M&C 2001 Salt Lake City, Utah, USA, September 2001.

INTERACTIVE 3D DISPLAY OF MCNP GEOMETRY MODELS

Kenneth A. Van Riper
White Rock Science
P. O. Box 4729, Los Alamos, NM 87544
kvr@rt66.com

Keywords: Graphics MCNP

ABSTRACT

We describe a new program for the display of three-dimensional (3D) geometry models. The image can be rotated, moved, and scaled with the computer mouse. It is based on the OpenGL library and can thus take advantage of hardware graphics acceleration prevalent in modern video cards. We describe a cell-based polygonalization method that quickly converts an MCNP (Briesmeister, 2000) model to the form required for display with OpenGL. Features for manipulation and interrogation of the 3D image include rotation, zooming, and panning with the mouse or keys, clip planes, and cell selection. In the fly through mode, the 3D viewpoint follows the cursor position in two-dimensional cuts through the model. Up to eight lights are available. Colors can be set by cell or material. Tracks of Monte Carlo particle histories can be shown in the 3D window together with the geometry.

1. INTRODUCTION

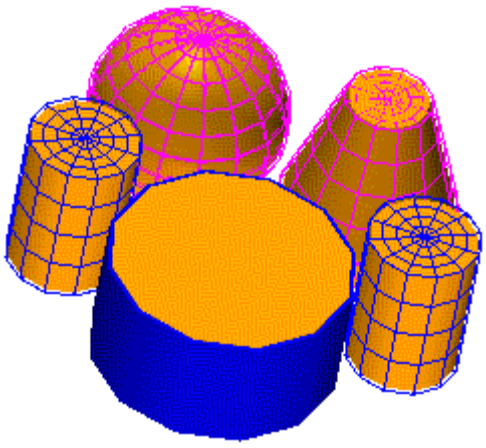

Fig 1. A Moritz 3D display.

Radiation transport codes require a description of the geometry in which the transport occurs. Three-dimensional geometry models, such as used by MCNP, can be difficult to construct for complicated objects. The ability to view the full three-

dimensional model aids the verification, understanding, and debugging of the model. Ray tracing programs, such as Sabrina (Van Riper, 2000), produce a static 3D image. Although the model can be viewed from any direction and with any desired set of cells made visible, each new image requires seconds to minutes to compute and display. A more interactive program would permit the user to manipulate—rotate, scale, and move—the image in response to computer mouse movements and keystrokes. The Justine (Lee, et al., 1994) program (and several predecessors) from Los Alamos National Laboratory offered this capability. We are currently developing the Moritz program that also offers interactive 3D graphics. We are, at first concentrating on the display of MCNP geometry; we intend to offer support for other transport codes in the future.

Figure 1 shows a typical 3D image. (Because of size limitations dictated by the format of these Proceedings, the figures in this paper are shown at reduced resolution. One reduction is the number of colors, resulting in mottled appearance on some surfaces caused by dithering.)

The Moritz 3D graphics display is built on the OpenGL application interface. OpenGL is a graphics standard that has been widely adopted for 3D computer graphics. In particular, it is widely used for computer games, which in turn has spurred video card manufacturers to include support for OpenGL, in the form of hardware graphics acceleration. As a result, 3D graphics performance on even moderately priced current personal computers is very good. Given a set of polygons and the viewpoint, the OpenGL library processes the transformations and hidden surface removals required to produce an image on the screen. Justine, in contrast, performed these tasks in software, bypassing any advantage that could be gained by hardware acceleration.

2. OVERVIEW OF THE MORITZ PROGRAM

In addition to the 3D graphics display, Moritz shows three two-dimensional (2D) views in cuts perpendicular to the X, Y, and Z axes. The 2D windows show the intersections of surfaces with the cut. Cells are shown by filling with color and/or displaying the portions of surfaces that bound them. Geometry editing features include the creation and modification of surfaces and cells, either graphically with the mouse or with dialogs. Surfaces can be positioned explicitly or by entering their distance from another surface. Images such as blueprints can be imported as the background for the 2D drawings to serve as a guide for geometry creation. Cells can be defined by mouse clicks. A number of settings allow for quick cell creation. New cells and modifications to existing cells or the surfaces that bound them are immediately displayed in the 3D window. Materials, importances, and universes filling a lattice can be assigned by clicking on a cell or lattice element.

Moritz reads and writes MCNP input files. The current version ignores data cards other than transformation cards. It can also read Sabrina command files. Commands that have meaning for Moritz, such as setting material and cell colors and visibilities, are acted on; the rest are silently ignored. We have also introduced a number of Moritz specific commands. These latter commands do not conflict with those understood by Sabrina so that both programs can use the same input file. Moritz writes a Sabrina input file that results in a ray-traced image of the view in the 3D window. Sabrina can be started and sent this file from Moritz. Moritz also writes a larger command file saving the entire state

of the program including, in addition to the geometry, user preferences and settings and items such as window positions.

Moritz uses the Windows *multiple document interface*. It consists of a container window and several child windows. The main *menu* spans the top of the container window just below the title bar. The *toolbar* is just below the menu. It contains buttons that invoke some action. The *status bar* is at the bottom of the container. The leftmost panel gives a description of the function of a selected menu item or feedback on what Moritz is doing. The next panel displays an expected user action or some of Moritz's settings, such as whether geometry testing in effect. The program is controlled by mouse activity in the graphics windows, keystrokes, menus, and dialogs. In addition to the main program menu, a right click in any of the graphics windows displays a *context menu* containing items specific to the cursor position. Most dialogs are grouped into tabbed *property sheets*, each tab of which is known as a *property page*.

