aided Design (CAD) files, allowing the user to electronically generate a valid MCNP input geometry.

2.0 Significant Accomplishments

The CAD conversion program is part of the MCNP Visual Editor Code, which is a graphical user interface written in C++ that is linked to a set of FORTRAN routines that contain the MCNP FORTRAN source code. The MCNP FORTRAN has been modified and additional subroutines have been written to enable the FORTRAN to interact with the graphical user interface.

This CAD conversion capability was integrated into the standard Visual Editor Package to be compatible with Version 5 (References 12 to 15) of MCNP. Additional information concerning the Visual Editor and the CAD conversion capability can be obtained by contacting the authors through the development website (www.mcnpvised.com).

The CAD conversion program is designed to read and convert 2D Drawing Interface Format (DXF) CAD files and 3D Standard ACIS Text (SAT) CAD files.

The 3D conversion capability allows for importing 3D CAD geometries that were created in CAD using “perimeter modeling” or “solid modeling”.

“Perimeter modeling” is designed for a geometry that is created with CAD by defining only the perimeter of each body. The conversion then determines each MCNP cell as the outer perimeter along with the algorithm to determine any inner perimeters for that cell. Perimeter modeling requires that the CAD bodies be completely contained inside each other, although they can share one or more common face. With perimeter modeling unions and intersections are not allowed.

“Solid modeling” allows for a CAD model where all space is defined, including void spaces, but doubly defined space is not allowed. This is the type of model that is typically used for manufacturing. With solid modeling a minimum number of subtractions and unions are allowed.

For conversion to MCNP, perimeter modeling is preferred because the CAD geometry is simpler and much more reliable for conversion. This document will focus on perimeter modeling unless otherwise stated.

2.1 Converting 2D CAD Files

The 2D CAD file must be exported from CAD in a DXF format in order to be read into the Visual Editor. The DXF format was used because it is a universal format that can be written and read by most CAD packages.

To implement the 2D CAD conversion, eleven new FORTRAN subroutines have been added. The C++ interface is used to read in the CAD file, convert the CAD surfaces to MCNP surfaces, and send these surfaces to the FORTRAN code, which will then construct cells from these surfaces.

Section 4.0 is a detailed discussion of the 2D CAD conversion. Appendix A provides additional technical details of how the FORTRAN code converts a 2D CAD file.

2.2 Converting 3D CAD Files

The 3D CAD file must be exported from CAD in a SAT format in order to be read into the Visual Editor. The SAT format was used because it is a universal format that can be written and read by most CAD packages.

The SAT format supports five different types of surfaces. MCNP has equivalents for the plane, cone (which includes cylinders), sphere, and torus. The program can successfully convert all these surfaces. SAT also supports a spline surface that is modeled by a third order polynomial (or greater). Because MCNP does not model above a second order polynomial, SAT splines had no direct MCNP equivalent. Figure 2.1 shows examples of some of the converted CAD objects.

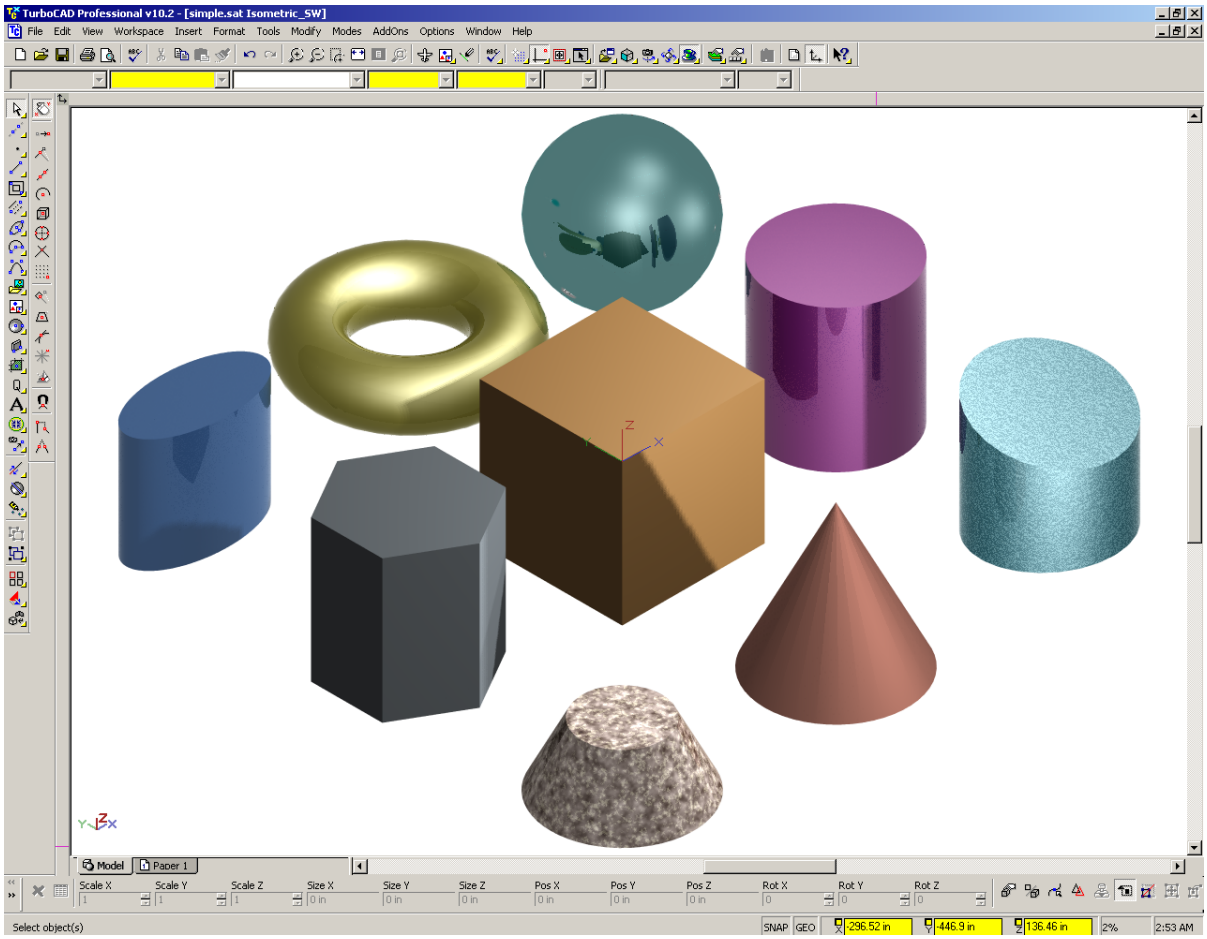

Figure 2.1. Assorted CAD Objects Drawn in a CAD Package

Figure 2.2 shows a 3D display of the CAD objects (shown on the right), after they have been imported into the Visual Editor. These were then converted to MCNP. The 2D MCNP plot of the geometry is shown in the left of the figure. The top of the input file is shown on the bottom left.

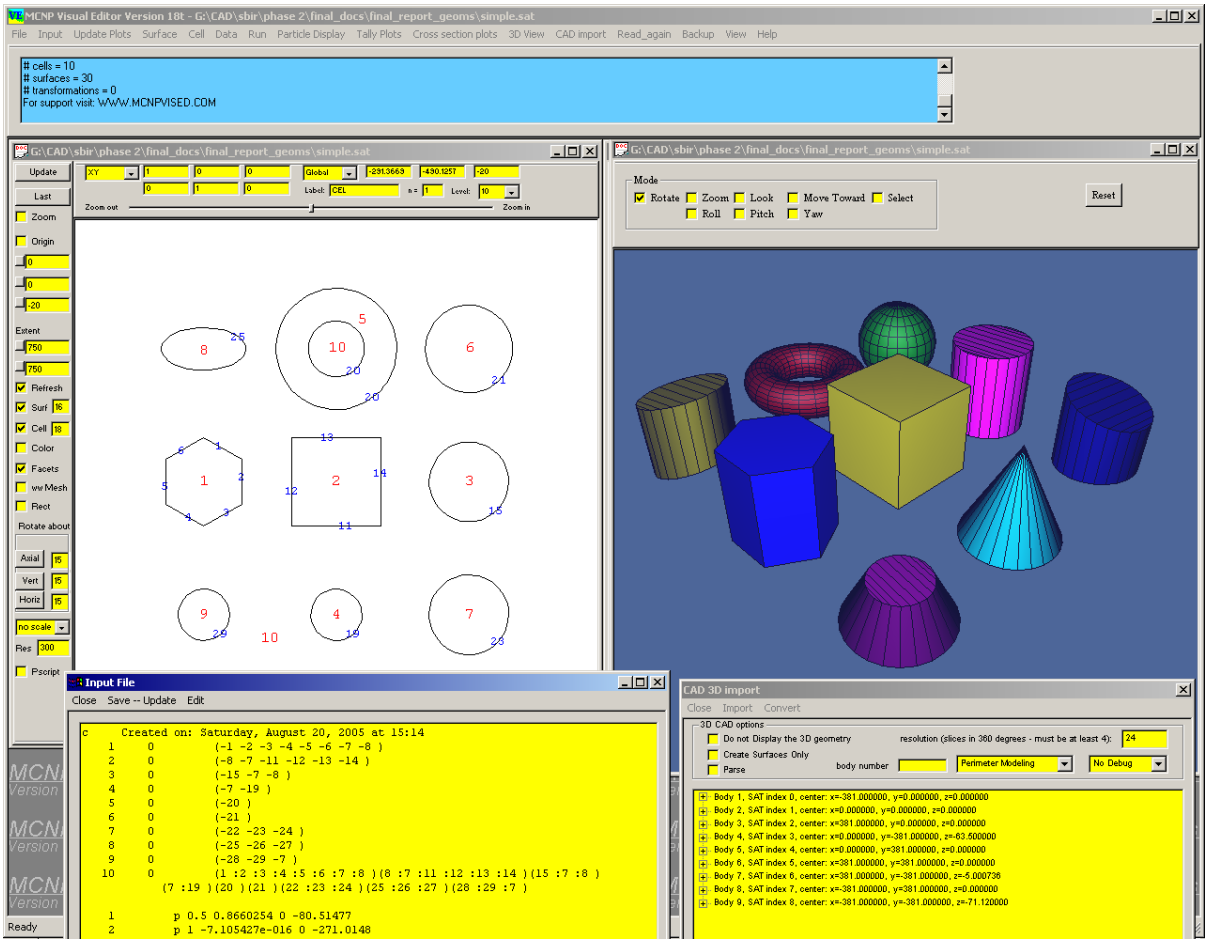

Figure 2.2. CAD Objects in the Visual Editor after Conversion

A paper summarizing this work was presented at the Monte Carlo 2005 meeting in Chattanooga, Tennessee (Reference 16).

A number of features were included in the Visual Editor CAD conversion program to aid in the conversion of 3D CAD files, including the ability to parse a complex object made of a number of unions into simpler objects that are easy to convert and the ability to convert objects with a small number of unions or intersections.

Many files created without prior intent for use with MCNP contain complexities that are not important for MCNP and as such need to be modified to meet the conversion constraints.

Section 5.0 is a detailed discussion of the 3D CAD conversion. Appendix B provides additional technical details of how the FORTRAN code converts a 3D CAD file.

2.3 Constraints/Restrictions for 3D CAD Conversion

Because CAD programs place no real world restrictions on the geometry created, it was necessary to define what constraints must be applied on the CAD software so that the resulting SAT file could be converted to a valid MCNP geometry.

1) Each CAD surface must be expressed as a general (second order) quadratic in x, y, and z. Splines cannot currently be converted.

2) The CAD model must define all of space. This means that regions of air need to be defined as objects so they can be converted to the proper MCNP cell.

3) The CAD geometry should be inside a large box, or cylinder, or sphere, where the region beyond this large box, or cylinder, or sphere in the conversion will be defined as a MCNP cell for the "outside world" cell; i.e., with an importance of zero.

4) A CAD region is limited in its complexity, so that the resulting MCNP cell does not exceed the limits of an MCNP cell. If the cell is too complex, it must be split into simpler cells.

5) For CAD solid modeling, a limited number of unions and intersections are allowed; however, if an object is too complex, it must be split into a number of simpler cells.

The specific 3D modeling constraints depend on what 3D CAD approach is used. See Section B.4.1 in Appendix B for a more detailed discussion of the specific constraints for CAD perimeter modeling and Section B.5 for a more detailed discussion of the specific constraints for CAD solid modeling.

Although these constraints add some additional burden to the CAD designer, it will result in a more efficient MCNP model that is not overly complex.

2.4 Using CAD as a Graphical User Interface for MCNP with Perimeter Modeling

If a CAD file does not currently exist, the conversion program allows for the import of a simplified CAD geometry that can be converted to an MCNP format. The special format defines solids that are entirely contained inside each other or sharing a common face. This algorithm is designed for a geometry that is created with CAD by defining only the perimeter of each body. The conversion then determines each MCNP cell as the outer perimeter along with the algorithm to determine any inner perimeters for that cell.

2.5 3D Display of Imported CAD Files

Once the geometry has been imported, a 3D visualization of the CAD geometry is displayed. The user can use the mouse to move around the geometry and change the visibility of individual cells (hidden, solid, transparent or wireframe). Each cell is displayed in a different color to help differentiate between the different imported bodies.

Figure 2.3 shows an example of the display of imported bodies as read from the SAT file. The top of the building is made transparent, and the pillars and central cone have the wire frame removed so they appear solid. Details concerning the SAT file are shown in the right side of Figure 2.3, where each body displayed is identified. When the user clicks the mouse on a body

in the plot window, the selected object will be identified in the bottom panel on the right. The user can rotate the 3D image and move around the object as desired, using the “rotate” button or the “yaw”, “pitch” and “roll” buttons.

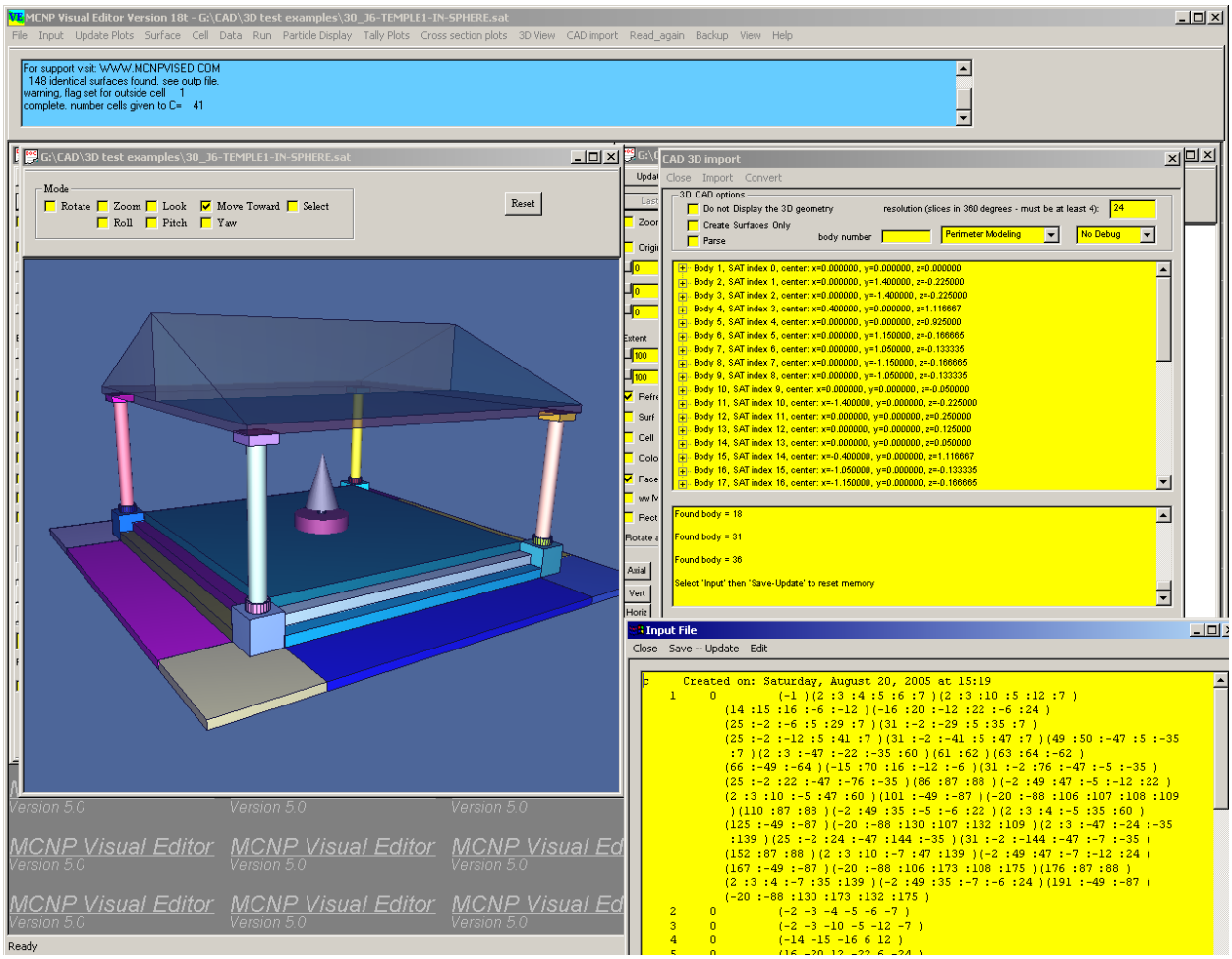

Figure 2.3. 3D CAD Visualization

2.6 Conversion of Large Files

To show that the CAD conversion works for large files, a 1,000-sphere case was created. Each set of 25 spheres was enclosed in a rectangular parallelepiped, to minimize the number of surfaces used in the creation of the cells in the resulting MCNP geometry. Without the parallelepipeds, the resulting air space between the 1,000 spheres would be too complex for MCNP. Figure 2.4 shows the 3D display of the spheres with the boxes hidden on the front, so the spheres can be seen, and some of the boxes set to transparent in the back. The resolution of the spheres has been reduced to enable faster geometry manipulation. The resolution used to generate a curved surface is set with the resolution text box and indicates the number of intervals to use in 360 degrees.

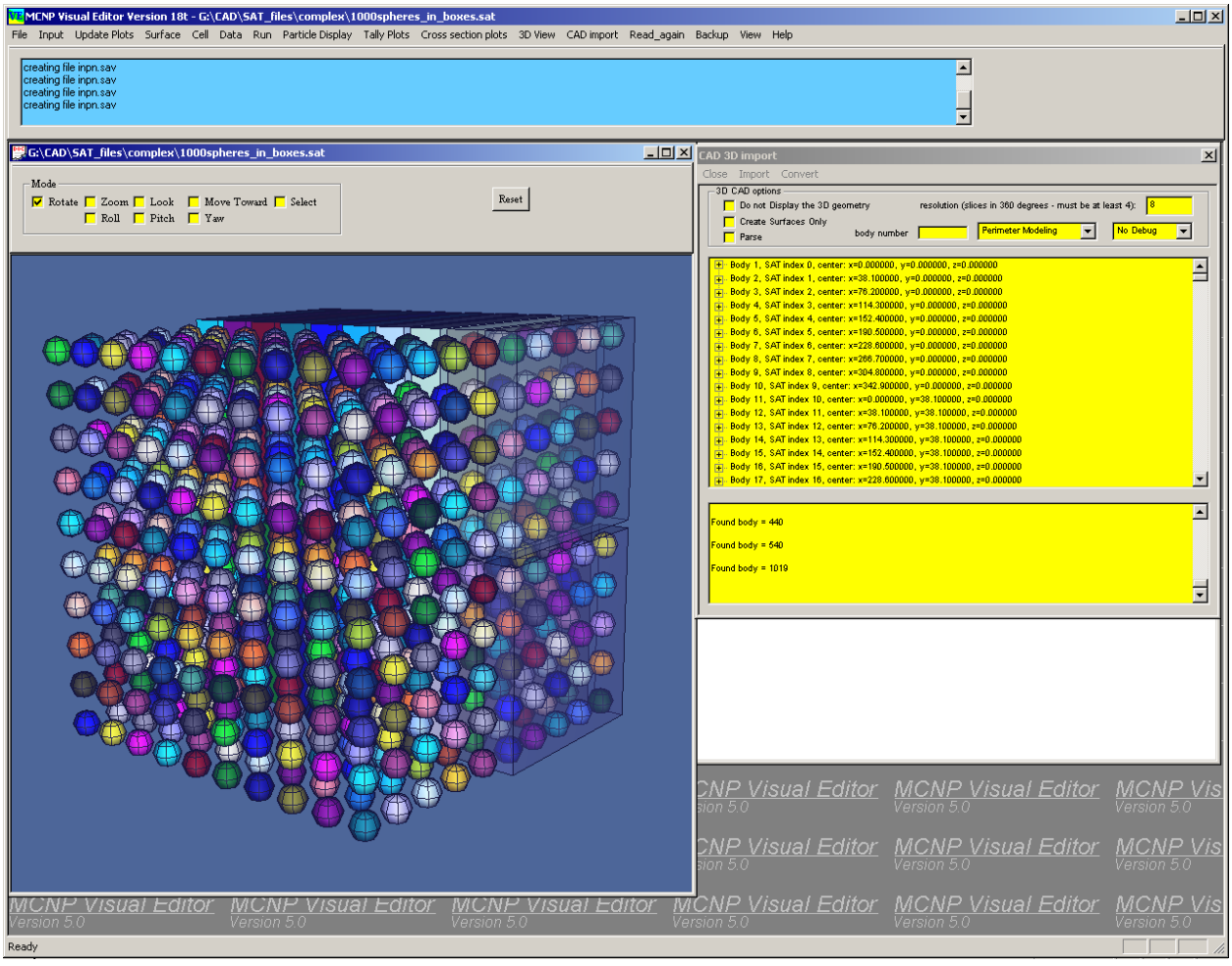

Figure 2.4. 3D Display of 1,000-Sphere Geometry

Figure 2.5 shows the resulting MCNP geometry and input file after the CAD file has been converted to MCNP.