Several windows show textual information. A *transcript* window displays messages from the program. *File listing* windows echo MCNP and command files read in. Various types of information go to the *report* window, such as previews of files that Moritz writes. The *information* window has several tabs, one of which is active at a time. The *position* tab gives the coordinates at the cursor position in 2D windows. The *cell* and *surface* tabs show properties of a selected item. The contents of the *hint* tab reflect the program state and active window, giving prompts about expected and possible user actions. The user has control over the appearance of all windows, including size, position, background color, and font. The contents of all windows can be printed.

Moritz is being developed on Windows and will be available at first for that platform. It will be available from White Rock Science. The availability of a Unix version will depend on user interest.

3. CONSTRUCTION OF THE 3D DISPLAY MODEL

Interactive 3D display requires that the model be described as a collection of polygons. All vertices of a polygon must be coplanar. The number of polygons depends on the complexity and shape of an object. A box, for example, can be represented by a single polygon for each of its six faces, while a sphere requires about 200 polygons for a smooth appearance.

MCNP geometry is defined in terms of cells bounded by quadric surfaces. Constructing polygonal representations from these descriptions is a complex task given the great variety of possible surfaces and their intersections. In the Justine program, each surface is polygonalized, followed by an intricate polygon intersection algorithm to resolve the cell descriptions. Since each polygon must be tested against each other polygon in the surfaces describing a cell, the computation time increases as approximately the square of the number of polygons possibly bounding a cell. Complex models require many minutes to be polygonalized.

In Moritz, the polygonalization takes place for each cell, rather than intersecting pre-polygonalized surfaces. The cell description is analyzed to determine if the cell can be classified as one or more of a number of known *shapes* for which the code has a

polygonalization algorithm. Moritz can process and display complicated models with hundreds of cells in seconds.

3.1 Surface Analysis

Moritz recognizes all MCNP surfaces, including planes perpendicular to a coordinate axis (PX, PY, PZ), general planes, spheres, cylinders and cones centered on or parallel to an axis, tori with major axis parallel to an axis, and special (SQ) and general (GQ) quadratics. Moritz treats several classes of SQ surfaces as distinct types: elliptical cylinders centered on or parallel to an axis and ellipsoids with axes parallel to the coordinate axes. Elliptical cones will be added. GQ surfaces are analyzed to see if they are non-aligned circular or elliptical cylinders, cones, or ellipsoids. The MCNP4C macrobody surfaces are supported.

3.2 3D Shapes

The shapes that Moritz can polygonalize are shown in Figures 2 through 9. Each shape is shown as a solid and a wireframe. Hidden surfaces are not removed in the wireframe view.

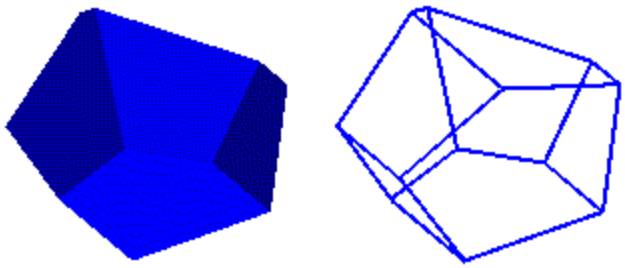

Fig. 2 A *Polyhedron* is a shape bounded by up to 20 plane surfaces.

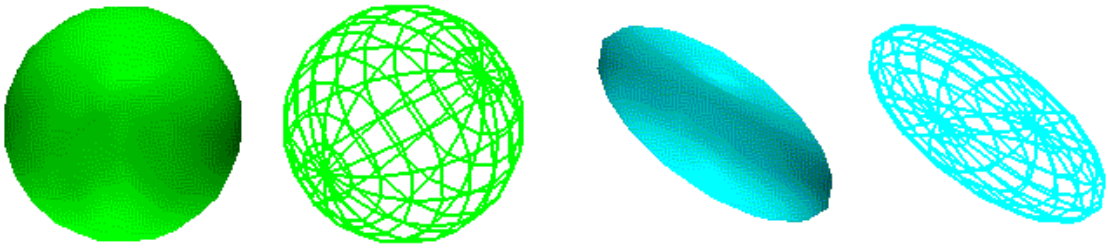

Fig. 3 A *Sphere* (left) and *Ellipsoid*, **(**right), are shapes inside a spherical or ellipsoid surface.

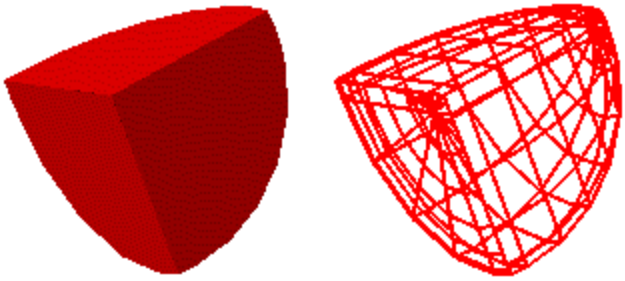

Fig. 4 A *Flattened Ellipsoid*.