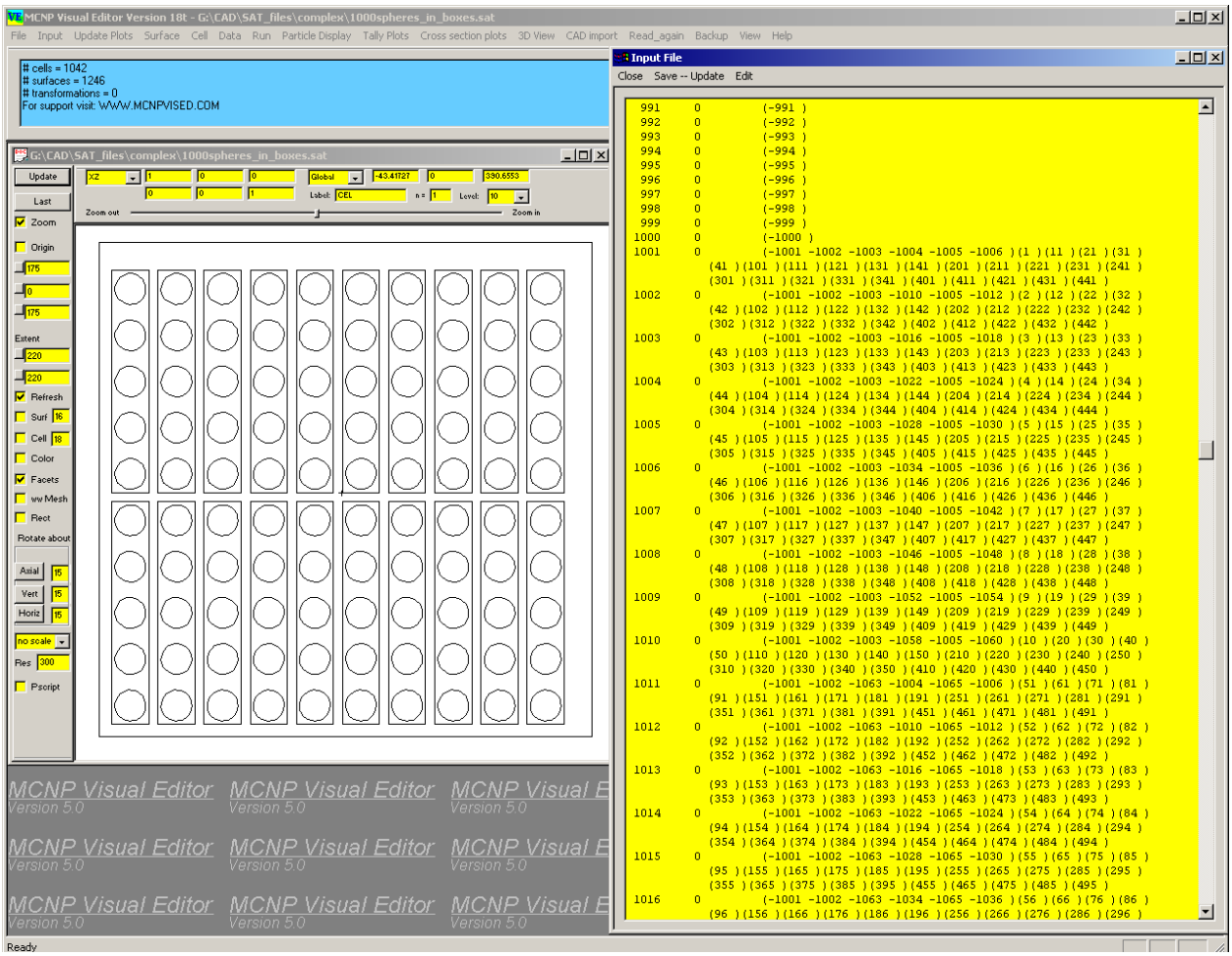

Figure 2.5. MCNP Geometry with 1,000 Spheres

2.7 Conversion of CAD Files from Industry

Input files were solicited from users to determine the types of problems that would be encountered in converting CAD files from industry. Figure 2.6 shows a file that was submitted by George Xu of Rensselaer Polytechnic Institute. The original CAD file consisted of a table as a single object. To convert the file, the object had to be sliced and then parsed using the "Parse" option available in the Visual Editor that will decompose and complex object into a number of smaller simple bodies. In Figure 2.6 the 3D display of the table can be observed along with a 2D plot of the converted MCNP geometry. The top of the input file can also be observed in this figure. Several of the cells have been made transparent to help show the complete geometry.

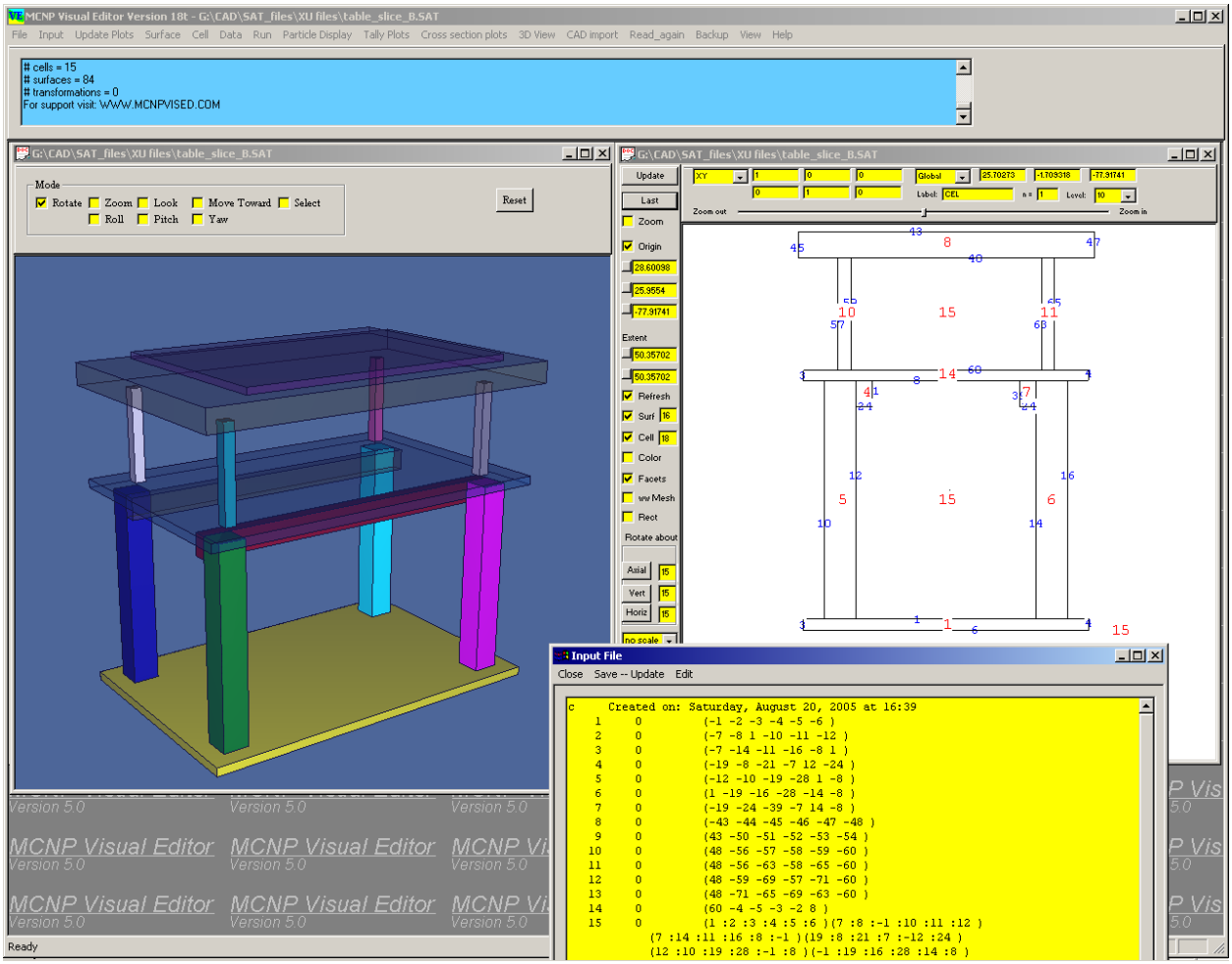

Figure 2.6 Conversion of the Xu Geometry

Figure 2.7 shows a second CAD file submitted by Bud Burns of the Primary Standards Laboratory at Sandia National Laboratories. The model is an aluminum vacuum chamber used to calibrate lead probe neutron activation detectors that measure 14 MeV neutrons. A tritium target is located on the inside of the chamber. A 3D display of the CAD geometry is shown on the left and the converted geometry is shown on the right. The top of the input file is also shown in this figure.

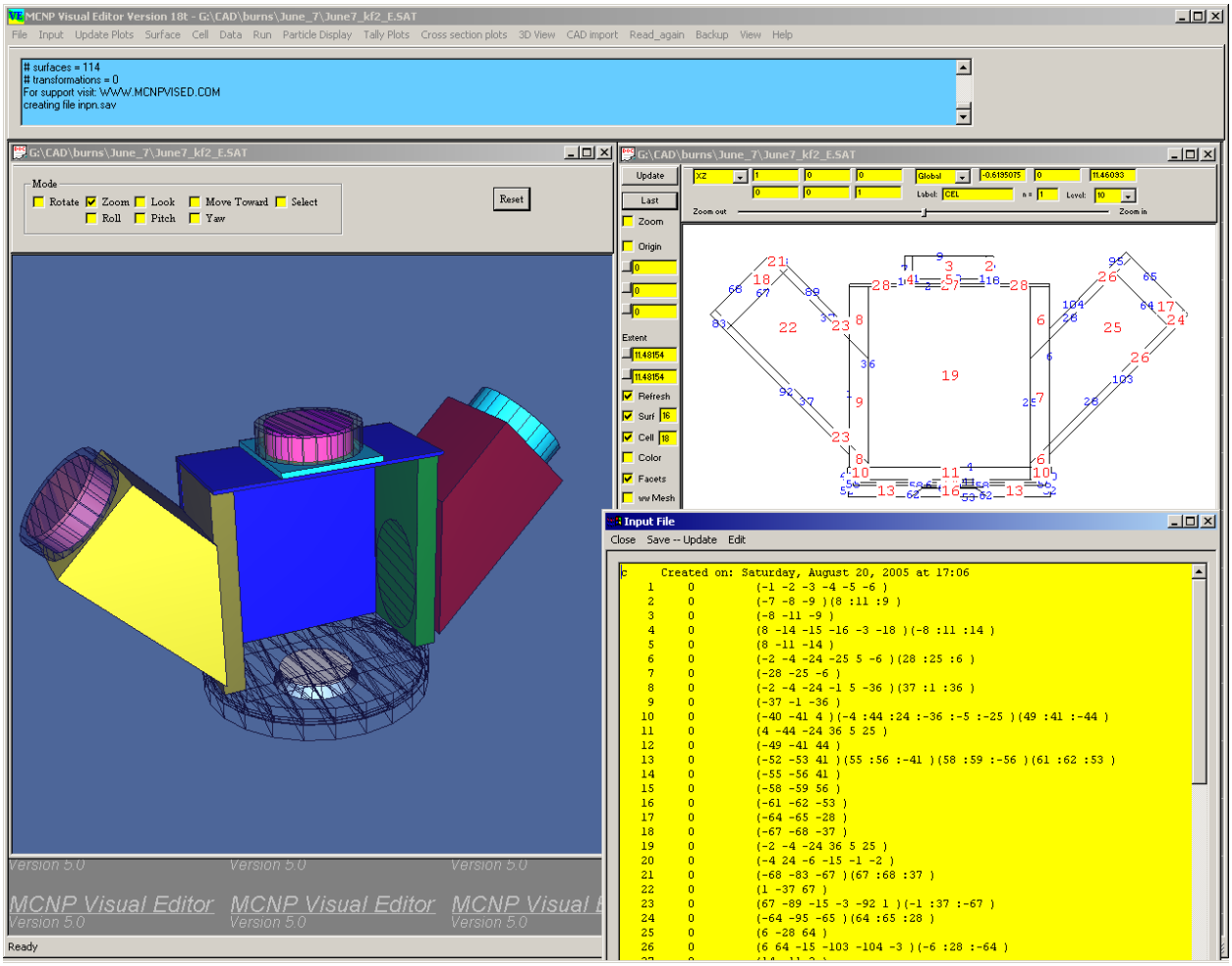

Figure 2.7 Conversion of the Burns Geometry

2.8 3D Display of Resultant MCNP Input Files

The algorithm used to generate the 3D display of the CAD file was adapted to also display the MCNP geometry. Figure 2.8 shows an example of a fairly complex MCNP input file that has been displayed in 3D with all of the cells set to be transparent. This geometry consists of polyhedra, symmetric cells, and asymmetric cells.

It was possible to generate these images by modifying the volume calculator currently in MCNP to provide geometry information for cells where the volume can calculated. This information was then sent to the 3D display routines to generate the 3D object.

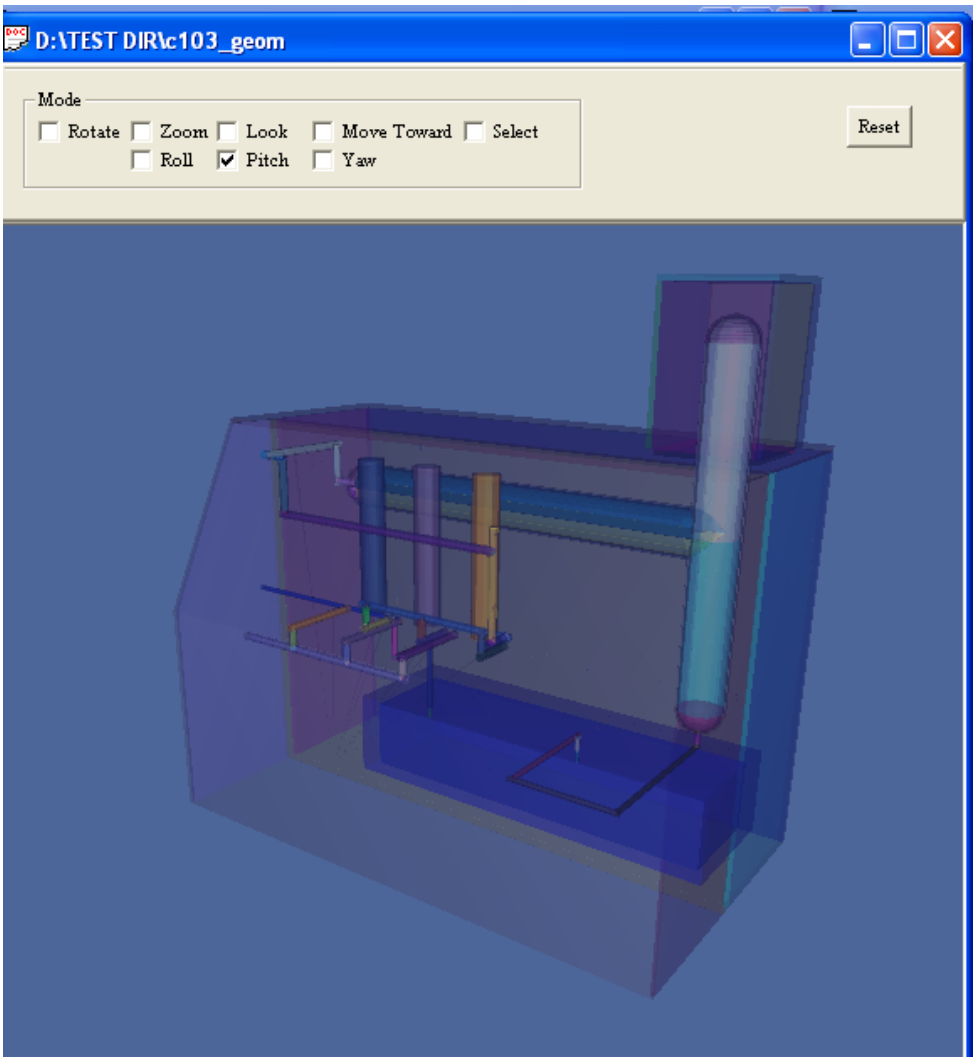

Figure 2.8. Transparent MCNP Geometry

2.9 CAD to MCNP Conversion Class

A CAD to MCNP conversion class was offered in April 2005 and provided an opportunity to work with users in the MCNP community involved in CAD to MCNP conversion work. The class provided valuable input from users and provided an opportunity to obtain existing CAD files that need to be converted.

3.0 Completion Status of Project Objectives

Below is a list of project objectives from the research proposal. Following each of these 10 objectives is the work accomplished to complete the objective.

1) Finish implementing the 2D CAD conversion and test on large realistic examples.

A complete description of the 2D CAD conversion is provided in Section 4.0 and Appendix A. The conversion algorithm was tested on a number of 2D geometries that included lines, circles, polygons, and ellipses and successfully converted to an MCNP geometry format.

2) Determine how to read 3D CAD surface descriptions.

The Visual Editor was used to read in the complete SAT file description. The geometry information contained in the SAT file was then placed in a structure that could be accessed by the code used to generate the 3D geometry display. This information is also sent on to the FORTRAN code to be used to create the MCNP cells.

The CAD geometry is read in when the user selects “import” from the 3D import panel, which can be accessed by selecting “CAD import->3D import” from the main menu. The C interface will bring up a file selection box. After the user has selected a valid SAT file, the C code will read through the SAT file and load the information into a linked-list structure. The structure consists of one or more “body” structures with sub-structures defining “lumps”, “shells”, “faces”, “loops”, “coedges”, “edges”, “vertices”, and “points”. Currently, “wires” and more esoteric “attribute” settings are ignored. A direction is stored on the “face”, “coedge”, and “edge” level of the structure and the points are reordered.

Figure 3.1 shows an example SAT file consisting of an array of 125 cubes. A 3D display of the geometry is shown in the left side of the figure and a detailed description of the SAT file as read in by the conversion code is shown in the window on the right side of Figure 3.1. The user can see the complete geometry structure by accessing the different levels of the geometry displayed in the tree shown in the figure.

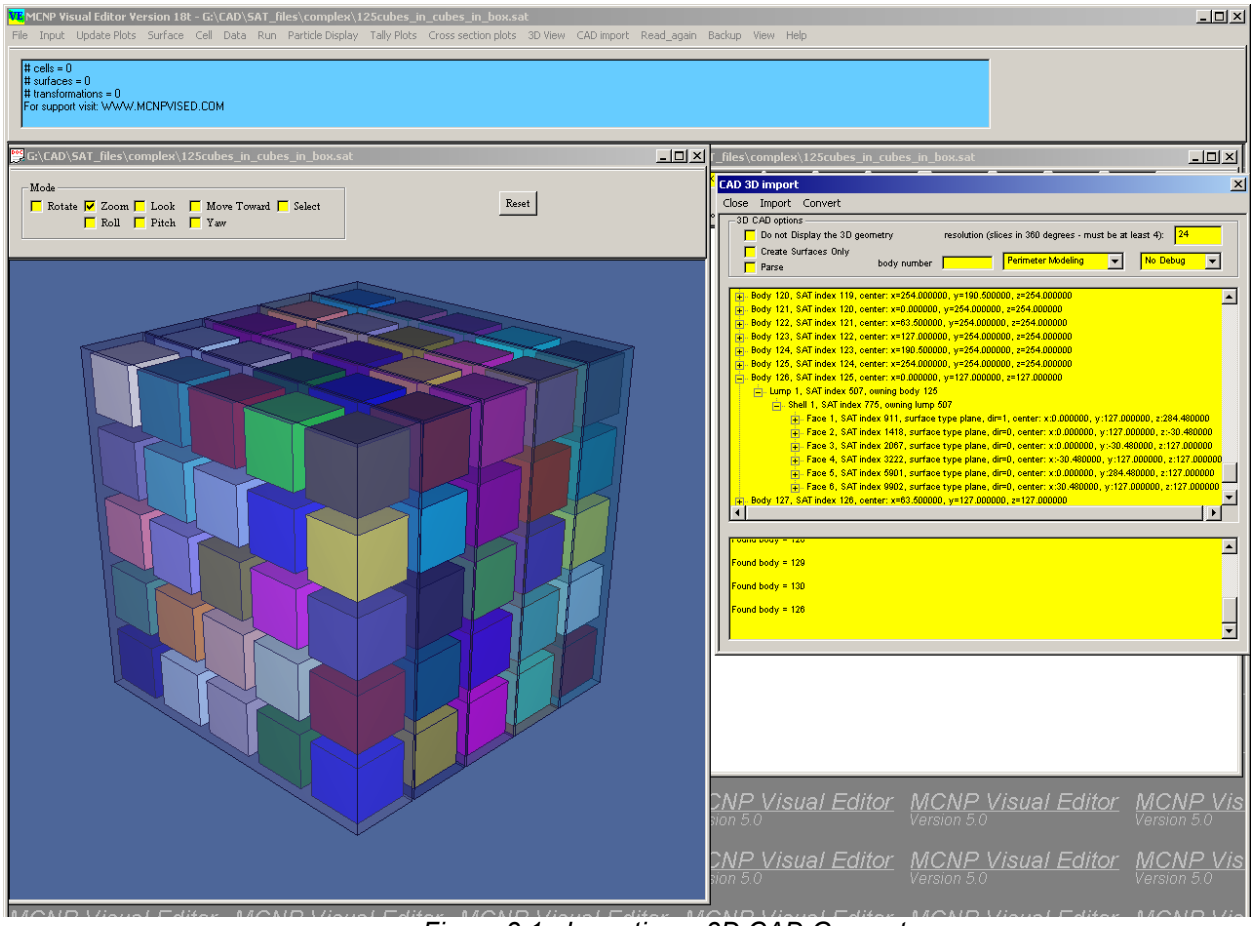

Figure 3.1. Importing a 3D CAD Geometry

3) Determine how to generate MCNP surfaces from 3D CAD descriptions.

Once the SAT file has been read in, the geometry information is passed on to the FORTRAN code. The C++ code calls the "CADCEL3" routine to tell FORTRAN that C now has all the information. The FORTRAN code then calls the CCADSUR3 function to request the surface information from the C++ code. The FORTRAN then takes this information and generates the MCNP surfaces. A detailed description of this function is included in Appendix B.

This information is then used by the FORTRAN code to generate the MCNP surfaces corresponding to the SAT file description. The user can optionally tell the FORTRAN to only generate the MCNP surfaces and to not do the conversion of the cells by selecting the "Create Surfaces Only" on the 3D import panel.

4) Determine how to generate cells from the 3D surface information.

The FORTRAN code generates the cell information from the surface information. A detailed description of how this is accomplished is presented in Appendix B. The code has been tested in converting CAD geometries that result in a number of diverse and complex MCNP cells.

5) Develop a 2D CAD plotting program that can show 2D views of a 3D geometry.

The purpose of the 2D plotting program was to allow the user to interrogate the CAD geometry. Instead of implementing the 2D display, this was accomplished by enhancing the 3D display to allow the user to click on any object in the display and the number of the selected object will be displayed. With this number, the user can use the geometry tree created when the file is read in to determine the specific properties for the object that is selected.

For example, Figure 3.2 shows a drawing of a building with a desk inside of it. The top of the desk has been selected and made solid while the rest of the geometry is transparent. The window to the right of the geometry indicates body 27 has been selected. It is now possible to look at the geometry tree to display the complete information about body 27 obtained from the SAT file.

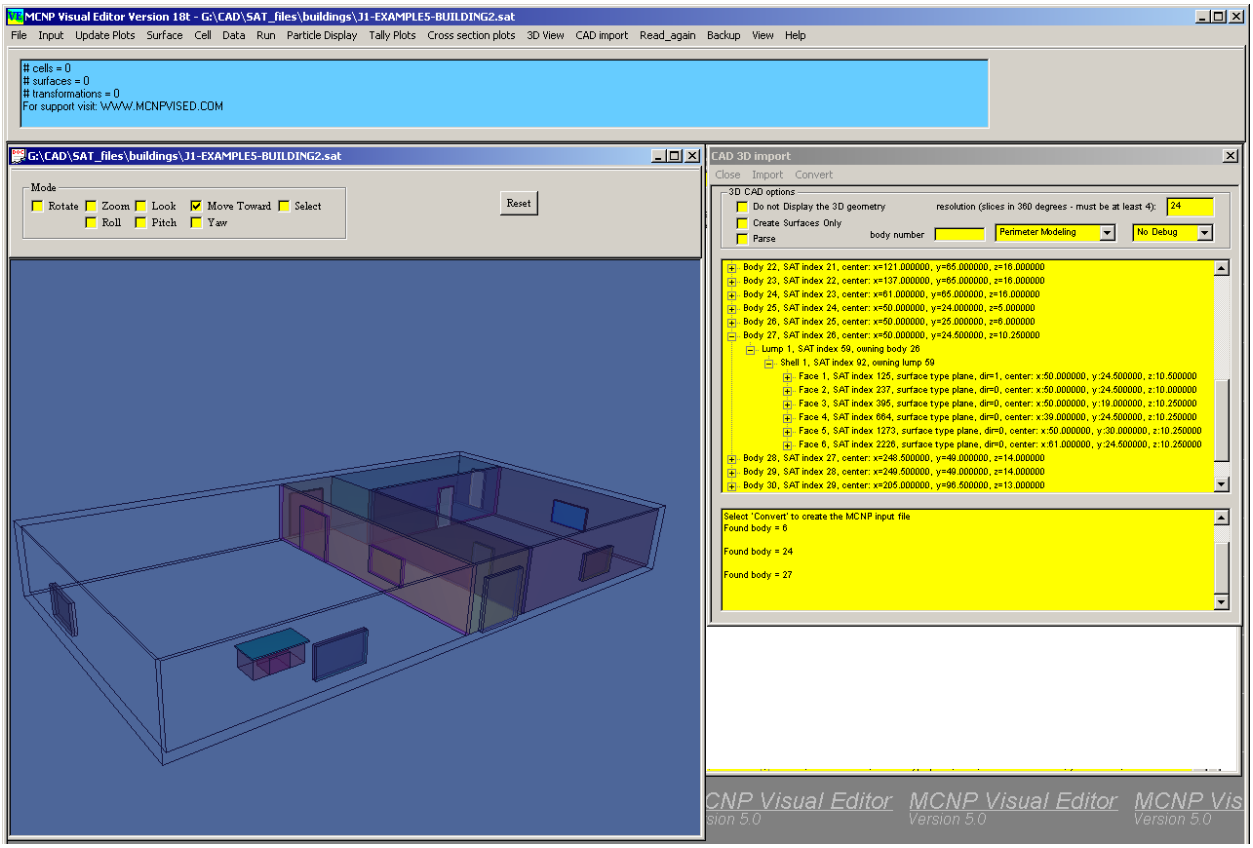

Figure 3.2. Getting Information about the CAD Objects

6) Develop the code to view the CAD geometry in 3D using polygons.

When the SAT geometry is read in it is displayed using a dynamic 3D geometry display. The geometry can be rotated and the viewpoint can be changed to move the view through the geometry. The transparency of the objects can be changed to either solid, wireframe, or transparent. In addition, individual objects can be removed from the display to allow the user to display only the geometry that is important.

Figure 3.3 shows an office after it has been read in from the SAT file. All of the objects are solid. In Figure 3.4 the objects have been made transparent, and in Figure 3.5 the geometry has been displayed in wireframe. Figure 3.6 shows how these different display options can be used to show a realistic view of the office.

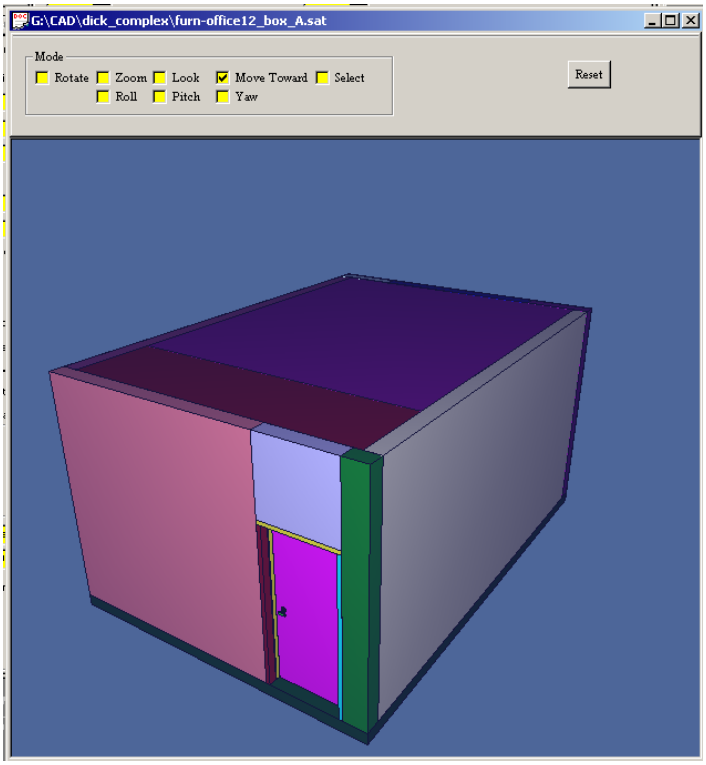

Figure 3.3. Solid Office

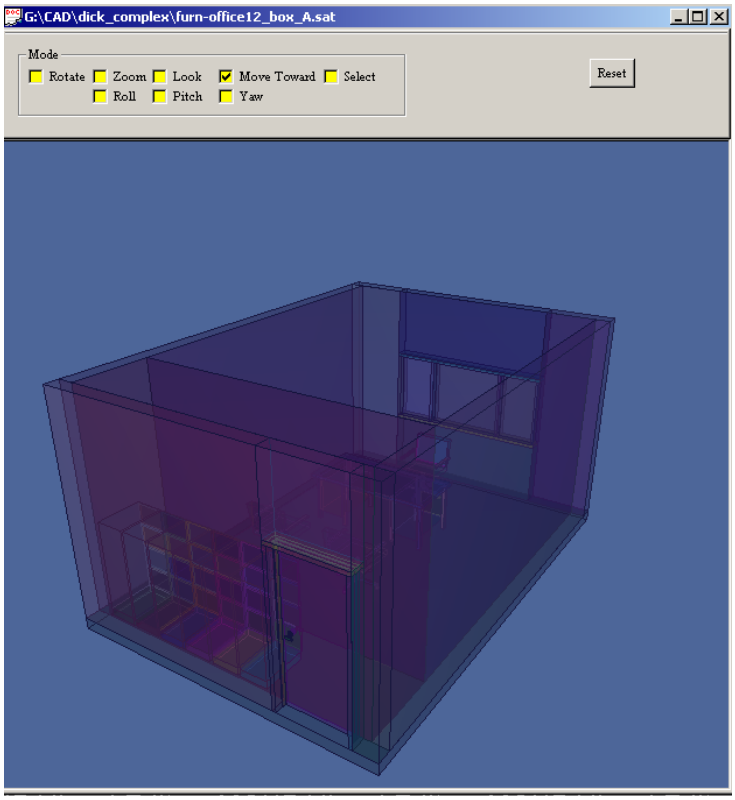

Figure 3.4. Transparent Office

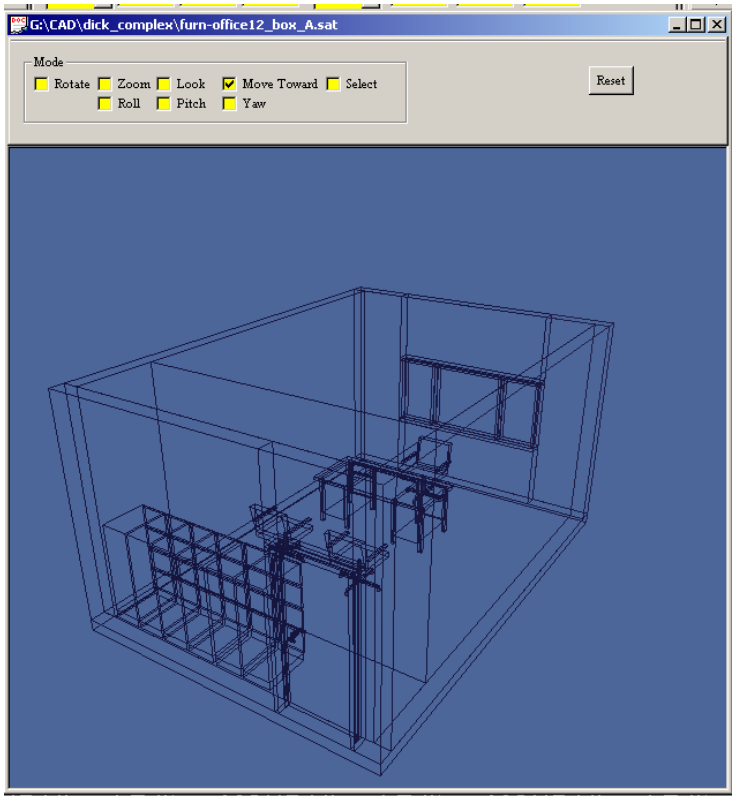

Figure 3.5. Wireframe Office

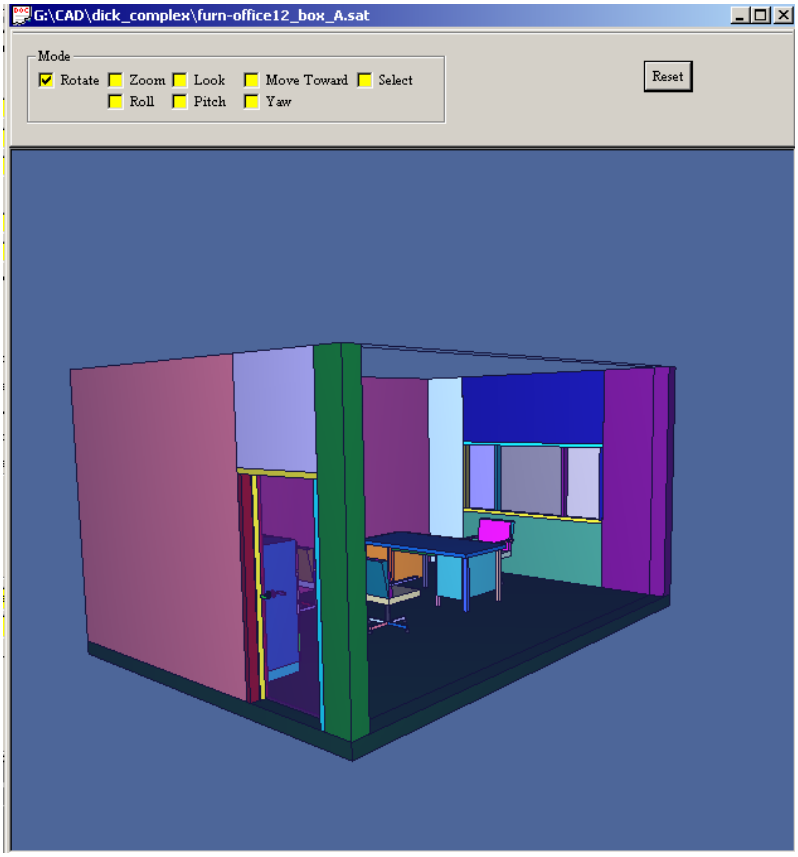

Figure 3.6. Mixed Display Office

7) Determine how to import and export advanced geometries.

A diverse set of CAD geometries were imported into the Visual Editor, including geometries contributed from the user community. However, no export capability, other than the generation of the MCNP input file, is available with this conversion tool. It was thought that an MCNP to CAD conversion should be developed, but this turned out to be beyond the scope of this current work.

8) Debug the enhanced 3D Visual Editor Code.

A number of test problems were developed to verify the diversity of the conversion code. Test problems were generated to test the importing of plane surface, cones, cylinders, spheres, and torus. These basic objects were then transformed and cut with planes to verify that the algorithm can handle a wide variety of objects.

Additional inputs were also solicited from the user community to test and debug the code.

9) Document the resulting enhanced visual editor code.

In addition to this document, the user manual for the Visual Editor was updated to describe how to convert 2D and 3D CAD geometries. Information has also been added to the Visual Editor web site for public viewing (www.mcnpvised.com).

10) Offer training classes in the CAD conversion code, with proceeds to supplement this grant.

A training class was offered in April 2005.

4.0 Conversion of a 2D CAD File

4.1 Importing the 2D CAD File

An algorithm was created to read in a CAD DXF file using a dialog added to the MCNP Visual Editor. To access the panel for importing a 2D CAD file, the user needs to select CAD import ->2D Import from the main menu. This will bring up the "CAD Import" panel shown in Figure 4.1. The user can then select "Import" to read in a CAD DXF file. The CAD geometry can then be displayed prior to converting it to an MCNP format by selecting the "Update" button for one of the plots.

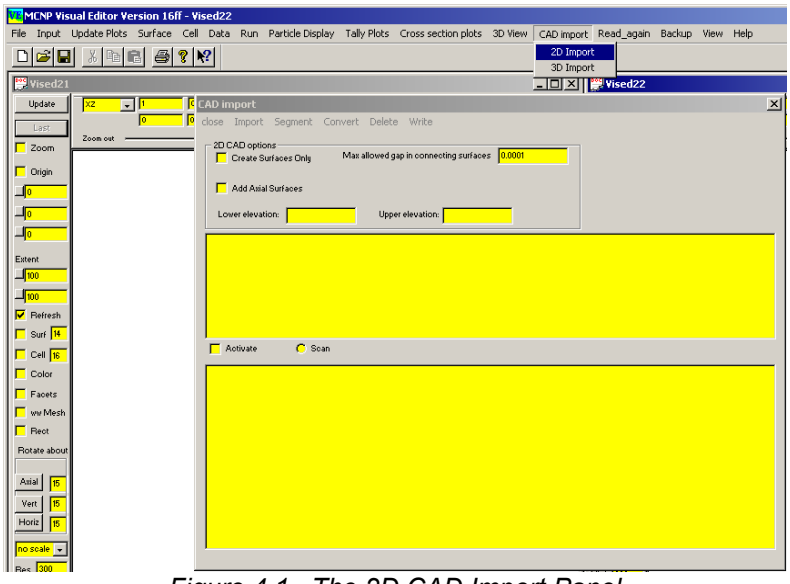

Figure 4.1. The 2D CAD Import Panel

When the user selects the "import" command the C interface will bring up a file selection box. After the user has selected a valid DXF file, the C code will read through the DXF file and search for the entities section, which contains the geometry information and will also read in and store information concerning "blocks" that exist in the file. A "block" represents a piece of

geometry that can be “inserted” into the geometry in multiple locations. By reading these blocks in, they will be available when they are needed for inserting into the geometry with an “insert” entity.

The following entities are currently recognized: ARC, CIRCLE, ELLIPSE, INSERT, LINE, LWPOLYLINE, MLINE, POLYLINE, SOLID. A warning message will be printed if other entities are encountered. Once an entity is found, the information for that entity is stored in memory.

For curved surfaces (ARC, CIRCLE, ELLIPSE) the center of the object, along with the radius and the starting and ending point of the curved surface, is placed in memory. For an ellipse, additional information concerning the major and minor axis is also read in and stored in memory. This information is complete enough to allow the C code to draw the objects after the DXF file has been read.

For line objects (LINE, LWPOLYLINE, MLINE, POLYLINE), information about the starting and ending point of each line segment is stored in memory using a linked list. This information is then used to draw a plot of the line objects after the DXF file has been read.

4.2 Segmenting the 2D CAD File

For every line that crosses another line, the Visual Editor will segment the lines. This is necessary to prevent multiply defined spaces in the MCNP geometry. It also allows the user to remove a line segment using the “scan” and “delete” options on the “CAD import” panel. To segment the lines, the user selects “Segment” from the “CAD Import” panel. Figure 4.2 shows a plot of the surfaces before and after segmenting. Notice that both the lines and circles are segmented. The geometry has still not been converted to an MCNP format. At this point the user could choose to delete a segment by scanning the segment in and then choosing the “delete” option.

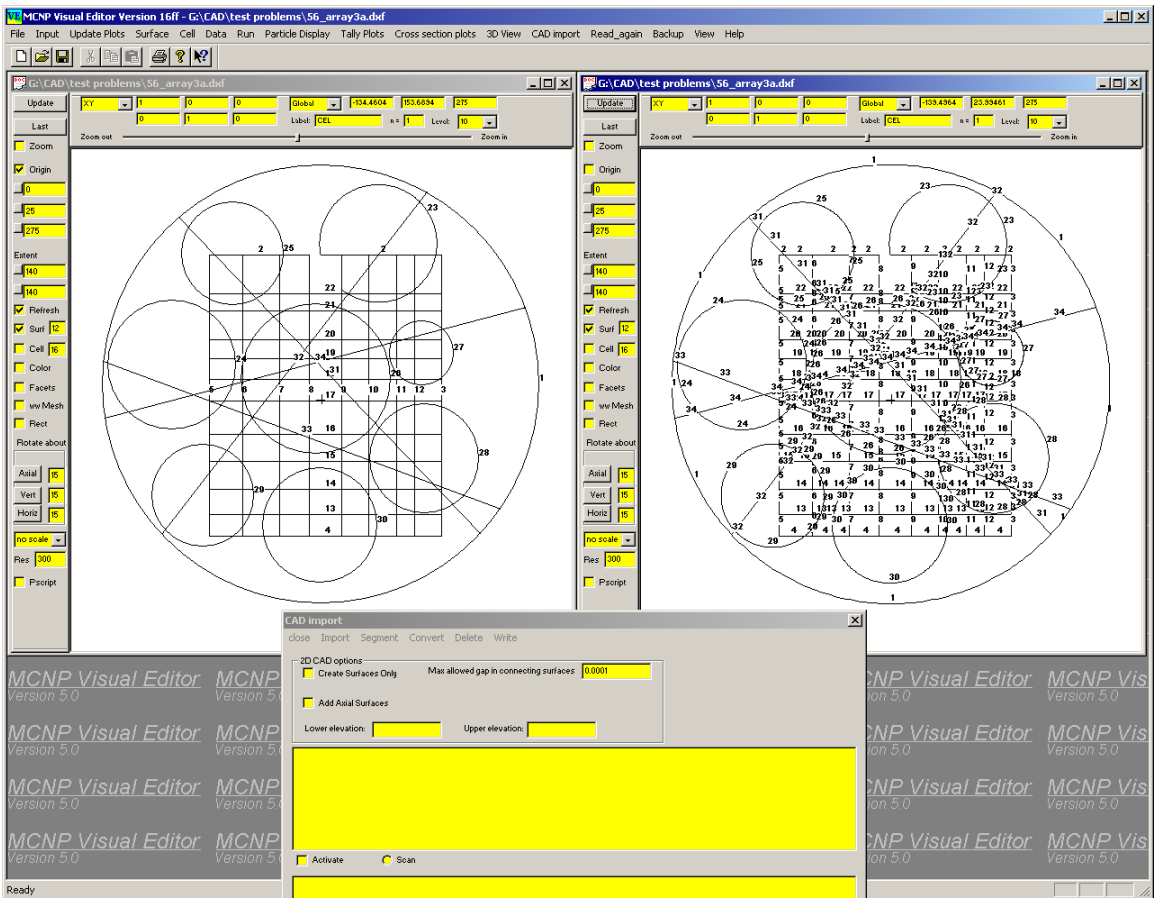

Figure 4.2. Segmenting the CAD Surfaces

To segment the geometry, each entity is looked at and compared to every other entity in the file to see if an intersection exists. For a line crossing a line, there is a simple test to determine if an intersection exists. For lines intersecting a curved surface, a more complex test is needed to look at both possible intersection locations on the curved surface and then determine if the line crosses either one of these locations. The current code does not yet handle the intersection of two curved surfaces.