A Flattened Ellipsoid is a shape inside a sphere or ellipsoid and bounded by up to six PX, PY, or PZ planes, but by not more than 2 planes of the same type.

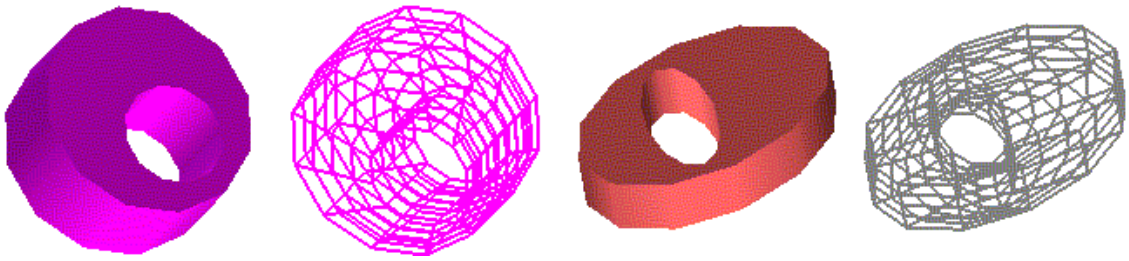

Fig. 5 A *Right* Circular *Cylinder* (*RCC*, left) and a *Right Elliptical Cylinder* (*ERCC*, right).

An RCC shape is inside a circular cylinder or between two circular cylinders parallel to the same axis and between two planes perpendicular to the cylinder axis. The inner cylinder must lie completely inside the outer cylinder. The ERCC shape is similar, but one or both of the cylinders is elliptical. If the cylinders intersect, the shape is usually successfully polygonalized as a general cylinder.

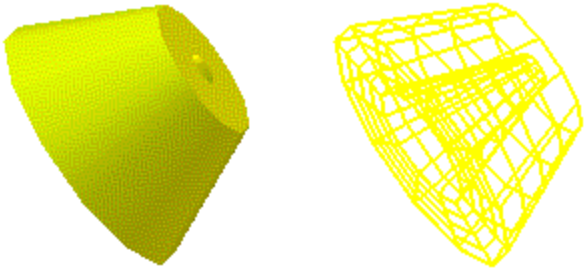

Fig. 6 A *Truncated Right Cone*.

A Truncated Right Cone is a shape inside a circular cone or between two circular cones parallel to the same axis and between two planes perpendicular to the cone axis. The inner cone must lie completely inside the outer cone. The polygon representation will be incorrect if a cone origin lies between the bound planes; we will add a provision to handle this case correctly in the future.

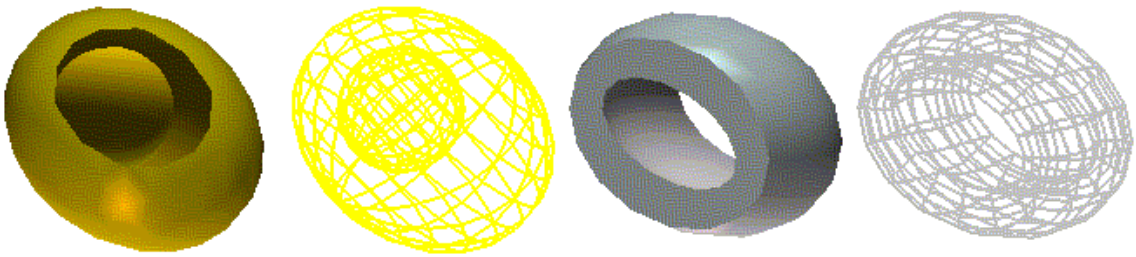

Fig. 7 A *Shell* (left) and a *Flattened Shell* (right).

A Shell is a shape between two spheres, between two ellipsoids, or between a sphere and an ellipsoid. The inner surface must be contained completely within the outer surface. In the solid representation on the left, a cutter plane is used to expose the inner surface. A Flattened Shell is a shape between two spheres or ellipsoids or between a sphere and an ellipsoid and bounded by 1 or 2 parallel PX, PY, or PZ planes.

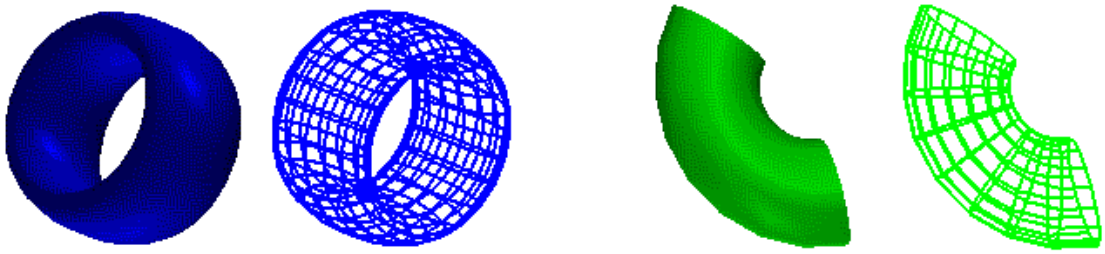

Fig. 8 A *Torus* (left) and a *Partial Torus* (right).