When an intersection is found, the original entity is modified to end at the intersection point and a new entity is created that extends from the intersection point to the end of the original entity.

For files that contain a large number of entities, this process can be computationally intensive. It is important to note that in CAD, lines do not always meet exactly, so there is an entry in the 2D conversion panel that specifies the "Max allowed gap in connecting surfaces". This will specify how close two line segments need to be in order for the code to determine that the segments connect. The user can modify this value if needed.

4.3 Deleting Line Segments

After reading in the file, the user can choose to remove lines from the geometry by selecting "scan", dragging across the surface, and then selecting "delete." After segmenting the file, the

user can opt to remove the individual segments. This provides the user the ability to modify the file before converting it to an MCNP input file.

4.4 Converting the 2D CAD File

Once the file has been modified by deleting unwanted segments, it can be converted by selecting the "convert" menu option shown in the 2D conversion panel. This will instruct the C code to create the MCNP surfaces and then send the surface information to the FORTRAN code by calling the "cadnsur" FORTRAN subroutine for each surface. For each line, the segments that make up the line are sent to the FORTRAN by sending the beginning and ending point for each line segment. For each curved line, the segments that make up the curved line are also sent to the FORTRAN by sending the beginning and end point of each segment of the curved surface.

The FORTRAN occasionally creates additional surfaces to generate the MCNP cells. This additional surface information is sent by the FORTRAN to the C code.

Once the surfaces have been created, the C code calls the FORTRAN "cadcel" subroutine to create the MCNP cells. The user can optionally specify an upper and lower axial extent to bound the 2D geometry when calling the "cadcel" subroutine.

Once the conversion has been completed, the user should open the "input" window and select the "save-update" option to reset all of the FORTRAN memory and update the plots.

To convert the file, the user selects the "Convert" option to create the MCNP surfaces and MCNP cells. It is not necessary to segment the CAD file before converting, if the user has not yet selected the segment option, the code will automatically detect this and do the segmenting prior to converting the file. Once the file has been converted to MCNP, the user should then select "Input" from the main Visual Editor menu and perform a "Save-Update" in the resulting "Input File" panel to display the MCNP plots.

This conversion works for most of the CAD geometric entities including, lines, polylines, multilines, circles, arcs, and ellipses and also works for the insertion of blocks. These geometric entities include most of the 2D geometries that can be created by CAD. The Visual Editor will display these geometric entities and allow the user to select any of these items and remove them from the geometry (by scanning them and selecting the "Delete" button) before converting them to MCNP. This can be done either before or after segmenting the surfaces. The Visual Editor will also allow the insertion of an upper and lower surface to bound the 2D geometry in the axial direction.

This CAD conversion process has been tested on complex 2D CAD files. Figure 4.3 shows an example 2D CAD file that has been converted to MCNP. The original CAD file is shown in the left plot window and displayed using the new Visual Editor CAD plotting capabilities. The converted MCNP file is shown in the right plot window. The original CAD file contains lines, polylines, polygons, multilines, and circles. The resulting MCNP geometry has 88 surfaces and 31 cells. The first few lines of the resulting MCNP input file can be observed in the input window at the bottom of the figure.

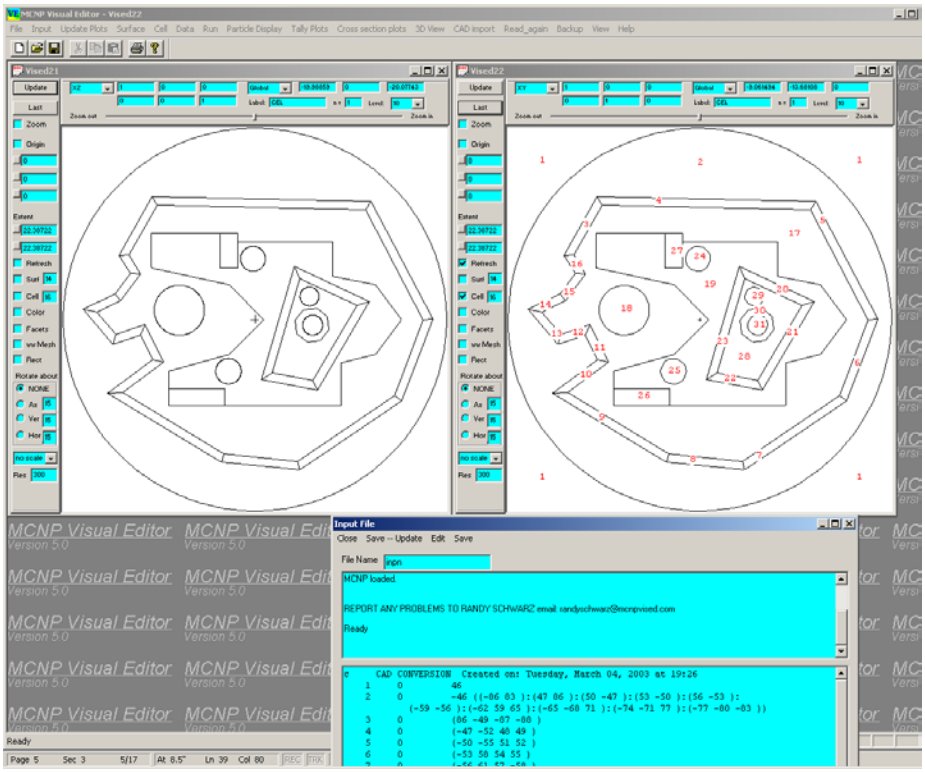

Figure 4.3. CAD Geometry Before and After Conversion

One important problem that has been addressed is the accurate conversion of arcs. The vector normal to the surface changes along a curved surface segment, whereas it is constant along a plane surface segment, so this complication must be included in determining each perimeter of the cell when curved surfaces are involved. An MCNP curved surface is a quadratic and so has two roots for crossings of this surface, for a given starting point and direction-of-flight, instead of only one root for a plane. This introduces the possibility of needing an ambiguity plane to exclude regions beyond the curved surface. For a complicated cell it is often necessary to describe the inside of each arc with a number of straight-line segments in order to be able to create the inner portion of the MCNP cell with the same powerful algorithm that is used when only planes are involved.

Figure 4.4 shows an example of a complex 2D file that can be converted with these upgrades. This figure shows an example of objects that are divided into multiple regions. The sectioned ellipses are particularly difficult to convert.

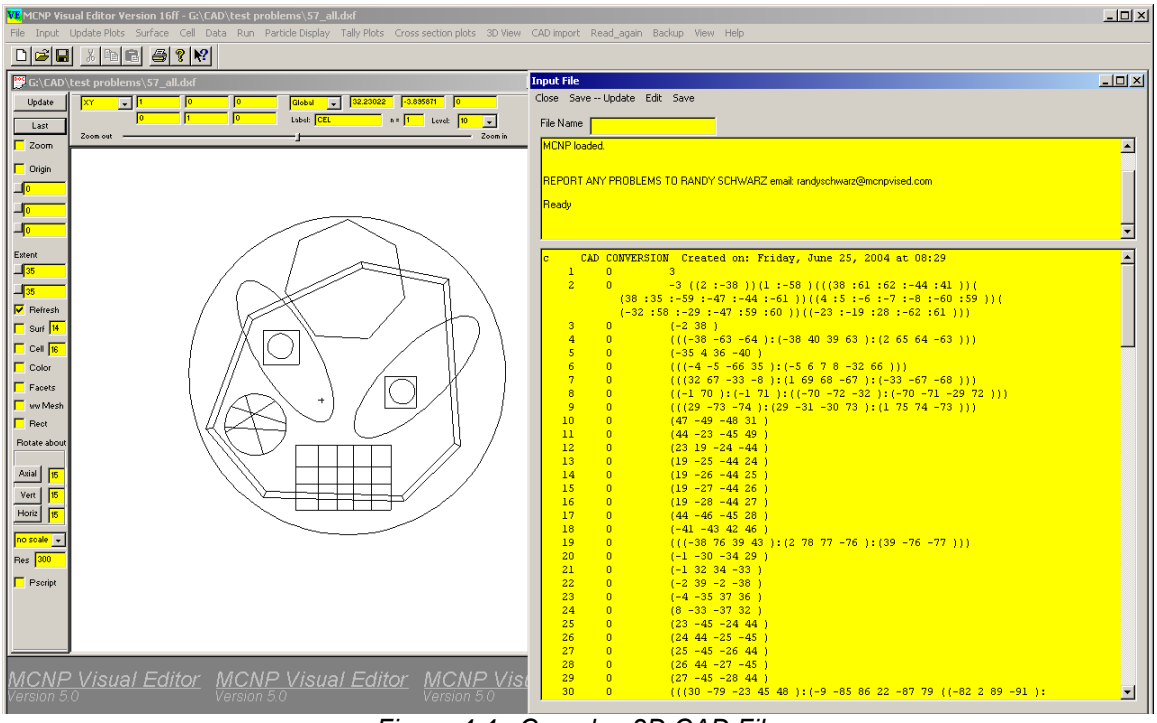

Figure 4.4. Complex 2D CAD File

The CAD DXF file should conform to the usual type of things required of an MCNP input file. If the CAD-to-MCNP conversion does not work, it may be because the CAD regions are too complex to be converted to a MCNP cell. A MCNP cell is limited in the number of surfaces allowed to define the cell. If this is a problem, the user should try simplifying the CAD geometry by creating more regions by segmenting complex regions with lines to create a number of less complex regions (fewer surfaces per region). In addition, the "outside world" beyond the outer perimeter should be bounded on the inside by a circle or a rectangle, or a boundary that involves all unions (angles facing the outside world greater than 180 degrees) to satisfy the usual MCNP "outside world" boundary condition.

5.0 Conversion of a 3D CAD File

To implement the 3D CAD conversion, 12 new FORTRAN subroutines have been added. The C++ interface is used to read in the CAD file, convert the CAD surfaces to MCNP surfaces, and send these surfaces to the FORTRAN code, which will then construct cells from these surfaces.

Appendix B provides additional technical details of how the FORTRAN code converts a 3D CAD file.

5.1 Importing the 3D CAD File

The 3D conversion reads a SAT (Standard ACIS text) file, which can be exported by most CAD packages. The SAT format was used because it is easier to interpret than the STEP format and also converters exist to convert a STEP file to a SAT file.

To access the panel for importing a 3D CAD file, the user needs to select CAD import ->3D Import from the main menu. This will bring up the "CAD 3D Import" panel shown in Figure 5.1. The user can then select "Import" to read in a CAD SAT file.

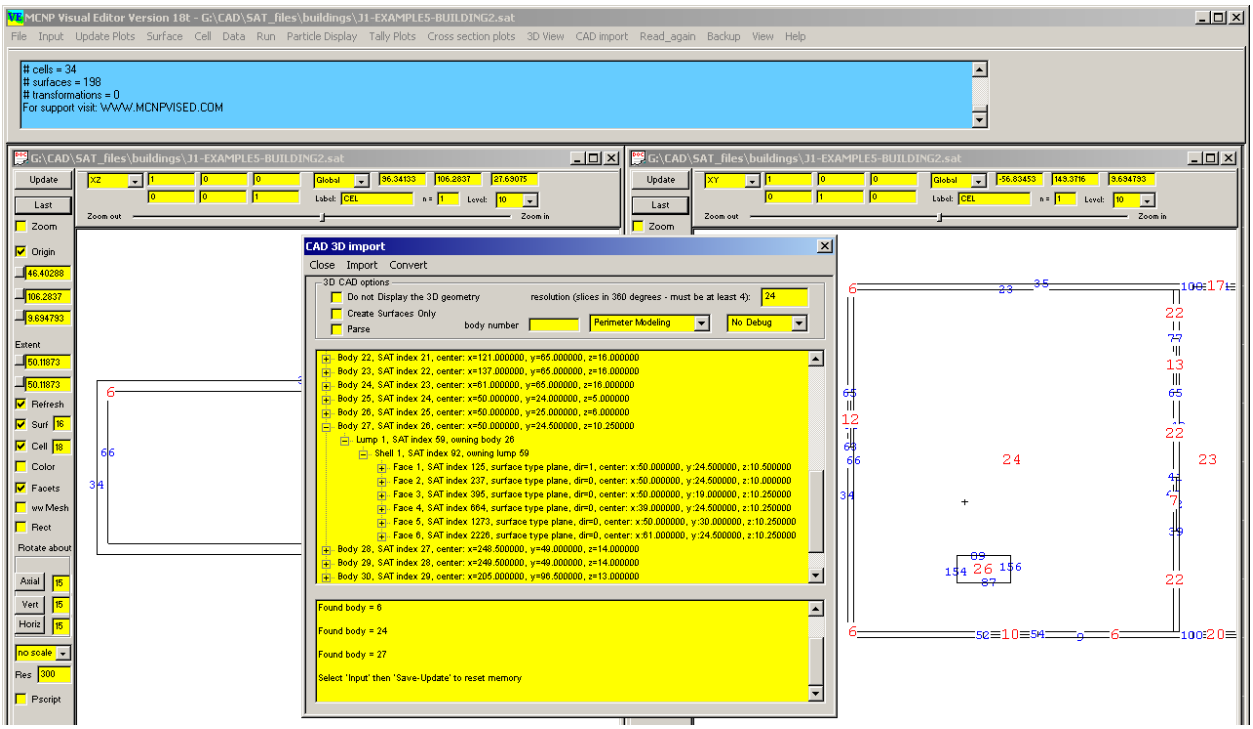

Figure 5.1. The 3D CAD Import Panel

When the user selects the "import" command the C interface will bring up a file selection box. After the user has selected a valid SAT file, the C code will read through the SAT file and load the information into a linked-list structure. The structure consists of one or more "body" structures with sub-structures defining "lumps", "shells", "faces", "loops", "coedges", "edges", "vertices", and "points". Currently, "wires" and more esoteric "attribute" settings are ignored. A direction is stored on the "face", "coedge", and "edge" level of the structure and the points are reordered before the information is sent to the FORTRAN or the visualization code such that all edges are traced in order for each face.

The units that the file was written in are read and the geometry is converted to centimeters to be compatible with MCNP. In addition, the transformation for each body is also read in. If a transformation exists, the body is appropriately transformed.

After the file is imported, the information is loaded into a treeview control to allow the user to view the information about the different bodies, faces, and points by expanding and collapsing the tree. In addition, a variety of diagnostic information is printed in the lower box to help the user with tracking the conversion. The quantity of debug information that is printed out is controlled by the debug level set in the debug combo box.

5.2 Displaying the 3D CAD File

For the 3D visualization, the geometry information is passed on to the 3D display functions. If the object is a plane, a list of vertices is obtained, along with the sequence in which they are connected. This information is used to create the plane. For spheres, cylinders and cones, the center of the object and the radius of the object are used to generate the 3D display. This information is placed in a display list and then shown to the user. The user can control the number of intervals to segment the curved surface by indicating the number of slices in a circle (360 degrees).

A number of options are available in the Visual Editor for viewing the 3D geometry. Individual bodies can be made solid, transparent, or wireframe, or can be removed completely from the display.

Additional options are available to allow the user to change the view by using the standard "roll", "pitch" and "yaw" features common to many flight simulators. The graphical interface also includes a trackball feature to rotate the geometry with the mouse.

5.3 Constraints on the 3D CAD File

It is necessary to make some observations about the geometry to be imported because the CAD program has fewer restrictions on the geometry than the MCNP program. The most significant of these restrictions is the requirement that all space be defined in the MCNP geometry. To deal with this, there are two different algorithms to convert CAD files.

The first type of algorithm consists of solids primitives (sphere, cylinder, box, cone, wedge, torus, or surfaces that define the outer perimeter [CAD shell] of the body). In the CAD file, these must either be completely inside or completely outside of each other, although they can have a common surface boundary. This is the type of geometry that would be created if the CAD program is being used as a front end to generate the MCNP geometry. The Visual Editor will then do the needed subtractions or unions in creating the MCNP geometry.

The second type of file consists of a fully defined solid geometry model where all space is defined. The CAD files of this type were often developed to be used with manufacturing or simulation packages and generally meet all or most of the MCNP constraints because the programs they were developed for have similar restrictions. Individual MCNP cells (initial CAD bodies) should be relatively simple to enable rapid random walk tracking.

See Section B.4.1 in Appendix B for a more detailed discussion of the specific constraints for CAD perimeter modeling and Section B.5 for a more detailed discussion of the specific constraints for CAD solid modeling.

5.4 Example Conversion Problem

Figure 5.2 shows an example of a simple radiation facility consisting of a rectangular waste container in the center of the building. A shield wall with a rectangular duct with a bend going through it is also modeled. Figure 5.2 shows the geometry as modeled in a CAD package. The geometry consists of solid objects completely contained inside each other. There is no subtraction or addition in this model. A subtraction occurs when a body is created and a second created body is cut from it. An addition occurs when two bodies are unioned together.

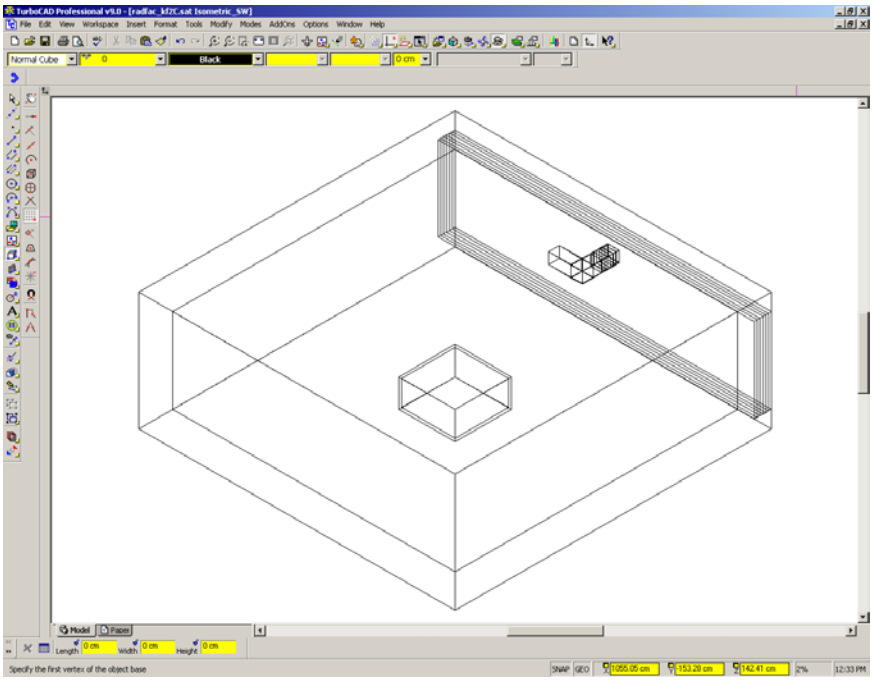

Figure 5.2. CAD Geometry of a Radiation Facility

This CAD file is then saved as a SAT file. The Visual Editor reads in this file and generates a 3D display of the geometry. Figure 5.3 shows the CAD geometry after it has been read in by the Visual Editor. In Figure 5.3, the outside of the room is shown in wireframe, the objects inside the room are transparent, and the shield wall is solid.