A torus is a shape inside a torus surface. A Partial Torus lies inside a torus surface and on one side of one or two planes that are parallel to the torus major axis and pass through or near the center of the torus.

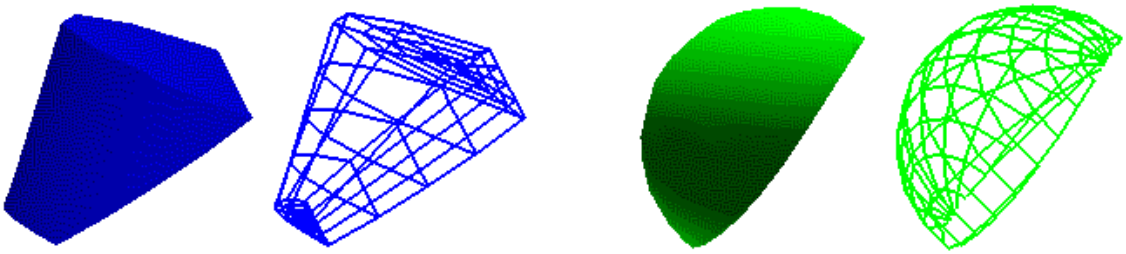

Fig. 9 A *General Cylinder* (left) and a *General Ellipsoid* (right).

The General Cylinder is bounded by at least one cylindrical or conical surface and two planes perpendicular to the axis of the cylinder or cone and that cannot be classified as one of the above shapes. The curved surfaces need not be aligned with a coordinate axis. If the intersection of the shape with either of the bounding planes is convex, that surface is polygonalized. The General Ellipsoid is a shape inside at least one ellipsoid or sphere surface and that cannot be classified as one of the above shapes. Only the ellipsoid surface of the shape is represented in the present version.

3.3 Cell Analysis

Moritz analyzes cell descriptions to find cells, or parts of a cell, that can be polygonalized as a supported shape. A cell *part* is either a simple cell (without unions or complements) or the part of a cell description between union operators.

The analysis first counts the number of union and complement operators and parentheses after redundant parentheses have been removed but before complements have been expanded. If the description consists of only signed surfaces, the cell is a *simple* cell and the shape analysis continues.

If the number of non-surface items is greater than the number of surfaces in the description, Moritz assumes the cell is complicated because a number of items are being excluded in the interior and that the first surfaces listed in the description could describe the outer surfaces of the cell. The program then attempts to see if the portion of the cell description of this *basket cell* consisting of signed surfaces before the first parenthesis, union, or complement can be characterized as a known shape. If so, that shape is given a default 3D style as wireframe. If one or more complement operators are found, the complements are discarded and the cell is treated as a basket. The wireframe cells in Figure 1 are basket cells.

The method of ignoring complements works satisfactory when the complements exclude cells in the interior of another. In that case, assuming the excluded objects themselves can be polygonalized, there is no need to represent the inner surfaces of the container cell. If this is the desired result, it is best to use complement operators in the cell description rather than the equivalent description in terms of surfaces and unions, in which case Moritz may be smart enough to polygonalize the inner surfaces. The method will not work if the complement is used to exclude a partially intersecting cell. In our experience with input files from hundreds of MCNP users, we have rarely seen this latter use of the complement operator.

If the cell is not found to be a simple or basket cell and the description contains unions and parentheses but no complements, Moritz attempts to remove the parentheses by moving the surfaces outside of the parentheses to each part of a union inside the parentheses. If the algorithm succeeds, which is usually but not always the case, the result is a cell description consisting of sequences of signed surfaces between unions, such as

1 –2 3 –4 : 5 –6 7 –8 : 9 –10 11 –12

Each sequence of surface numbers constitutes a *part*. The shape analysis is performed on each part of one of these *union* cells.

If the analysis does not find a supported shape and if all surfaces of a part carry the same transformation, Moritz repeats the analysis on the part using the original untransformed surfaces. If the analysis is successful, the transformation is applied to the polygons when they are drawn. We have encountered several cases where all surfaces of a part except a plane carry the same transformation. To permit the polygonalization to proceed, we will introduce a new plane with the transformation.

3.4 Unnecessary Surfaces

An unnecessary surface is an entry in a cell description that is less restrictive than another surface in the description. For example, when specifying the volume below plane 1 (x = 10) and below plane 2 (x = 0), plane 1 is unnecessary. Unnecessary surfaces can cause a part to not be recognized as a known shape or can lead to a bad polygonalization. Because not all users write perfect input files, Moritz checks for unnecessary planes, spheres, ellipsoids, and tori. When one is found, a message is written, and the surface is not used for the polygonalization analysis.

3.5 Alternate Cell Descriptions

Sometimes cell descriptions include one or more surfaces to avoid a very small overlaps with other cells and these extra surfaces prevent the cell from being recognized as a known shape. Such small overlaps are not a problem for the 3D display. In other cases, a cell description can be changed into a recognizable shape by eliminating redundant interior surfaces or decreasing the number of non-surface operators used. To accommodate such changes, an *alternate cell description* can be defined.

3.6 Viewing Cell Descriptions

A dialog is available to view and change cell descriptions. Three descriptions are shown. The *Original Cell Description* is the description exactly as read from an input file without complement operators expanded. It cannot be changed. The others are the description with complements expanded and redundant parentheses removed and the optional alternate description. Double clicking on a surface in a description list highlights the surface in the 2D windows and shows the surface type above the description list. If the user holds down the control key, previously selected surfaces remain highlighted. Otherwise, only one surface at a time is highlighted. This feature is useful when trying to understand a cell description.