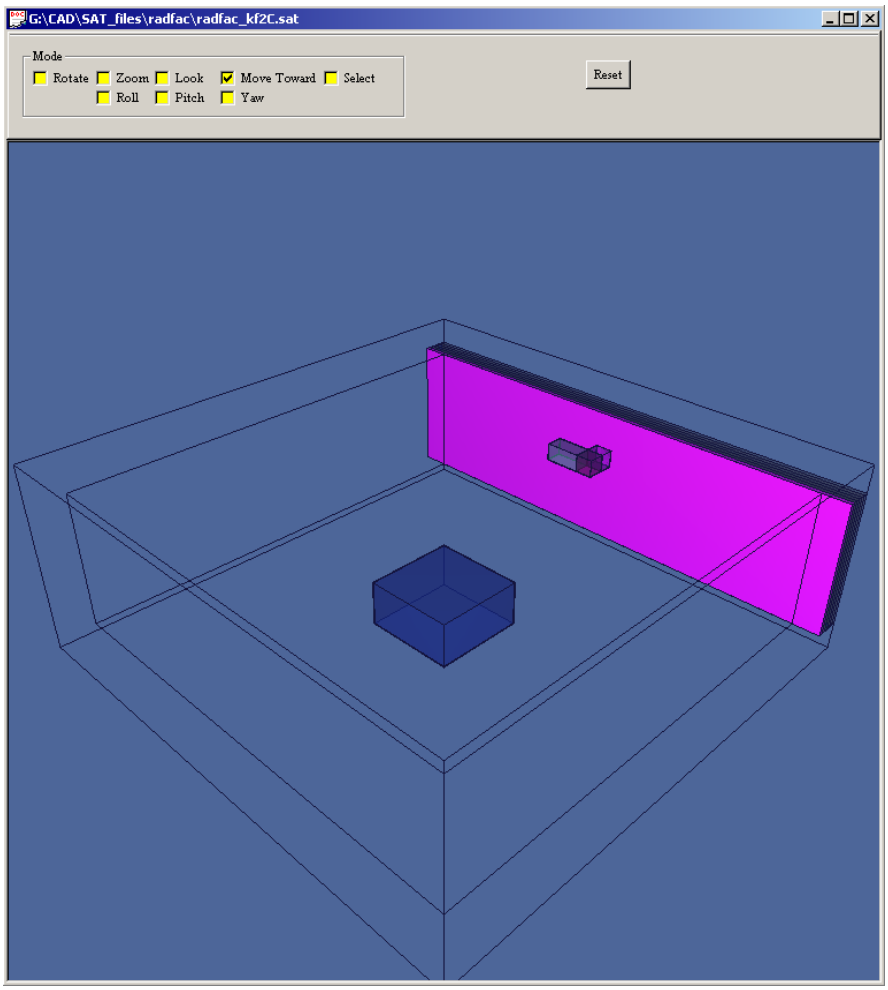

Figure 5.3. Display of the CAD Geometry in the Visual Editor

Figure 5.4 shows the geometry after it has been converted to MCNP. A close up view of the duct can be observed on the right side of the plot. A portion of the resulting input file is also shown in Figure 5.4.

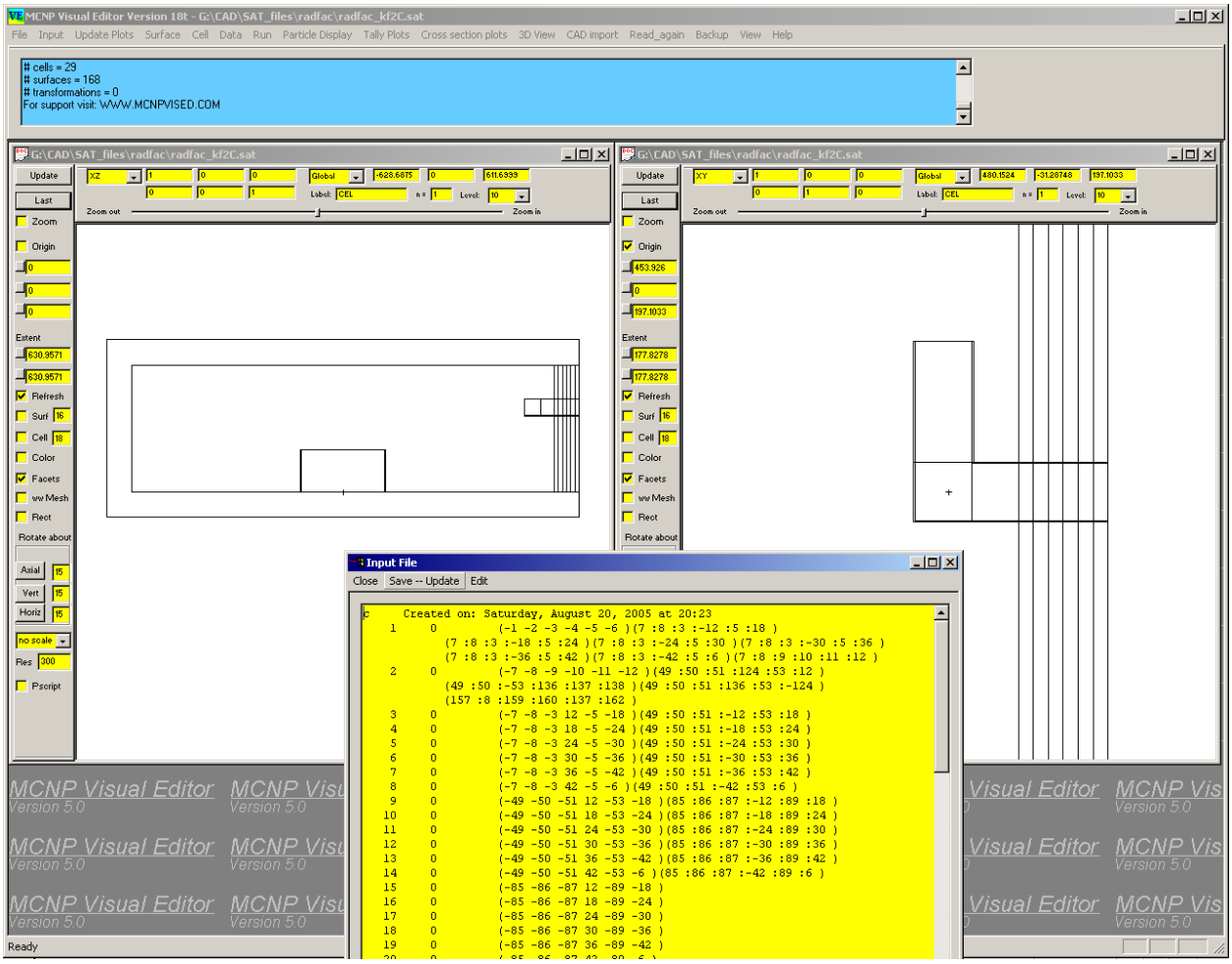

Figure 5.4. MCNP Geometry from Converted CAD File

After conversion to MCNP, the user can view the resulting MCNP geometry using the same 3D viewer that is used to view CAD geometries. This feature was added by extending the FORTRAN subroutines involved in the MCNP volume calculation algorithm to optionally calculate the vertices of the cell and then send this information to the graphical interface to be displayed.

Figure 5.5 shows the geometry again in 3D; however, this time it is using the MCNP description of the geometry to generate the 3D model.

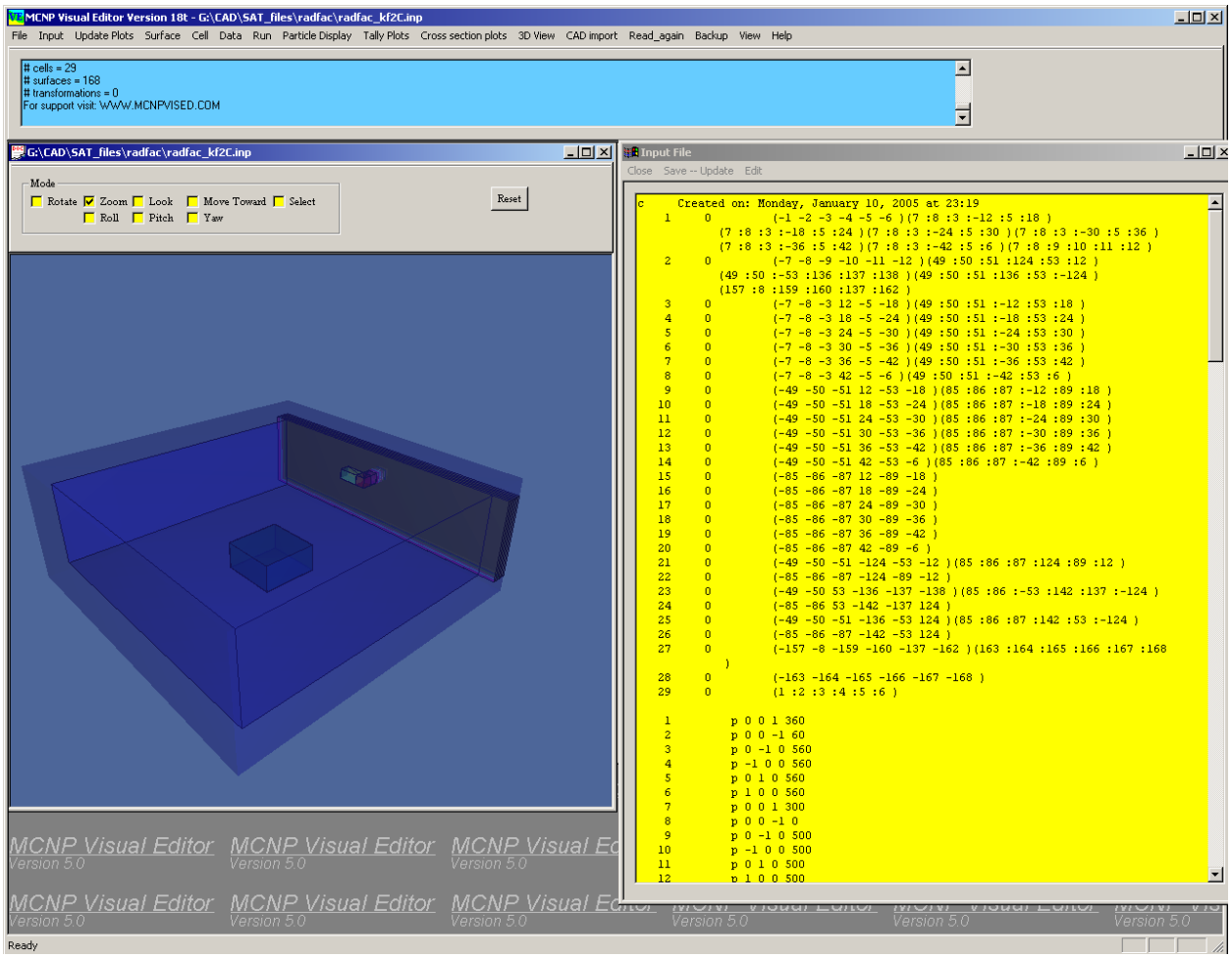

Figure 5.5. 3D Display of the MCNP Geometry

The MCNP geometry viewer is capable of displaying a wide array of MCNP geometries as demonstrated in Figure 5.6. Most of the general surfaces that are available in MCNP can be displayed. A resolution of the mesh used to create the plot can be controlled by changing the default mesh value of 100 (show in Figure 5.6) for the different types of surfaces. The code differentiates between polyhedra, symmetric surfaces, and asymmetric surfaces. The default value of 100 is usually sufficient for polyhedra and symmetric surfaces. For asymmetric surfaces, however, it is sometimes desirable to increase the resolution of the mesh used to display the geometry.

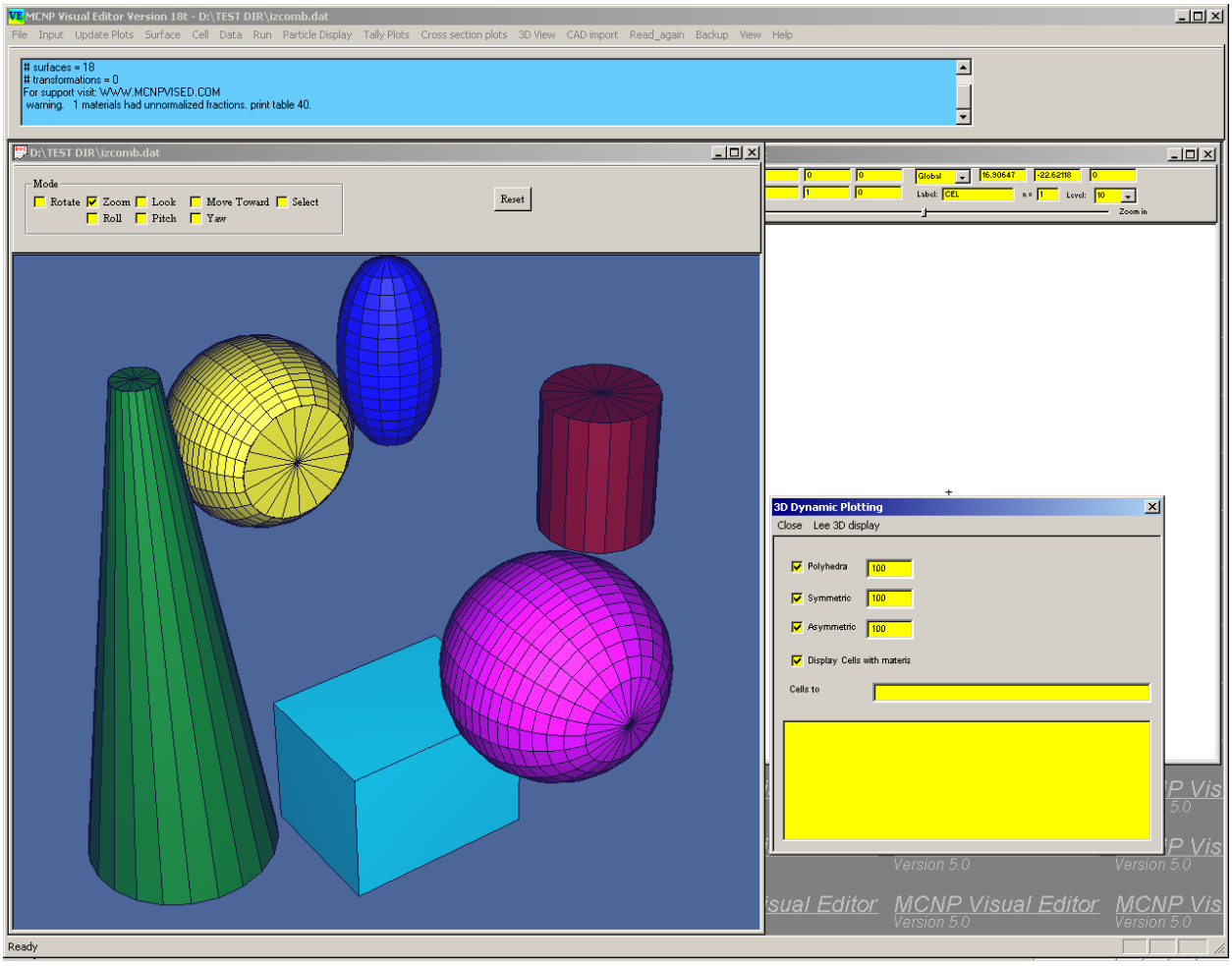

Figure 5.6. MCNP Cells that can be Displayed in the Visual Editor

6.0 Technical and Economic Feasibility

The CAD conversion code is mature enough to use in generating MCNP geometries and has already been released to the public through RSICC. Future funding for the code will come from the CAD to MCNP conversion classes and the financial support of organizations that need this type of capability and need support and training to adapt it for their applications.

7.0 Future Work

Additional time could be spent improving this work in the following areas:

- Improve the conversion of complex CAD objects. The goal is to increase the capability of the parsing algorithm to break complex objects into simpler objects that can be easily converted.
- Create an algorithm to approximate spline surfaces.
- Test the conversion code on increasingly complex geometries from industry.

- Continue to teach CAD to MCNP conversion classes.
- Convert MCNP geometries to CAD format.

8.0 References

1. J.F. Briemeister, Ed. "MCNP - A General Monte Carlo N Particle Transport Code, Version 4C," LA-13709-M (April 2000).

2. R.A. Schwarz, L.L. Carter, T P. Dole, S.M. Fredrickson, and B.M. Templeton, "Graphical User Input Interface for MCNP," *Trans. Am. Nucl. Soc.*, **69**, 401 (1993).

3. R.A. Schwarz, L.L. Carter, and N. Shrivastava, "Creation of MCNP Input Files with a Visual Editor," *Proc. 8th Int. Conf. on Radiation Shielding*, 454-459, April 1994, Arlington, Texas.
4. R.A. Schwarz and L.L. Carter, "Visual Editor for Creating MCNP Input Files," *Proc. Int. Conf. on Mathematics and Computations, Reactor Physics, and Environmental Analyses*, 1146-1147, April 30 - May 4, 1995, Portland, Oregon.
5. R.A. Schwarz and L.L. Carter, "Visual Editor for Creating MCNP Input Files," *Proc. Int. Conf. on Mathematics and Computations, Reactor Physics, and Environmental Analyses*, 1146-1147, April 30 - May 4, 1995, Portland, Oregon.
6. L.L. Carter and R.A. Schwarz, "Visual Creation of Lattice Geometries for MCNP Criticality Calculations," *Trans. Amer. Nucl. Soc*., **77**, 223 (1997).
7. R.A. Schwarz and L.L. Carter, "Visual Editor to Create and Display MCNP Input Files," *Trans. Amer. Nucl. Soc.*, **77**, 311-312 (1997).
8. R.A. Schwarz, L.L. Carter, K.E. Hillesland, and V.E. Roetman**,** "Advanced MCNP Input File Creation Using the Visual Editor," Proc. Am. Nucl. Soc. Topical, *Technologies for the New Century*, **2**, 317-324, April 1998, Nashville, Tennessee.
9. L.L. Carter and R.A. Schwarz**,** "The Visual Creation and Display of MCNP Geometries and Lattices for Criticality Problems," *Trans. Amer. Nucl. Soc*., (1999).
10. L.L. Carter, R.A. Schwarz, and J. Pfohl, "A Microsoft Windows Version of the MCNP Visual Editor," *Trans. Amer. Nucl. Soc*., **81**, 256-257 (1999).
11. R.A. Schwarz, L.L. Carter, and W. Brown**,** "Particle Track Visualization Using the MCNP Visual Editor," Proc. Am. Nucl. Soc. Topical *Radiation Protection for Our*

National Priorities Medicine, the Environment and, the Legacy, 324-331, 2000, Spokane, Washington.

12. X-5 Monte Carlo Team, "MCNP-A General Monte Carlo N-Particle Transport Code, Version 5 - Volume I: Overview and Theory," LA-UR-03-1987, April 2003 revised September 2003, distributed with RSICC package C00710.

13. X-5 Monte Carlo Team, "MCNP-A General Monte Carlo N-Particle Transport Code, Version 5 - Volume II: Users Guide," LA-CP-03-0245, April 2003, distributed with RSICC package C00710.

14. X-5 Monte Carlo Team, "MCNP-A General Monte Carlo N-Particle Transport Code, Version 5 - Volume III: Developers Guide," LA-CP-03-0284, April 2003, distributed with RSICC package C00710.

15. L.L. Carter and R.A. Schwarz, "MCNP Visual Editor Computer Code Manual," 2002, distributed with RSICC package C00710.

16. R.A. Schwarz, A.L. Schwarz, and L.L. Carter, "Conversion of Computer Aided Design (CAD) Output Files to Monte Carlo N-Particle (MCNP) Input Files," Monte Carlo 2005 topical meeting, *The Monte Carlo Method: Versatility in a Dynamic Computing World*, April 17-21, 2005, Chattanooga, Tennessee.

Appendix A. Technical Discussion of the FORTRAN for Conversion of 2D CAD *.dxf Files to MCNP

A.1 Introduction

The Final Technical Report Document for the 2D CAD conversion summarizes the 2D conversion and describes the steps involved to convert a 2D CAD file that is in a dxf format. The purpose of this appendix is to describe in some detail the FORTRAN subroutines and additional common that were added to do this conversion. The corresponding upgrade of the C for this conversion will be described elsewhere. Besides providing additional panels for the 2D CAD conversion, the C also does the initial work for such a conversion. It reads the CAD file and provides the plot information so the user can view the plot of the CAD geometry. From the information on the CAD file it generates the standard information (surface coefficients) for each MCNP surface as well as the (x,y) values at the beginning and end of each segment that is used along this surface. After giving this information to the FORTRAN, it then turns the effort over to the FORTRAN for the FORTRAN to determine the MCNP cells (each cell is a contiguous "white space" on the 2D plot) for the geometry. After the FORTRAN determines the cells and any other MCNP surfaces it needs, it then gives this information back to the C for the C to prepare for plots of the MCNP geometry and to optionally save the final MCNP input file.

The FORTRAN task is quite complicated because it starts out with only the MCNP surfaces and their segments with the assignment to create the complete MCNP geometry. The intent in Section A.2 is to describe how this is done in a fair degree of technical detail.

A.2 Technical Discussion of the FORTRAN Upgrade

A.2.1 Overview of the Flow in the FORTRAN 2D Conversion

When the FORTRAN is called, the only information it has is the MCNP surfaces along with the (x,y) values on each end of all the segments that are part of the CAD geometry involving each surface. A surface segment begins/ends whenever the white space is intercepted by another surface on either side of the surface. There are also two flags for each segment, flags that are set by the FORTRAN for each side of the surface segment when the segment is first used for the new MCNP cell containing the "white space" on a side of the segment. When all these flags have been set, the calculation of the MCNP cell cards is complete since there is a cell on each side of each segment.