3.7 Overlapping Polygons

When two 3D polygons lie in the same plane, the resulting image is not well defined and flickering occurs as the model is rotated and moved. The left panel of Figure 10 shows an example; the checkerboard-like pattern changes with the slightest movement of the image. Coplanar polygons occur often as a consequence of adjacent cells. Flickering is most evident when cut planes or other methods are used to view the interior of a model. One method of eliminating the problem is to draw common surfaces of adjacent cells once, as is done in the Justine program. Doing so, however, leads to lengthy calculations and we have not implemented such an algorithm. We may introduce a shared surface algorithm in the future where it can be done quickly in a limited set of cases.

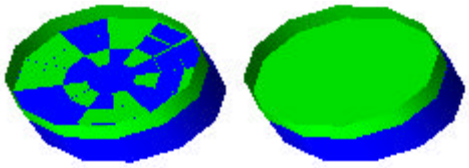

Fig 10. 2 adjacent cylinders without (left) and with (right) cell shrinking.

Polygon overlap can be reduced in Moritz by slightly shrinking the 3D shapes. Cells may be shrunk by a either a fraction of their dimensions or by an absolute value in each dimension. Shrinking by a small fraction, such as 0.005 as is used in the right panel of Figure 10, usually gives the most satisfactory results. If the shrinking fraction is too large, noticeable gaps will appear. Another use of shrinking is to introduce a visible gap to show cell boundaries; in this case, shrinking by a constant value is useful. Figure 11 shows an example of shrinking by value.

Another option to deal with overlapping polygons is to omit inner shells. When the option is in effect, the inner cylinder in RCC and ERCC shapes and the inner sphere or ellipse in Shell shapes are not drawn in 3D. The option is useful for models consisting of nested cylinders and shells. It can be applied globally or on a cell-by-cell basis.

4. VIEWING THE 3D DISPLAY

The usefulness of the 3D model depends on the user's ability to interact with and understand it. Moritz has a rich set of controls for navigating through and querying the model.

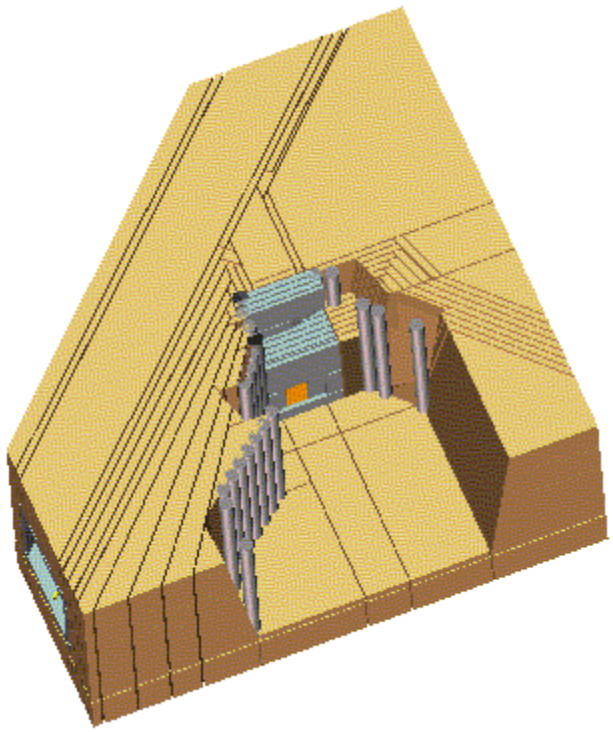

Fig 11. A model with cell divisions made visible by shrinking by value.

Once the polygons are defined, OpenGL performs a series of transformations to produce a scene. The details of projecting the 3D model onto a 2D screen are complex. We give here some brief descriptions of the process as necessary to describe the program

operation and some user accessible settings. For further information, the reader is referred to an OpenGL reference book such as Woo, et al., 1999.

4.1 Projection

There are two types of projection. In the *perspective* projection, the relative size of objects depends on their distance from the viewpoint or eye. Objects further away appear smaller. In the *orthographic* projection, objects remain at their same size independent of distance from the viewpoint. According to the OpenGL documentation, orthographic projections are useful for CAD and architectural work. The projection type can be changed in the 3D context menu and on the projection property page. The projection page contains a number of other settings that affect the scene.

For both projections, one must define a set of clipping planes that bound the scene. Any objects that lie outside of the planes are clipped. The clipping planes depend on the global bounding box of the model. How the planes are defined depends on the projection. The perspective clipping planes define a frustum with the smaller end near the viewpoint. The frustum is defined by the position of the near (relative to the viewpoint) and far planes and the opening angle. The near and far planes are set to enclose the global bounding box. Multipliers on the projection property page permit changing the near and far plane locations. The page also has a field for the opening angle (the default angle is 45°). The orthogonal clipping volume is a rectangular box that encloses the global bounding box. The projection page contains multipliers for each of the orthogonal clipping volume planes.

4.2 3D Image Motion

The 3D model can be rotated, moved, and scaled with the mouse and keys. The last submenu of the 3D context menu shows the mouse and key bindings. The bindings are also displayed in the *Hint* tab of a separate information window.