The outer perimeter of a new cell is determined as follows. The FORTRAN begins with a CAD surface segment and a direction from the segment into the new cell corresponding to a flag for the segment that has not yet been set. Then the next segment along this outer perimeter of the cell is determined by comparing all the (x,y) ends of the segments, with flags that have not yet been set, to the (x,y) end of the first segment. If there is only one such segment, then this is the next segment of the perimeter. If there is more than one segment that matches, then the angle between the original segment and a possible new segment is computed for each of the matching

segments and the segment that leads to the minimum angle becomes the second segment. This is then repeated to find the third segment and so forth until the perimeter is completed when the opposite end of the first segment is reached. The flag is set for each of these segments on the perimeter, where there is a flag for the positive direction from the surface segment and another for the negative direction, so the cell being created determines which flag is set. The segments for the perimeter are stored in a temporary variable along with direction cosines along each segment, the direction cosines normal to the segment that go into the new cell, and the angle between the segment and the next segment in the chain.

The next task is to determine the surface relationships for the new MCNP cell card that describe this outer perimeter for the cell. The MCNP surfaces involved for the perimeter correspond to the all the surfaces that contain one or more of the segments for the perimeter. Some of the angles between segments may be exactly 180 degrees so both segments in this case involve the same surface so that surface will not be repeated on the new MCNP cell card. If all the angles between adjacent segments around this outer perimeter are less than (or equal) 180 degrees, there are no unions involved. Then the surface description on the new cell card for this perimeter is simply all these surfaces, with appropriate surface sense corresponding to the direction from the corresponding segment into the cell, separated by blank spaces. If the angles are not all less than 180 degrees, at least one union is involved and the surface relationships for the outer perimeter are more complicated. Then a new MCNP surface is created at each angle greater then 180 degrees such that this new surface bisects the angle. These new surfaces split the cell into two or more sub-cells. Each sub-cell will then have all angles less than (or equal) 180 degrees and the surface relationship on the MCNP cell card for this sub-cell is simply the surfaces around the sub-cell perimeter, with appropriate sense, separated by blank spaces. The full perimeter is then described by the union of these surface descriptions for each of the different sub-cells.

The new MCNP cell at this point is just described by its outer perimeter. However, this new cell may have one or more inner perimeters that exclude certain interior portions from the contiguous "white space". Ray tracing is used from a number of points located on the outer perimeter of the cell, where these rays are directed inward and toward each of the as yet unflagged CAD surface segments. If none of these rays reach an unflagged CAD surface segment before they reach the boundary of the outer perimeter, then the description of the new cell is complete. If a ray reaches an unflagged CAD surface segment before the outer perimeter, then the inner perimeter must be constructed as described in the next paragraph.

The inner perimeter of the new cell is determined much like the outer perimeter was determined by finding the unflagged surfaces segments that link together for a complete inner perimeter. Again the flag is set for the appropriate sense direction for each of these segments on this inner perimeter and the segments for the inner perimeter are stored in a temporary variable along with direction cosines along each segment, the direction cosines normal to the segment that go into the new cell, and the angle between the segment and the next segment in the chain. The surface relationships for the new MCNP cell card that describe this inner perimeter for the new cell are now determined. If all the angles between the segments around this inner perimeter are greater than (or equal) 180 degrees, there all are no intersections involved. Then the surface description on the new cell card for this inner perimeter is simply all these surfaces, with appropriate surface sense corresponding to the direction from the corresponding segment into the cell, and each

separated by a union ":" operator. If one or more angles is less than 180 degrees, the determination of the surface relationship for the inner perimeter becomes much more complicated. This surface relationship is obtained by first doing a compliment (in MCNP notation, a # operation) of the inner perimeter to temporarily convert it to an outer perimeter. Then the surface relationship becomes the union of sub-cells as described above for an outer perimeter. Finally, another complement (another # operation) is performed to convert the expression back to the desired surface relationship for the inner perimeter. The above procedure is then repeated with more ray tracing to find any additional inner perimeters for the new cell. Then the new cell description is complete.

A segment on the outer perimeter of the new cell that does not have its flag set for the other side (opposite side of the new cell) can be used as a starting point to determine an outer perimeter of the next new cell, which is at least partially on the other side of the last new cell. The above procedure for finding the outer perimeter of this next new cell is repeated and then the procedure for finding its surface relationship for the cell card and the procedure for finding the surface relationship for any internal perimeters is repeated.

The actual flow of the coding differs in two aspects from that described above: (1) after doing an inner perimeter, any new cells within this inner perimeter are also created by beginning with one of the segments on the inner perimeter and forming an outer perimeter on the other side, etc.; and (2) the first initial perimeter is actually chosen to be the inner perimeter of the outside world selected by finding the approximate (x-bar,y-bar) center of the segments based on a linear average of their (x,y) ends and beginning with a segment the greatest distance from this center and using the outward direction from the segment.

A.2.2 Detailed Discussion of the FORTRAN 2D Conversion by Subroutine

The following subroutines are described in the chronological order in which they tend to be used. At the beginning of the summary for each subroutine, a few sentences are included to briefly describe what the subroutine does, what subroutines it calls, and the information the subroutine returns. Because the upgrade discussed here is based on version 4C2 of MCNP (with similar considerations for Version 5 with additional "#keyword=VECAD" instead of using prpr and using FORTRAN 90), the upgrade relies on a "patch" file that contains the changes to version 4C2 using the prpr module. These changes not only include the changes for the CAD conversion, but also the FORTRAN changes to go to the Visual Editor. In addition to "vised" and other keywords in the patch file for the Visual Editor, the keywork "vecad" must be included for the CAD conversion portion of the upgrade. This "vecad" brings in an additional block of common, 11 subroutines for the CAD 2D conversion, and a number of subroutines for the CAD 3D conversion that are not discussed here. We begin by discussing the additional block of common and then discuss each of the 11 subroutines. Note: there is a corresponding Version 5 with its keywords that is not discussed here, but the subroutines are the same except for the conversion to FORTRAN 90.

A.2.3 Additional Block of Common for 2D CAD Conversion

The new subroutines follow the MCNP FORTRAN conventions in naming variables: variables that start with (i,j,k,l,m, or n) are fixed point variables; those that start with anything else are floating point variables (usually double precision unless specified as real) except for variables that are specifically designated as hollerith; and variables that are more than two characters are in common while variables with only one or two characters are local variables. The following common variables are in the common block "viscad" and are available in each of the 11 subroutines for specific 2D CAD conversion tasks:

1) Iacad is the current MCNP cell number being addressed (=1 is outside world);

2) Icadp is the number of the cell perimeter currently being addressed;

3) Jcads is the number of CAD segments in the current perimeter;

4) Jnicad is the number of increments per surface segment to use in FORTRAN solution, (default 2);

5) Ncadp is the total number of active perimeters (<=31);

6) Mxjcad is the current total number of MCNP surfaces;

7) Xcadc=approximate x-center of unused cad surface segments;

8) Ycadc=approximate y-center of unused cad surface segments;

9) Jcadtt(I,k) is fixed point information for an inner or outer perimeter for a MCNP cell. Here, the perimeter number I=icadp varies from 1 to a maximum of 31 [when a perimeter is no longer needed that I is eliminated by moving everything from "I+1" down by one]. The slot k contains the following information:

   K=1, is "1" if it is an outer perimeter or "2" if it is an inner perimeter;

   K=2, is the number of CAD surface segments for this perimeter;

   K=3, is the MCNP cell number where this perimeter is used,

   K=4, if an outer perimeter is the next CAD surface segment number in this perimeter to be checked for determining a possible inner perimeter on the other side of the segment.

10) Jcadss(I,j,k) is fixed point information for an inner or outer perimeter for a MCNP cell that gives information about each CAD surface segment along the perimeter. Here, I=icadp varies from 1 to a maximum of 31 perimeters [when a perimeter is no longer needed that I is eliminated by moving everything from "I+1" down by one], j=jcads

varies from 1 to the number of CAD surface segments in the chain of segments for the perimeter with a maximum of 200. k contains the following information for each CAD segment:

K=1, is the MCNP surface number for the CAD segment; [a positive surface number means the cell is on the positive side of the segment]

K=2, is the CAD segment number; (negative if first end is end of vector)

11) Gcadss(I,j,k) is the corresponding floating point array for the perimeter:

K=1, is the rotation angle (radians) from this CAD segment to the next for the white space of the MCNP cell;

K=2, is the x component of the surface normal pointing toward the cell at the end of the CAD segment,

K=3, is the y component of the surface normal pointing toward the cell at the end of the CAD segment,

K=4, is the x component along the surface segment in the direction around the perimeter,

K=5, is the y component along the surface segment,

A.2.4 New Subroutines

Following are the subroutines that were added/modified/used to do the 2D CAD conversion to an MCNP input file, where the order here is roughly the order of the code flow.

1) Subroutine cadnsur(i1,i2,kz,jg) is called by the C to give the FORTRAN each new MCNP surface for the 2D conversion. It parallels subroutine visnsur() so the first three entries have the same interpretation, where i1 must be 0, i2 is not used so its interpreted as 0, and kz is not used so its interpreted as 0. If jg is 0 or 1, the geometry is the usual (x,y), while jg=2 the geometry is (x,z) so the FORTRAN converts to (x,y) or jg=3 the geometry is (y,z) so the FORTRAN converts to (x,y) – jg=2 or =3 has not been tested very much although many of the lines in this subroutine involve these seldom used options. The surface information is passed by the C in the icvis() and cdvis() much like with subroutine visnsur(). An important difference is that each CAD segment along the surface is also passed [see the comment cards at the beginning of cadnsur] with icvis(1) being the total number of surface coefficients passed including this segment information. Following the usual surface information in cdvis() is the number of "CAD segments" followed by six entries to describe each CAD segment [typically two flags for each direction from the surface, the (x,y) for one end of the segment, and the (x,y) for the other end of the segment]. This CAD segment information is also put in the FORTRAN surface coefficient array scf().

2) Subroutine cadcel(jq,jp,jm) handles the code flow to determine the MCNP cells calling other routines as needed. This subroutine is called by the C after the C has given the FORTRAN all the MCNP surfaces with calls of cadnsur(). Here jq is a convergence parameter representing the number of increments on a CAD surface segment [usually 2 but could be set to 10 if the conversion fails], jp is the axial bounding surface of the 2D problem in z+ and jm is the axial bounding surface of the 2D problem in z- [if jp is zero there are no axially bounding surfaces desired – otherwise these two bounding surfaces are put on each MCNP cell card with appropriate senses]. After initializing a few parameters, this subroutine calls cadcen() to determine the (xcadcen,ycadcen) center for all CAD surface segments that have not yet been flagged [none flagged as we start, or later if all are flagged js will be zero after this call for transfer to statement 120]. It then enters a loop starting at statement 20 where it determines the inner chain of the outside world [begins with this] or an inner chain of a regular cell [for which it has already obtained the outer perimeter] by calling cadperc(). After writing information for the chain to the outp file, it eventually reaches statement 50 where it then looks on the other side of the segments of the last chain to find an "other-side" segment that has not been used. If such a segment is found it goes to statement 60 to start a new cell, or if it doesn't find anything, it will go to statement 100 since this sequence is complete. After setting some parameters after statement 60 for the new cell it calls cadperc() again [note that the third parameter in the call is "1" indicating an outer perimeter will be determined] to determine the outer perimeter of cell mxa. After writing some information to the outp file it eventually reaches statement 90 with a call of cadinn(js,jk,je) to search for an inner perimeter of the cell. If an inner perimeter is found [js.ne.0] it loops back to statement 20 to determine the inner chain of this cell. Otherwise it reaches statement 100 where the current perimeter is complete. It then looks for another potential inner perimeter. If there are no more perimeters it goes to 102. If there is an inner perimeter it goes to 50 to examine this for potentially creating another cell. If this is an outer perimeter it goes to statement 90 to look for another inner. At statement 102 it looks through all the flags of the segments to see if one side of a segment has not been used, but the other side has been used in which case it goes to statement 60 to deal with this "js" surface and "jk" segment by creating a new cell on this other side. Otherwise, it returns to statement 10 to deal with the remaining segments whose flags have not been set. When all flags have been set, the call of cadcen(js,jk) will return a js=0 and there is a jump to statement 120 where eventually the "do 140" loop gives each cell card to the C with a "call cnew(4,0)". In the process of creating the cells, the FORTRAN may have created some extra surfaces and these are given to the C in the "do 145" loop with a "call cnew(3,0)" for each surface. Following this there is either a return or else the FORTRAN deals with the optional axial z surfaces, and will eventually loop back to statement 120 to give the cells to the C with these axial surfaces included. Subroutine cadcel has a number of fatal error checks where, if there is a fatal error, it jumps to statement 980 and displays the fatal error flag in the "Input" window and returns.

3) Subroutine cadcen(js,jk) computes the (xcadc,ycadc) average geometric center of all segments that have not yet been used (flagged), and determines the MCNP surface

number, js with sense, and CAD segment, jk [with jk negative if the beginning of the segment or positive if the end] of the segment that is the largest distance from the center. It also determines (xxx,yyy) as the location of a point on this segment and (uuu,vvv) as the direction from (xxx,yyy) to the (xcadc,ycadc). It sets js=0 if all segments have been used; i.e., all flags have been set on both sides of each segment.

4) Subroutine cadperc(js,jk,jp,je) begins with MCNP surface js and CAD surface segment jk to create an MCNP surface chain perimeter for the cell, where if jp=1 it is an outer perimeter and if jp=2 it is an inner perimeter. The variable je is an error flag. This subroutine calls cadperm to do most of the work and then based on the information from cadperm, it jumps to statement 400 or statement 460 to create the MCNP surface chain for this perimeter if its fairly simple or it falls through and calls cadnew for a complicated perimeter that requires the union of a number of simple sub-cells [each of which has only intersections]. In either case it eventually reaches statement 460 where the MCNP surface parameters for this perimeter are set for the cell [in the lja() array,etc – see especially the "do 500" loop], and this information is also written to the outp file. The "do 525" loop also writes out the cells to the outp file in a usable MCNP format, which is sometimes useful for debugging.

5) Subroutine cadperm(js,jk,jp,je,is,nc,mc) begins with MCNP surface js and CAD surface segment jk to create an MCNP surface chain perimeter for the cell, where if jp=1 it is an outer perimeter and if jp=2 it is an inner perimeter. The variable je is an error flag. This routine sets the variable "is=-1" if the perimeter is a simple full circle or sets "is=0" if it's a simple perimeter with intersections or "is=1" for more complicated things. The parameter "nc" is the number of MCNP parameters in the chain that is in the array "mc()". Beginning at statement 10, all other surface segments whose flags have not been used are examined [the "do 50", "do 40", and "do 30" loops] to see which one has an end that links up [exactly matches with the end of the current surface segment] with the current segment of the MCNP surface chain. At statement 25 it has found a match and then if there is more than one matching segment, it eventually selects the segment with the minimum angle for the "white space" [see the "go to 40" statement about 20 statements after statement 27] and sets the gcadss(icadp,jcads,) and jcadss(icadp,jcads,) parameters for this new segment with jcads the segment number in the chain icadp. At statement 55 it sets some parameters to prepare for the next segment and jumps back to 10 for the next CAD segment unless the chain is complete in which case it falls through to statement 60. Following statement 70 in the "do 90" it will start the chain at a different place to avoid an angle of 90 degrees [straight ahead segment] for the last segment if it is 90 degrees. The "do 160" creates the MCNP cell card for the chain [surfaces with sense in array mc() with total number nc], and it is all through if its simple [is=0]. Otherwise, it will be split into sub-cells for the cell definition after the return to cadperc().

6) Subroutine cadnew(nc,mc,jp,je) determines a complicated perimeter as the union of sub-cells. The parameter "nc" is the number of MCNP parameters in the chain that is in the array "mc()", jp is the perimeter type, and je is an error flag. The jcadss() and gcadss arrays are written to a temporary space (np=ncadp+1) for this determination.

Then call cadseg(np), incrementing np by one as necessary, to check for sub-segments, where there will eventually be a jump back to statement 20 to do each sub-segment. Arcs are isolated with their chord (or additional complicated chords if necessary) with a call cadarc(). The arrays km and qm are used to show intersections of new surfaces and ambiguity surfaces – this use of these arrays is defined in the comment cards. Each angle greater than 90 degrees has a new surface created in the "do 190" loop to bisect, where each bisect leads to a new sub-segment and locators and data is moved beginning at "do 95" to accommodate this including the evaluation of the surface sense. The coding beginning at statement 200 looks for ambiguity surfaces. Surfaces are put in memory after statement 280. There is a jump to statement 300 (right after statement 200) after all ambiguity surfaces have been treated. Then the sub-cell surface string is determined and put in the array mc() beginning at statement 310 with an eventual jump to statement 500 [Note: this is quite complicated so should study comment statements for details]. Following statement 500 there is a jump back to statement 20 if there are any more sub-segments. Otherwise we are through unless "jp=2" in which case a "#" is used on mc() to turn it back to an inner perimeter.

7) Subroutine cadseg(np) checks for "jp=2" for sub-segments of the cell and increments np by one for each one found. This information for each sub-segment is put in jcadss(np,,) and gcadss(np,,). The comment statements describe what is being done.

8) Subroutine cadarc(nc,mc,np,m9,m8) reduces a minor perimeter to all planes by finding a chord for each (negative) arc with a call of cadchord(). Then separate sub-cells are created for each chord with appropriate adjustments to the jcadss() and gcadss() arrays. Here np is the temporary sub-segment locator, and the m9 identical surfaces are stored in the m8(m9,) array.

9) Subroutine cadchord(j1,j2,np,ms) determines the chord (ms=1) between the ends of the arc from j1 to j2 and creates this surface with np the temporary sub-segment locator. If the chord intersects the geometry (ms>1), the first angle of this chord is reduced until it does not intersect, and ms is increased by one to create another chord, etc. Each new chord is put in memory following statement 40. The distance to a surface jf and "CAD" segment jc [if jc=0, do for all segments] is determined with a call of cadnsp(jf,jc).

10) Subroutine cadnsp(jf,jc) determines the distance dls to a surface jf and "CAD" segment jc [if jc=0, do for all segments].

11) Subroutine cadinn(ks,jk,je) determines by ray tracing if there are any inner perimeters in cell iacad for outer surface perimeter sequence ncadp. If one is found, it returns the MCNP surface ks and segment jk. The "do 200" loop is over the outer perimeter segments so rays are started along each of these segments in the "do 190" loop. Then the "do 180" loop is over all of the MCNP surfaces to find valid potential inner segments. Then vistrk() is called to find the distance from the outer to the inner, and if this distance is less than the distance to the outer boundary of the cell, the ray has found a potential valid inner surface (provided there is not one with a smaller distance). If so, this information is stored at statement 120.

A.2.5 MCNP/VisEd Subroutines and Functions

1) Subroutine chkcel(ia,2,j) is an MCNP subroutine that determines if point (xxx,yyy,zzz) is in cell ia. It returns j=0 if the point is in the cell.

2) Subroutine track(ih) is an MCNP subroutine that determines the distance, dls, to a surface of cell ih for a ray starting at (xxx,yyy,zzz) going in direction (uuu,vvv,www).

3) Subroutine vistrk(j) determines the distance, dls, to surface j for a ray starting at (xxx,yyy,zzz) going in direction (uuu,vvv,www).

4) Function angl() is an MCNP function that determines the positive normal, ang, to suface jsu for location (xxx,yyy,zzz).

5) Function dotpro(v1,v2) determines the dot product between the vectors v1 and v2, both dimensioned to three.

6) Subroutine crspro(v1,v2,v3) determines the cross product with v3=v1xv2.

Appendix B. Discussion of the FORTRAN Routines for the Conversion of 3D CAD Files

B.1 Introduction

It is assumed in the first four sections of this appendix that the *.sat output from CAD entirely uses the surface "skin" representation of each object. These first four sections discuss the algorithm to go from this CAD-3D representation to an MCNP input file with the control "kf=2". In Section B.5, we also include some discussion of a more recent "kf=4" option that allows limited solid modeling from CAD with CAD Intersections, Subtractions, and Unions in the *.sat file that are converted to an MCNP input file.

It is assumed that the output file for each "skin" CAD surface (actually a CAD body since it may include many surfaces, and this will usually be referred to as body) contains sufficient information to construct the MCNP outer perimeter for the MCNP cell containing each "body. If that is not true, the CAD body would need to be replaced with two or more simpler body representations; i.e., it is currently assumed that each body perimeter has no MCNP unions involved after conversion.

In Section B.2, the algorithm is summarized for constructing the MCNP surfaces and cells corresponding to the CAD body representations with "kf=2". In Section B.3, simple examples of box-in-box and sphere-in-box are shown and discussed. In Section B.4, constraints on the CAD body geometry are discussed in order to allow the conversion to the corresponding MCNP geometry. Section B.5 includes a discussion of the recent "kf=4" option that allows limited solid modeling from CAD with CAD Intersections, Subtractions, and Unions in the *.sat file that is converted to an MCNP input file. Finally, Section B.6 summarizes the current implementation in the Visual Editor based on Version 4C2 and Version 5, where the subroutines for Version 5 are essentially the same except they use FORTRAN 90. This conversion of the CAD representation to the corresponding MCNP geometry is rather independent of the MCNP Version. Section B.6 includes a specific discussion of each FORTRAN subroutine involved in the conversion.

B.2 Algorithm to Convert CAD 3D Skin Representation (kf=2) to MCNP Input File

The algorithm to construct the MCNP surfaces and cells corresponding to the CAD body representation is as follows:

1) Read in each CAD surface of a body from the CAD *.sat output file and determine the corresponding MCNP surfaces and then determine the corresponding MCNP cell card that at this point only has the outer perimeter for the CAD body involving these surfaces. In addition, for each body [outer perimeter of the MCNP cell] determine a set of (x,y,z) points that are along each surface, but just internal to the skin by a delta of perhaps 0.00001 cm. These points will be used to determine if this body is inside

another body. Also, one or more specified bodies must include all the space that is the interior to the outside world; for example everything interior to a sphere or everything interior to a cylinder and its two ends. A flag for each of these outer CAD bodies [MCNP cells] is set to "-1" – otherwise the flag is initially zero for the cell. An outside world MCNP cell will automatically be created that is beyond these flagged cells. For the boundary conditions of a valid outside world, this outer perimeter as viewed from the outside world should not have any concave angles. These flagged cells cannot cut through the interior of any un-flagged cells; i.e., can only be along the exteriors of any interior cells – can only pass between two interior boxes, for example, and not through either one. Set "n=0" in preparation for step 2.a.

2) (2.a) Increment "n" by "+1" until a MCNP cell outer perimeter "n" from step #1 is found that has a flag of "0". A check is then made to see if any of the other cells [bodies] without a flag set are interior to it using the MCNP subroutine chkcel for each (x,y,z) point for the cell [for a valid interior designation all points must be interior since, for example, the axial of two cells might have the same bounding surface]. If no cells are interior to cell "n", transfer is made to step 2.b. If there is an interior cell, "n" is advanced to the next cell that has its flag ="0", which is examined for being interior. Eventually a cell "n" should be found without another cell as an interior so the flag is set to "1" and we proceed to step 2.b, or else if all flags have be set so there is not another fully interior cell, we have dealt with all the cells and go to step 3. (2.b) If the flag of cell "n" is "0", it is set to "1". The cells are searched to find a cell "m" whose flag is not "+1" or "–1" and for which cell n is interior to cell m. [if more than one cell "m" is found, these cells are compared to each other to find which is interior to which and the most interior becomes "m"]. If the flag of cell "m" is >1, transfer is made to step 2.c. An alternative is that the flag of cell "m" is "0", in which case a check is made of all the other cells whose flag is not set to see if there are any additional cells interior to it besides cell "n" – each of these cells only counts if it doesn't have an exterior cell that is still interior to cell "n", and for each such cell the flag of cell "m" is incremented beginning with "2" for the first such cell; i.e., there may be another cell interior to cell "m" that is not interior to cell "n" and cell "n" is not interior to it. Then transfer is made to step 2.c. The final possibility is that such a cell "m" is not found, in which case each of the cells with flag=-1 [next to outside world] are examined. Then such a cell that has cell "n" interior becomes cell "m" and transfer is made to step 2.c [if such a cell doesn't exist, it is a fatal error]. (2.c) The # [see MCNP manual] of cell "n" ["skin" only (original body) of cell "n"] is included on the surface description of cell "m". If the flag of cell "m" is zero or has just been set ">1" in step 2.b, temporarily "n" is set to "m" and transfer is made to step 2.b to find subsequent exterior cells up the chain. Otherwise, the flag of cell "m" should now be "-1" or ">1". If it is ">1", reduce it by 1. In either situation, set "n" back to the last valid value of "n" in step 2.a and return to step 2.a.

3) The outside world MCNP cell is now created whose surface description includes the # of each of the cells [body portion only] that have a flag value of "-1"; i.e., are adjacent to the outside world and their volume sum includes all the internal volume. If there is

only one cell with "-1" and this outer perimeter is a sphere, then the outside world cell will only contain this surface number (positive since outside the sphere).

During implementation it was found convenient to temporarily store the values of "m" and "n" in the step involving the first sentence of step 2c rather than modifying the surface description of cell "m". Then at the beginning of step 3, these modifications are made one at a time.

A simple example of the above algorithm is shown in Section B.3. In the above algorithm the cell flag is set to "-1" initially for an MCNP cell that is adjacent to the outside world [the cell (body) is contained by no other cells (bodies)], where this cell may be the perimeter for a number of cells. Otherwise, the flag is set to "1" when a cell "n" is found that has no inside cells or only has one unique inside cell perimeter; i.e., there is a cell internal to cell "n" that contains all the other cells that are internal to cell "n'. The flag is set to "1+q" for a cell that has "1+q" unique inside cell perimeters, and then this flag is reduced by "1" as each unique cell is treated until it reaches "1".

The above algorithm assumes that the outer perimeter of an interior cell is completely contained within the outer perimeter of another cell, although they may have common surfaces. This is a constraint on the CAD geometry creation. See the second paragraph of Section B.4 for a more precise statement of this uniqueness requirement. In Section B.3 a simple example is shown to illustrate why this is necessary, or at any rate, why it is a very simplifying assumption that assures uniqueness.

Note on current implementation status: The above discussion is for an original "kf=1" implementation, where the FORTRAN reads the *.sat file. We have since upgraded to a "kf=2" implementation, where the C reads the *.sat file and gives the information to the FORTRAN. Typically with this "kf=2" the C gives only one (x,y,z) point inside the perimeter of a body and the FORTRAN uses random rays from that point to the boundary of the body (or MCNP cell) to find points near the perimeter of the cell.

B.3 Box-In-Box and Sphere-In-Box Examples Converting CAD 3D Skin Representations (kf=2) to MCNP

Figure B-1 is a MCNP simple example that illustrates how the above algorithm works. This is illustrated using an MCNP macrobody as to describe each CAD surface body. This is convenient to do using the MCNP macrobody, which can each be thought of as similar to a CAD body, but each of these could actually also represent a number of regular MCNP surfaces if those were used instead of a MCNP macrobody. Below the views of Figure B-1, the MCNP input file is shown. Here, surfaces 1 to 6 [ignore surface "1" at the top and bottom of the right hand plot since that is the way MCNP numbers with identical surfaces] represent the CAD surface bodies with surface 6 being the sphere boundary for the outside world, and cells 1 to 7 are the resulting MCNP cells after applying the algorithm.

The algorithm would begin by reading in these CAD surfaces for each body, then converting to MCNP surfaces [in this case the MCNP macrobodies that are used to simplify the discussion], determining the MCNP outer perimeters of the 6 cells [the first (negative) surface entry on each

of these cell cards], and setting the flag of cell 6 to "-1". Then it would search through the first 5 cells [not cell 6 since its flag is set] for an outer perimeter that does not bound another cell, where either cell 1 or cell 4 satisfy this criterion. This is done by looking at the (x,y,z) points near the surface "skin" to see if they are in any other cell that does not have its flag set. Assume here we determine cell 1 as valid, so we would set its flag=1. Then we would look for the cell that only contains the outer perimeter of cell 1 whose flag has not been set; i.e., cell 2. We set its flag and do the # of the original cell 1 on the cell 2 card "4" and repeat to find cell 3 and set its flag and do the # of the original cell 2 on the cell 3 card "3". Now we find there are no cells that contain cell 3 whose flag has not been set so we look at cell 6 [the only cell with flag=1] and it contains cell 3. Hence, we set the flag of cell 3 and do the # of the original cell 3 and put it on cell 6; i.e., the macrobody surface "1".

Then the only cells remaining whose flags have not been set are cells 4 and 5. Repeating the above, we include a "5" on cell 5 and a "2" on cell 6, setting their flags to "1". If the surfaces had been regular MCNP surfaces [each CAD surface would correspond to a regular MCNP surface in an actual conversion] instead of macrobody surfaces, each surface in this illustration would be replaced by the intersection of the involved surfaces with appropriate senses, and the "#" would change all signs and replace each intersection with a union. Identical surfaces are eliminated during the conversion.

Then we create the outside world cell 7; i.e., the # for the body of cell 6.

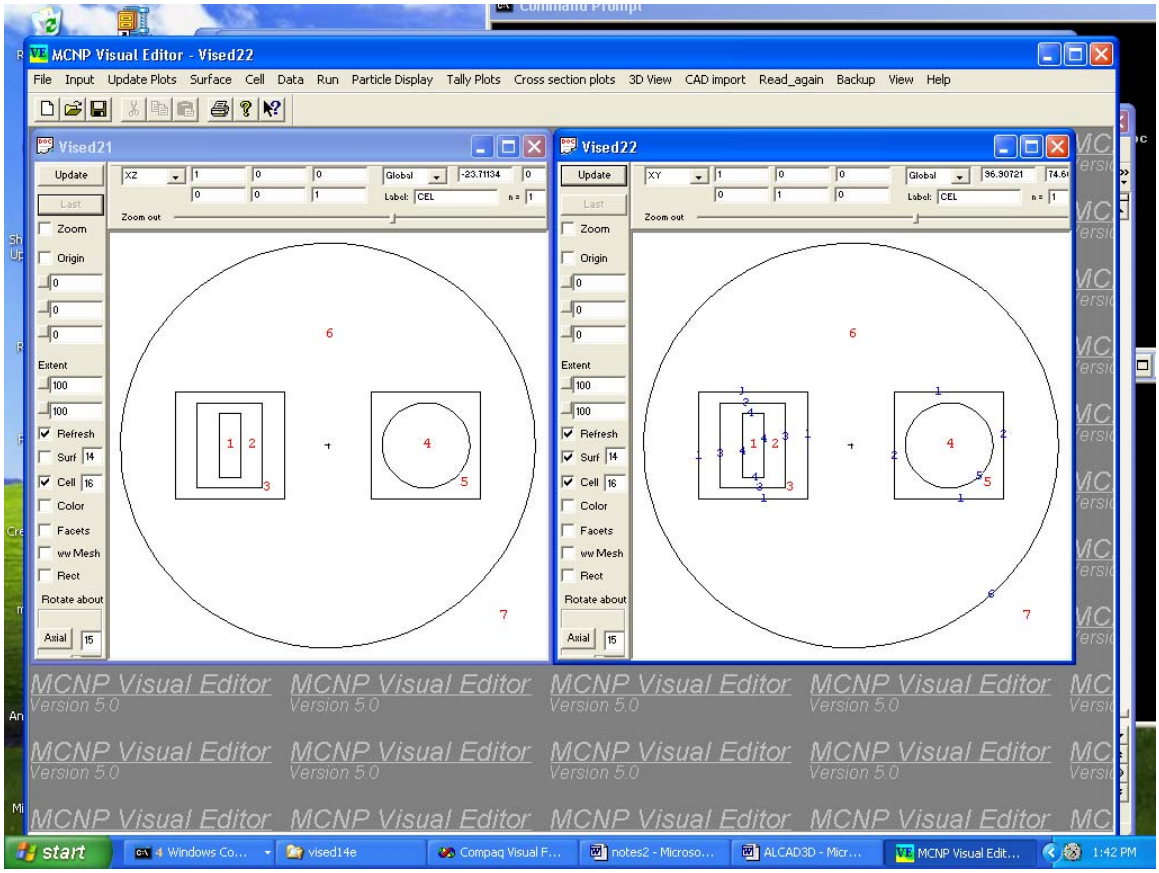

Figure B-1. Illustration of 3D CAD to MCNP Algorithm ([x,z] View on Left, [x,y] on Right)

MCNP example converted input file to illustrate using MCNP macrobody notation:

```
example for 3-D cad in MCNP format for discussion, icad1
    1     0          -4
    2     0          (-3 )(4 )
    3     0          (-1 )(3 )
    4     0          -5
    5     0          (-2 )(5 )
    6     0          ((-6 )(1 ))(2 )
    7     0          6

    1        rpp -70 -20 -25 25 -25 25
    2        rpp 20 70 -25 25 -25 25
    3        rpp -60 -30 -20 20 -20 20
    4        rpp -50 -40 -15 15 -15 15
    5         sx 45 20
    6         so 95
```

B.4 Constraints on CAD "Bodies" for Converting (kf=2) a 3D Skin Representation to an MCNP Input File

Not every *.sat file can be converted to a MCNP input file. The constraints on the creation of the CAD geometry for a successful conversion are consistent with typical MCNP input files; i.e., all of space internal to the outside world must be defined and even though MCNP will treat very complicated geometries, each local cell (CAD body) should be reasonably simple for rapid random walk sampling and flexibility in the tallies.

This section considers the (kf=2) conversion, where the *.sat file must be created with only the outside "perimeter" of each body defined. Following are the constraints on this modeling approach that will typically ensure that the conversion to a MCNP input file will work.

B.4.1 Perimeter Modeling Constraints (kf=2)

1) Each CAD surface must be expressed as a quadratic equation in x, y, and z; i.e., plane, sphere, cylinder, cone, ellipse, and a general quadratic is also allowed for surfaces such as a slanted cylinder. Other surfaces such as splines are not allowed. One fourth order surface is also allowed --- the torus but its axis must be parallel to an x, y, or z axis.

2) Only the outer perimeter of each CAD body is defined. This outer perimeter can consist of any number of surfaces joined together for the shell, but these interior angles where the surfaces join must all be less than 180 degrees (concave). CAD unions of such perimeters are typically not allowed for the description of a body unless the global outer perimeter after the union has all angles less than 180 degrees. CAD subtractions are not allowed. Curved surfaces are allowed, but the curvature must be inward with respect to the body.

3) Uniqueness of space in the conversion is guaranteed by the requirement that each CAD body "a" is related to any other CAD body "b" as follows: either (3a) "a" and "b" do not have any common interior points [such as a box adjacent to a box]; or (3b) all points inside "a" are also inside "b" [such as a small box "a" inside a large box "b"]; or (3c) all points inside "b" are also inside "a" [such as the reverse with a small box "b" inside a large box "a"]. The above does allow for a common surface for bodies "a" and "b".

4) The CAD model should typically include a large box or sphere as the outermost boundary of the model. The conversion will determine a MCNP cell beyond this boundary as the "outside world" with an MCNP importance of zero.

The conversion will use the uniqueness requirement of item #3 to determine any inner perimeters of each cell. For example if one body is a large box and there are two bodies defining smaller side-by-side boxes that are inside the large box, then the cell of the large box after the conversion will contain the inner perimeter of each of the two smaller boxes (with appropriate senses and unions) even though in the CAD creation only the outer perimeter of each box completely defined the body.

B.4.2 Illustration of Non-Uniqueness

Figures B-2 to B-5 illustrate the non-uniqueness represented by the simple statement that two bodies are formed from the MCNP macrobody surfaces 1 and 2, where surface 1 is a rectangle long in y and surface 2 is a rectangle long in x. Here we are thinking of surfaces 1 and 2 as two potential CAD "bodies" or their intersections as up to 5 CAD bodies. Any one of the four figures could be made from this description, so it is not unique. A CAD user might give five bodies that represent the first five cells in Figure B-2 instead of only giving the two "bodies" that overlap. This would satisfy the all the constraints given above. The CAD modeling shown in Figure B-3 or Figure B-4 or Figure B-5 would not satisfy criteria #2.

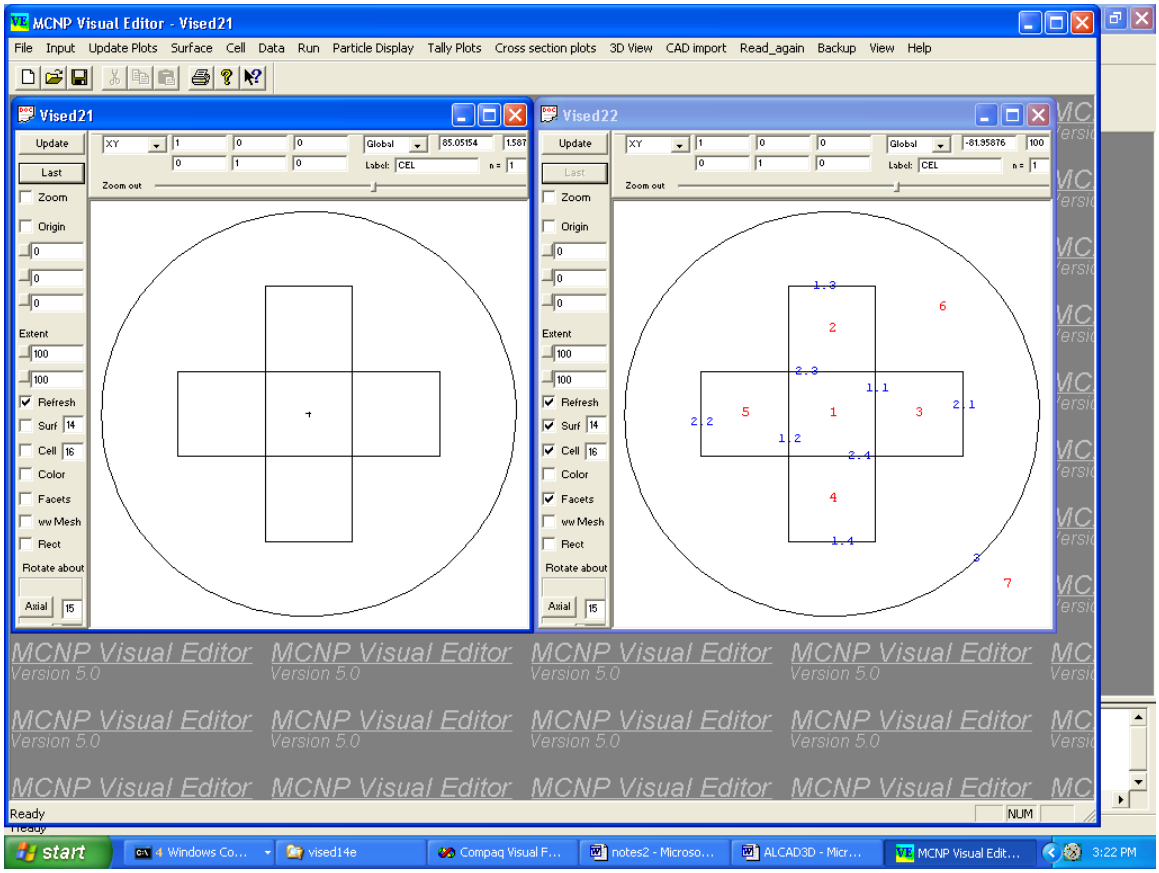

Figure B-2. MCNP Example Input File to Illustrate Non-Uniqueness, Where this Shows All Possible Cells

MCNP example input file to illustrate:

```
example for 3-D cad in MCNP format that is ill defined, icad2
    1     0          -1.1 -2.3 -1.2 -2.4 -1.6 -1.5
    2     0          2.3 -1.1 -1.3 -1.2 -1.6 -1.5
    3     0          1.1 -2.4 -2.1 -2.3 -1.6 -1.5
    4     0          2.4 -1.1 -1.4 -1.2 -1.6 -1.5
    5     0          1.2 -2.4 -2.2 -2.3 -1.6 -1.5
    6     0          ((-3 )(1 ))(2 )
    7     0          3

    1       rpp -20 20 -60 60 -40 40
    2       rpp -60 60 -20 20 -40 40
    3        so 95
```

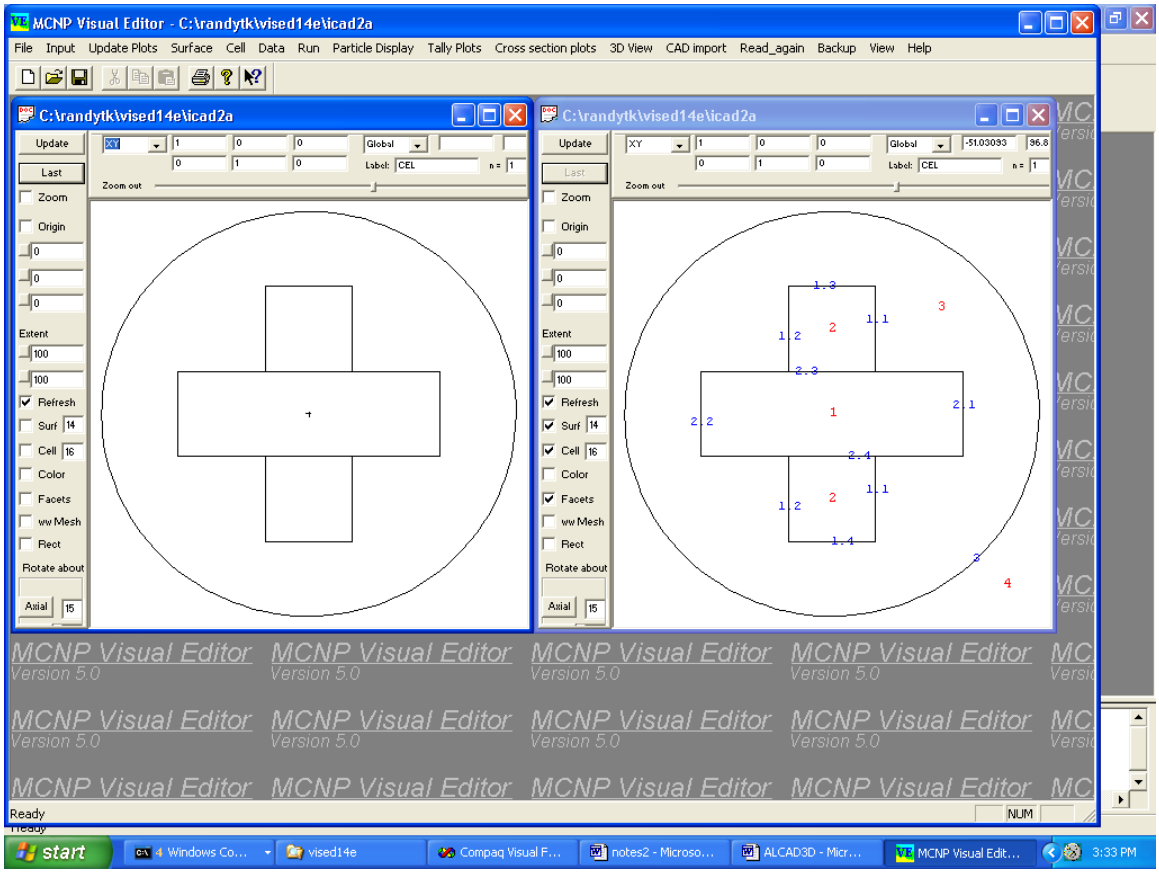

Figure B-3. MCNP Example Input File to Illustrate Non-Uniqueness, One Major Cell in x and One Split Cell in y