4.2.1 Rotation

Moving the cursor with the left mouse button held down rotates the model. The rotation is in the direction of the mouse motion. The rotation is based on a trackball algorithm that intersects the cursor position with an imaginary sphere surrounding the model. The user may change, on a property page, a rotation speed factor that multiplies the conversion from cursor motion to angle change. Holding down the shift, control, or alt key while rotating with the left mouse button constrains the motion to rotation about the X, Y, or Z axis, respectively. The default center of rotation is the origin (0, 0, 0) in model space. The center can be changed by clicking on the desired location in one of the 2D windows.

4.2.2 Auto Rotation

The rotation will continue on its own if the cursor is moved from the 3D window before the button is released. The initiation of auto rotation depends on the detection of mouse movement in the window border. If the cursor is moved out of the window too quickly, a motion event in the border will not be detected and the auto rotation will not begin.

Clicking the left button in the 3D window stops the auto rotation. Right button actions—context menu, pan, and zoom—suspend the rotation for the duration of the action. The auto rotation will continue when the menu is dismissed or the pan or zoom stops.

The auto rotation speed depends on how often the image is redrawn and a speed factor that determines the change in angle on each redraw. A property page contains fields for changing both the refresh rate and speed factor. The default rate is 33 milliseconds. The image could be redrawn less often than the requested refresh rate if the computer is busy elsewhere or cannot keep up with the graphics demands. The speed factor can also be changed with *Spin More Quickly* and *Spin More Slowly* items in the 3D context menu.

4.3 Look Towards

The *Look Towards* submenu of the 3D context menu contains items for looking towards the + and – X, Y, and Z axes. Selecting an item changes the orientation to the desired view.

4.4 Image Size

The image size is initially set to fit the global bounding box. It is changed by the *Zoom In* and *Zoom Out* toolbar buttons, the + and – keys on the numeric keypad, or by cursor motion with the right mouse button and control key both held down. Holding down the control key and pressing the + and – keypad keys results in a greater change in size. In the perspective projection, the viewing volume is fixed. Making the image smaller is accomplished by moving the model further from the viewpoint. If it is moved too far away, it intersects the far clipping plane and will eventually be clipped out of existence.

4.5 Panning

Panning, or moving the image around the 3D window, is accomplished with the arrow keys or by cursor motion with the right mouse button and the shift key both held down. When using the arrow keys, holding down the shift or control keys accelerates the motion.

4.6 Fly Through Mode

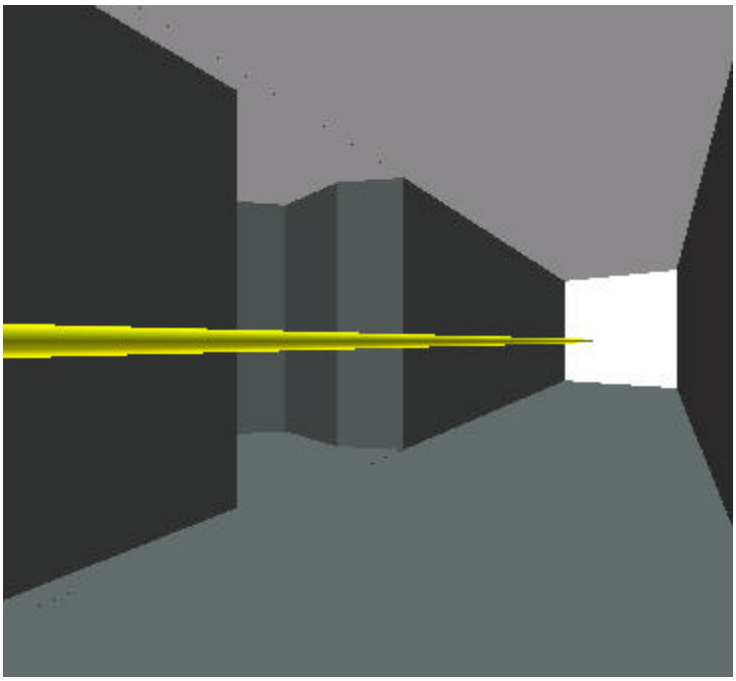

Fig 12. Flying through a model

In the *Fly Through Mode*, mouse movements in the 2D windows control the viewpoint. Figure 12 is from a fly through of an accelerator tunnel model. The mode works properly in perspective projection. It will also change the viewpoint in the orthographic projection, but not in an intuitive manner.

When the fly through mode is active, a small triangle appears at the cursor position in the active 2D window to show the viewpoint location. The view is in the direction of the opening angle of the triangle. Cursor motion with the left button held down or use of the arrow keys moves the viewpoint. Moving the cursor with the left button and control keys held down rotates the view direction. Rotation can also be accomplished with the arrow keys with the control key held down.

The vertical direction in the 3D window is rotated with the left mouse and either the shift or alt key held down. An arrow points in the up direction. When the shift key is used, the direction is constrained to multiples of 10°; there is no constraint when the alt key is used. The up direction and view direction should not be the same. If they are, the 3D window is blank.