MCNP example input file to illustrate:

```
example for 3-D cad in MCNP format that is ill defined, icad2a
    1     0          -2
    2     0          -1 2
    3     0          ((-3 )(1 ))(2 )
    4     0          3

    1        rpp -20 20 -60 60 -40 40
    2        rpp -60 60 -20 20 -40 40
    3         so 95
```

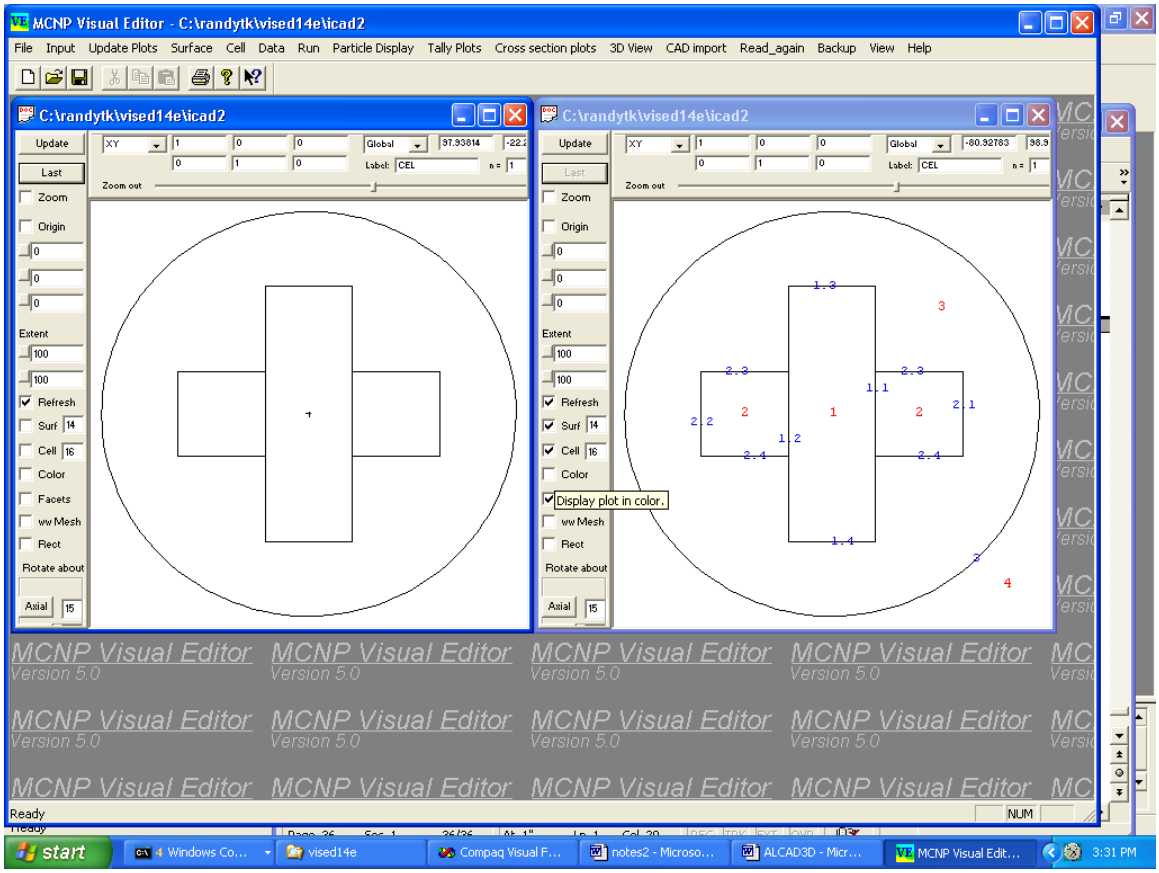

Figure B-4. MCNP Example Input File to Illustrate Non-Uniqueness, One Major Cell in y and One Split Cell in x

MCNP example input file to illustrate:

```
example for 3-D cad in MCNP format that is ill defined, icad2b
    1     0          -1
    2     0          -2 1
    3     0          ((-3 )(1 ))(2 )
    4     0          3

    1        rpp -20 20 -60 60 -40 40
    2        rpp -60 60 -20 20 -40 40
    3         so 95
```

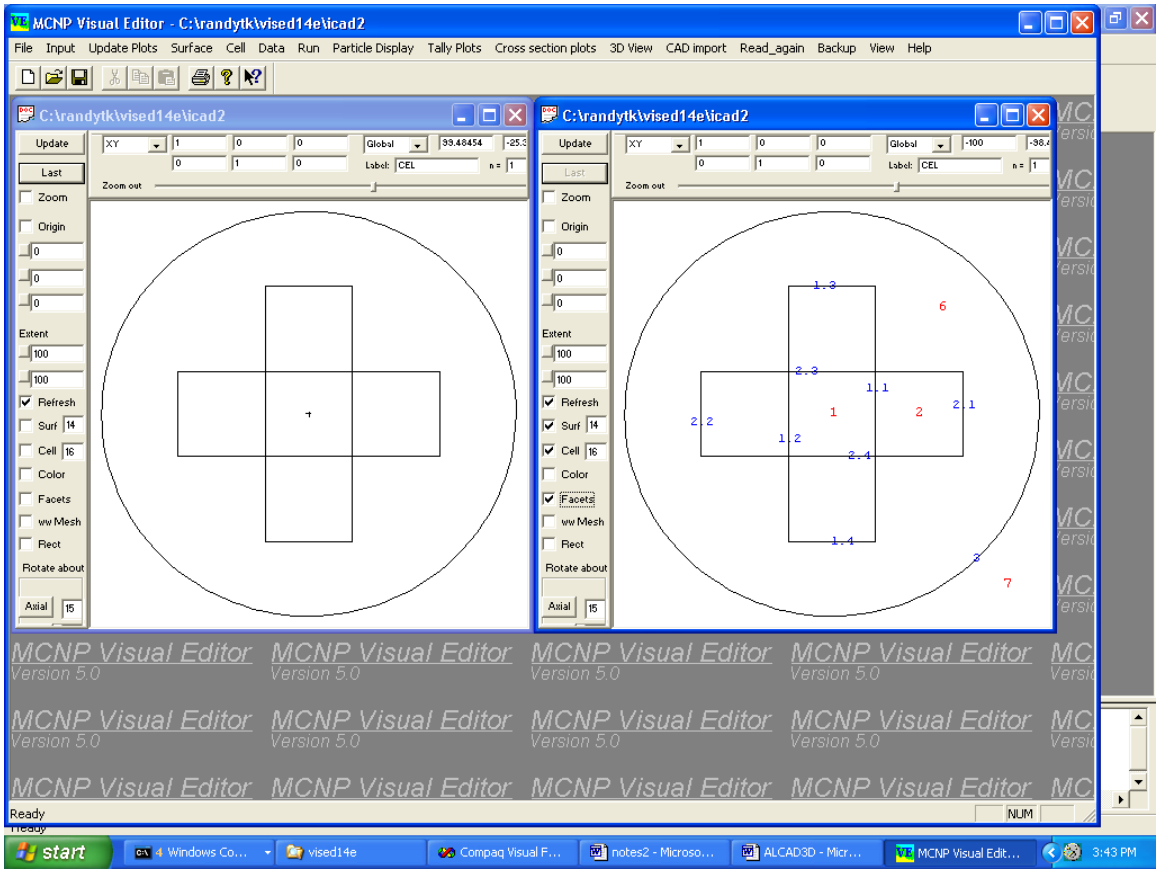

Figure B-5. MCNP Example Input File to Illustrate Non-Uniqueness, One Center Cell + Disjoint Cell that Includes the Remaining Volume

MCNP example input file to illustrate:

```
example for 3-D cad in MCNP format that is ill defined, icad2
    1     0          -1.1 -2.3 -1.2 -2.4 -1.6 -1.5
    2     0          (-1 :-2 ) #1
    6     0          ((-3 )(1 ))(2 )
    7     0          3

    1       rpp -20 20 -60 60 -40 40
    2       rpp -60 60 -20 20 -40 40
    3        so 95
```

B.5 CAD Geometry Constraints On "Bodies" (kf=4) to Allow Conversion (kf=4) of CAD Solid Modeling Representation to an MCNP Input File

Not every *.sat file can be converted to a MCNP input file. The constraints on the creation of the CAD geometry for a successful conversion are consistent with typical MCNP input files; i.e., all of space internal to the outside world must be defined and even though MCNP will treat very complicated geometries, each local cell (CAD body) should be reasonably simple for rapid random walk sampling and flexibility in the tallies.

This section considers the (kf=4) conversion, where the *.sat file must be created with CAD solid modeling and all space interior to a large outside perimeter must be defined. Following are the constraints on this modeling approach that will typically ensure that the conversion to a MCNP input file will work.

B.5.1 Solid Modeling Constraints (kf=4)

1) Each CAD surface must be expressed as a general quadratic in x, y, and z; i.e., plane, sphere, cylinder (only with axis parallel to a major axis is currently allowed for the conversion), and cone (only with axis parallel to a major axis). Currently ellipses and other general quadratics that do not include the above are not allowed.

2) The outer perimeter of each CAD body can consist of any number of surfaces joined together for the shell, but these interior angles where the surfaces join must all be less than 180 degrees (concave). CAD unions are typically not allowed for the description of a body for a successful conversion unless the global outer perimeter after the union has all angles less than 180 degrees. CAD subtraction is allowed to cut out an interior portion, but the interior portion being subtracted must also have all interior angles where the surfaces join with angles less than 180 degrees (concave). A subtraction example would be a small box interior to a large box, where the small box is subtracted from the large box to form the body between the two boxes. Multiple CAD subtractions to form the body are allowed (each portion subtracted must be completely inside the outside perimeter although they can share a common surface), but no more than two side-by-side subtractions (where the subtractions share a common surface) are allowed (so you cannot subtract a box immediately adjacent to a box immediately adjacent to another box). If such complex subtractions are needed, you can avoid them by splitting the original large body from which they are being subtracted into two or more smaller bodies. This is consistent with a good MCNP geometry philosophy of keeping individual cells relatively simple.

3) The CAD solid modeling must define all of space (not double defined anywhere) inside a large box, or cylinder, or sphere, where the region beyond this large box, or cylinder, or sphere in the conversion will be defined as a MCNP cell for the "outside world" cell; i.e., with an importance of zero.

Conversion of solid modeling test cases satisfying the above criteria have been demonstrated with a number of cases when only planes are involved. Fewer tests with curved surfaces have

been converted, but the conversion has been working satisfactorily. The conversion does not deal with the occasional need for ambiguity surfaces when curved surfaces are involved, but the need for them is typically evident because of the presence of dashed lines in the MCNP plots after the conversion.

B.6 Present 3D Implementation in The Visual Editor

The upgrade patch file for Version 4C2 utilizes the keyword "vecad" for the CAD-specific additions, which are both for 2D and 3D –--Version 5 utilizes the keyword "VECAD". For the 3D upgrade, the following variables have been included in common and are utilized as follows:

Jfcadc(ic)=flag for cell ic, =-1 if next to outside world, =0 initially.
Jncadc(ic)=number integers to define CAD body on MCNP cell card ic.
Jxcadn(ic)=number (x,y,z) triplets inside cell ic.
Jxcadl(ic)=beginning location of (x,y,z) triplets in cxyzcad() for cell ic.
Jycadn(I,1)=is the cell number that will have cell jycadn(I,2) put inside of it.
Ncadch=maximum value of "i" in jycadn(I,1) or jycadn(I,2).
cxyzcad(1),cxyzcad(2),cxyzcad(3)=first (x,y,z) triplet, etc.

The CAD creation of the *.sat file must follow some specific constraints, where the "kf=2" outer perimeter modeling CAD constraints were summarized in Section 6 and the "kf=4" solid modeling CAD constraints were summarized in Section 5.

B.7 FORTRAN Subroutines for the 3D Conversion

The following FORTRAN subroutines have been included for the 3D conversion, where these are given roughly in the order that they are called. Each subroutine is briefly described here, where the discussion is based on 4C2 but is basically the same for Version 5 except it uses FORTRAN 90. Additional information is available in the comment statements of the subroutine.

1) Subroutine cadcel3(kf)

This is the FORTRAN subroutine that is called by the C for the 3D conversion of the *.sat file to a MCNP input file. Originally this subroutine utilized a "kf=1" or "kf=8" option where the FORTRAN reads the *.sat file, but the mode now is with other kf options with the C reading the *.sat file. However, the "kf=1" or "kf=8" options have been left in the subroutine in case some of these features are desired in the future, but these are not discussed in detail here.

This subroutine is typically called with kf=2 [CAD perimeter modeling] or kf=4 [CAD solid modeling], but may sometimes be called with kf=3 [only deal with surfaces in the conversion and do not do cells, where sometimes this is useful if the cell conversion does not work].

For "kf=2,3,or 4", the FORTRAN subroutine cadrc3d is called, which then asks the C to read the entire *.sat file and give the surface/body information back to the FORTRAN. After a return from cadrc3d, subroutine cadcel3 then calls subroutine cadids to eliminate any identical surfaces.

If kf is 2, it then calls cadflag(nf) to set flags for cells next to the outside world. Then beginning at statement 10 in subroutine cadcel3 for kf=2, it determines the inner perimeters of cells. [It looks for a completely interior cell and the cell surrounding such a cell and puts these cell numbers in the arrays jycadn(I,1) and jycadn(I,2). Then it looks for the next surrounding cell, etc, as in the algorithm discussion of Section II. It also sets flags as cells are found. Finally, it uses the jycadn(I,1) and jycadn(I,2) arrays in a call of subroutine cadnot in the "do 200" loop to include the outer perimeter of cell jycadn(I,2) as an inner perimeter of cell jycadn(I,1) after doing a #.]. Then after dealing with the outside world cell in the "do 210" loop, it calls subroutine cadinp3 to give the cell cards to the C at statement 220.

If kf is 4, subroutine cadrc3d does all the work of determining the cell cards with CAD solid modeling and then cadcel3 immediately jumps to statement 220 to give the cell cards to the C.

2) Subroutine cadrc3d(kf)

This subroutine has a loop starting at statement 10 that increments the cell number (mxa) in preparation for obtaining the information for that cell/body from the C. Then there is a loop starting at statement 20 that increments the surface numbers for this cell, where "ja" is the local surface number for the cell and mxj is the global surface number. Then for each surface of the cell/body, there is a call of ccadsur3() asking the C to give it the surface and its associated information including sense for the cell.

If the C passes back "jq" as positive following the call of ccadsur3, the C passes the FORTRAN information for surface jq including the CAD surface coefficients that are translated by the FORTRAN to a MCNP surface, the (x,y,z) vertices for this surface and shell, and the other surface numbers that intersect this surface. Some of this information is stored in temporary arrays [see comment statements regarding arrays at the beginning of this subroutine], such as the MCNP surface number along with its sense is put in jd(). The MCNP surface coefficients are put in the scf() array [see statements from before statement 75 through statement 130 for the different MCNP surface types]. Then at statement 150 information about the surface is written to the outp file and the FORTRAN calls cnew(3,0) to give the MCNP surface to the C after statement 170, and then returns to statement 20 to obtain the next surface from the C.

If the C passes back jq as "-1" following the call of ccadsur3, the current CAD body is complete and C passes the FORTRAN "nv" x,y,z points that are in the body which the FORTRAN puts in the array cxyzcad() [this is particularly important for "kf=2"]. The FORTRAN then jumps to statement 190, where it then jumps to statement 405 to make this outer perimeter shell of the body the outer perimeter of an MCNP cell if "kf=2". If "kf=4" the statements from 200 to 400 determine whether the shell is an outer perimeter or an inner perimeter [with MCNP unions by setting the flag array jf()], and then goes to statement 405 to form the corresponding portion of the MCNP cell. [This is quite complicated since its trying to determine both the outer perimeter and the inner perimeters for the MCNP cell. Particularly important is the "do 730" loop that checks if there are two side-by-side cuts. There are additional complications if curved surfaces are involved.] For either "kf=2" or "kf=4" it eventually reaches statement 435 where it puts the cell mxa in memory and if "kf=4" it returns to statement 200 to do the same thing for the next shell of the body. In either case, it eventually reaches statement 450 where it writes out

information about the MCNP cell to the outp file and returns to statement 10 for the next surface of the next body. Prior to this return to statement 10 with "kf=2", the FORTRAN calls cadrp(mp) to use ray tracing to determine points near the perimeter of the cell.

If the C passes back jq as "-2" following the call of ccadsur3, all bodies have been treated and the FORTRAN jumps to statement 500 and tries to include the outside world cell if kf=4.

Additional notes on coding from statements 200 to 400 for kf=4. The loop beginning at statement 200 deals with a shell "is" one at a time to determine if this shell has an outer perimeter and any inner perimeters from this shell. The array ks(j) contains the shell number of surface j so it only considers those that have shell number "is" in this loop. It initializes a number of flags in the "do 198" loop and utilizes/changes these flags as pertinent. The variable "ka" is usually "1", but may be two when it is dealing with the second of two side-by-side cuts. Curved surfaces introduce additional complications that are dealt with throughout. Initially it looks for an outer perimeter in "do 205" and "do 210" and sets the flag to "-999" for such surfaces. The "do 270" loop looks for the potential beginning of a group of unions for those surfaces of the shell that are candidates with an appropriate flag. It looks for a pair of surfaces that cross, j and jt, which have the appropriate flag [that has not been used and these surfaces are not part of the outer perimeter]. When it finds such a pair of surfaces, it sets the flags to "-nf" at statement 255, where "nf" is incremented for each union string. Then it jumps to loop 280 where the "do 372" loop is like the "do 270" loop except now it is looking for a surface that crosses one of the already found surfaces of the string and sets its flag at statement 355. Then after finding all such surfaces, it deals with the union string at statement 390. At the "do 740" it looks for a potential splitting surface when there are two side-by-side cuts. If such a splitting surface exists, it sets "jz" to this splitting surface after statement 750 in which case all of the logic in the "do 270" and "do 372" is more complicated when "jz" is not zero since it must only accept vertices that are on the correct sense side of this splitting surface [one side for the string with ka=1 and the opposite side for the string with ka=2]. The strings are put in the icadtv() array, with unions as necessary, in the "do 394" loop.

3) Subroutine cadr3d

This subroutine is obsolete since it is for the FORTRAN, rather then the C, to read the *.sat file. It is called with "kf=1" to read the *.sat input file and put the information for each body in FORTRAN memory. The actual reading of the file is with a call of subroutine cadrr(…). It gives each surface of a CAD body to the "C" with a call cnew(3,0), and gives the body to the FORTRAN with a call cadbody3(…). This routine is typically not called, but "kf=2" is used instead for the "C" to read the *.sat file.

4) Subroutine cadrr(kt,nb,nc,nd,ni,ns,je,ii,aa)

This subroutine is also obsolete. It reads a line of the *.sat file when called by cadr3d with "kf=1" and returns the information in the arguments. Here kt is a line type identifier [body,lump,shell,face,surface], nb is the body number [set to "-1" when no more bodies to read], nc is the line count relative to line 4 being "0", nd is the number of lines to move [negative or positive] from the previous read, ni are the number of entries just beyond each $ that are passed

in the array ii(), ns is a sense control based on “forward” or “reversed” on the “face” card, je is an error flag, and the arrays ii() and aa() contain information from the line read.

5) Subroutine cadbody3(ic,nj,nt,nw,jf)

This subroutine is also obsolete since it is called by cadr3c. It puts the outer perimeter (CAD body) of cell ic in FORTRAN memory with “kf=1, where nj is the number of MCNP surfaces to be given to the FORTRAN for this CAD body [through the call of cadr3d when reading the *.sat file with FORTRAN], nt is the number of (x,y,z) triplets of points being passed to the FORTRAN for this CAD body, nw is the number of intergers to define the perimeter of the cell for the cell card, and jf is the flag for this cell. Cadbody3 will be called by the FORTRAN for each cell as the FORTRAN is reading the CAD *.sat file in subroutine cadr3d. The same information is set in subroutine cadrc3d when the C is reading the *.sat file and giving the information to the FORTRAN.

6) Subroutine cadflag(nf)

This subroutine examines the outer perimeter of each MCNP cell with “kf=2” whose flag has not been set to see if it has no cells bounding, in which case sets its flag to “-1” to denote an outer cell next to the outside world. This subroutine would not be needed if the user set all such flags.

7) Subroutine cadnot(i1,i2)

This subroutine with “kf=2” includes the outer perimeter of cell i2 as an inner perimeter of cell i1 after doing a #.

8) Subroutine cadinp3

This subroutine gives each cell card to the “C” with a call of cnew(4,0), where the information is passed in the icvis() array.

9) Subroutine cadids

This subroutine eliminates each surface that is identical to a previous “master” surface. It changes to the master surface on each cell card with sense change if the senses are opposite.

10) Subroutine cadrp(mf)

For “kf=2” cadrp(mp) uses ray tracing to determine points near the perimeter of the cell by using points within the cell as a beginning point for a ray or sequence of rays.