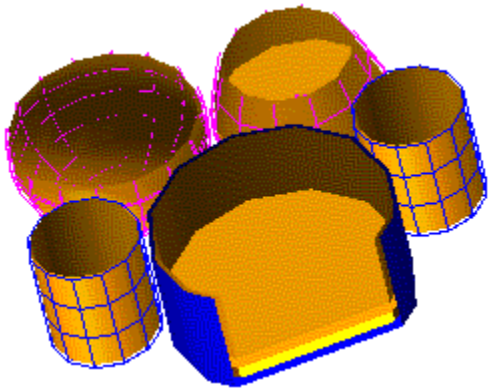

Fig 13. The model of Fig. 1 with 2 clip planes active.

4.7 Clip Planes

Clip planes (not to be confused with the planes used in the clipping volume defining the projection) cut away a portion of the model on one side of the plane. Figure 13 shows the use of 2 clip planes. Six planes are available in Moritz to clip the model from above and below in each of the X, Y, and Z directions. For each direction, there is a + and – plane. The +X clip plane, for example, eliminates elements of the model with X coordinate greater than the plane value. The OpenGL library executes the clipping.

Initially, the clip planes are undefined and disabled. The planes are defined (or changed) by setting their value graphically in one of the 2D windows. The *3D Clip Planes* items in the main and 2D context menus contain items for dragging the planes. Once one of these items is selected, moving the cursor in a 2D window with the left button held down moves the corresponding clip plane. Once a plane has been defined, it can be enabled or disabled. The *Clip Plane* submenus (there is also one in the 3D context menu) contain entries for changing the enabled/disabled state of each plane and a *Disable All* item.

4.8 3D Selection

A right click (without any modifier keys) in the 3D window shows the 3D context menu. If the click is on a cell, that cell blinks and the cell and material submenus show items pertinent to that cell and its material.

When selecting a cell drawn as wireframe, the cursor must be over one of the lines constituting the wireframe, not just anywhere on the surface represented by the lines. If wireframe cells are difficult to select, the size of the *Selection Mask* can be changed in the 3D options property page. The mask is the size, in pixels, of a rectangle centered at the cursor location. An object must fall within the rectangle to be selected. The object closest to the viewpoint is selected if multiple objects lie within the mask.

The blinking of the selected cell is accomplished by alternately displaying the scene with and without the cell visible. A property sheet field is available to change the blink rate. The default rate is 1/3 second. The blinking is cancelled when an item in the 3D context menu, including *Kill Selection*, is selected.

The entire model must be processed to figure out over which cell the click occurred and processed again to draw the scene with the selected cell left out. For complicated models, there may be a momentary lag before the context menu appears.

5. 3D APPEARANCE

5.1 Solid Styles

Shapes are drawn either as solid surfaces or in wireframe or are not drawn at all. The visibility and style is set for cells and materials on the respective property sheets and in the respective cell and material submenus of the 2D and 3D context menus. By default, cells are drawn solid unless they are characterized as a basket cell that includes a complement in its cell description or if they contain a filling universe. If multiple MCNP materials exist in a model, cells with material 0 are invisible.

The 3D context menu has items for *Draw All as Solid* and *Draw All as Wireframe*. When either item is checked, the individual styles are not changed but rather ignored. From the context menu one can *Hide all Solid* and *Hide all Wireframe* objects. Unchecking the items restores the original visibilities. The 3D style property page contains choices for *Draw All As Solid*, *Wire Frame*, or drawing with the *Individual Settings*. The line width used for wireframe objects is user-selectable.

The cell submenu of the 3D context menu contains an item to *Hide All Other Cells*. Only the current cell remains visible. Selecting *Restore Visibilities* in the 3D context menu restores the visibility of the other cells.

5.2 Lighting Model

The brightness and color of a surface depends on the colors of the light sources and cell color and on the position of the light source. Each light source and cell color has three components: *ambient*, *diffuse*, and *specular*. Ambient light is independent of the position of the light source. Diffuse light reflects off a surface depending on the angle between the surface and light source. Specular light also depends on the angle with a more peaked dependence on the angle. How peaked depends on the *shininess* value of the cell.

5.2.1 Light Sources

Up to 8 light sources can be used. In addition to the individual sources, there is a global ambient light. Light colors and positions are defined with the lighting property sheet. For each light source, the 3 colors can be set separately.

The lights are defined to be infinitely far away. The position, shown and set on the light position property page, defines the direction of the light source. The property page has a *Locate with Mouse* button that enables positioning of the light source by moving the cursor with the left button held down in one of the 2D windows.

The light positions are fixed. They do not rotate with the model. As such, the X, Y, and Z labels on the light position property page are misleading. We will add an option to have the lights remain fixed or rotate with the model.

5.2.2 Material Colors

The OpenGL terminology for the colors and shininess of an object is *material properties*. We use the same terminology regardless of whether an object's color is set by individual cell, MCNP material, or some other quantity such as importance.

At present, the interactive setting of individual ambient, diffuse, and specular colors and the shininess is on a per MCNP material basis. These quantities are set on the material property sheet. When 3D objects are colored by cell color, the relative contributions of the three colors are adjusted to be the same as the material colors and the material shininess is used. The individual cell colors can be set in a command file.

6. PARTICLE TRACKS

The MCNP ptrac file contains data about the paths of selected particles. Moritz can process the file and show the tracks of particles in the 3D display. Figure 14 shows an example. The tracks can be shown as connected line segments, as dots at the event locations recorded in the file, or both. The user can select the line width and dot size. Changes in the track display and selection options take effect immediately. Moritz can write a ptrac file containing only those tracks that meet the filter conditions in effect.

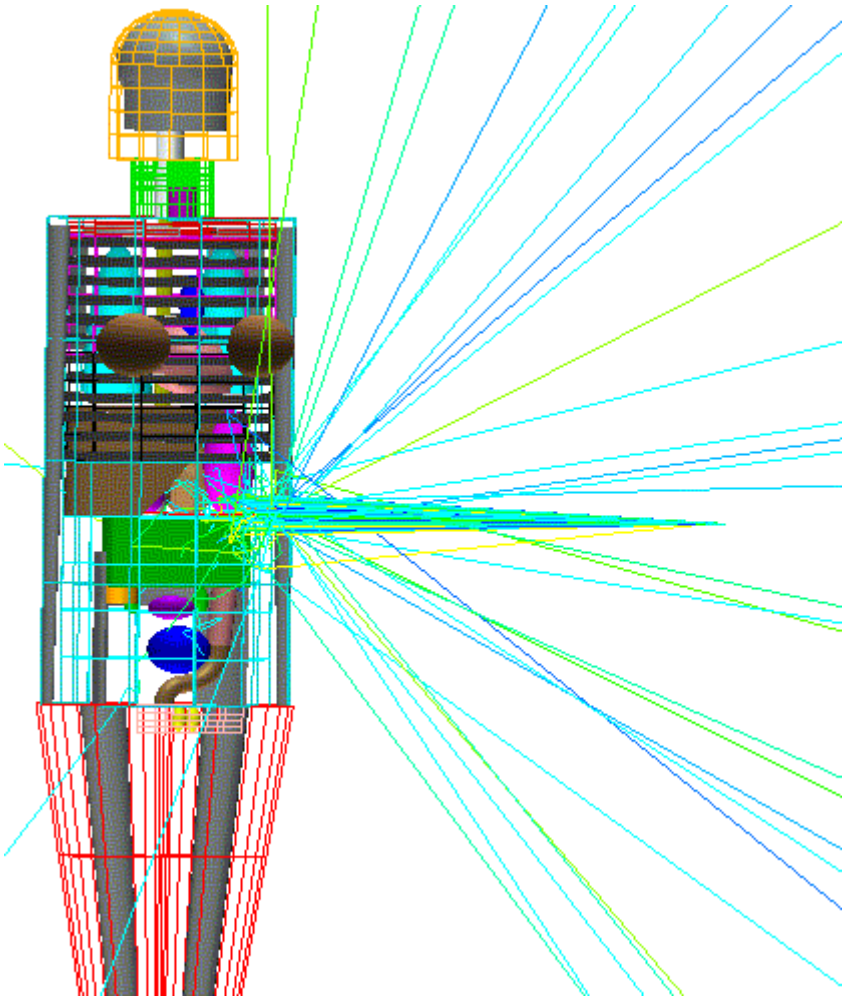

Fig 14. A human model with particle tracks.

6.1. Colors

The tracks can be colored by energy, weight, time, or particle type. For coloring by energy, weight, or time, the color coding can be by a discrete number of bins or continuous with a color scale that goes from blue to cyan to green to orange to red. The tracks in Figure 14 are continuously colored by energy. The energy and weight are constant along a track segment between 2 events, as is the color based on those values. When coloring by time, which is not constant along a segment, the colors at the segment end points is defined and OpenGL interpolates between the end colors along the segment.

6.2. Filters

A filter is a condition a particle history or branch must meet to be displayed. The filters include creation and termination type, entering or interacting with a specified cell, crossing a specified surface, interacting with a specified material or isotope, and having exactly, at least, or no more than some number of interactions. Any, all, or a given number

of filters must be satisfied. The tracks can be limited to a specified history or a range of histories.

6.3. Bounds

Minimum and maximum bounds can be imposed in the X, Y, and Z directions, the distance from the origin, and on energy, weight, and time. Track segments that cross a bound are clipped at the bound value.

7. CONCLUSIONS

The analysis and polygonalization routines embodied in the Moritz program quickly process MCNP-like geometries for interactive 3D viewing. There remain a number of cell shapes that cannot be polygonalized. We are working on methods to reduce that number. Further work will include support for repeated structures and solid bodies.

REFERENCES

Briesmeister, J.F. MCNP ™–A General Monte Carlo N–Particle Transport Code, Los Alamos National Laboratory Report, LA–13709–M, March 2000.

Lee, S.R., Anderson, J.W., Collins, D.G., Fowler, J.D., Hammar, N.A., Hughes III, H.G., Keen, N.D., Koch, K.R., Soran, P.D., Takamine, J.A., Van Riper, K. A., 1994. Unified Graphical User Interface for the Los Alamos Radiation Transport Code System, *Transactions of the American Nuclear Society*, **71**, 402.

Van Riper, K.A., 2000. New Features in Sabrina, Proc. Top. Meeting on Radiation Protection for our National Priorities, American Nuc. Soc., La Grange Park, IL, pp. 316–323.

Woo, M., Neider, J., Davis, T., Shreiner, D., 1999. *OpenGL Programming Guide*, Addison Wesley, Reading, MA.