STANDALONE VISUALIZATION TOOL FOR THREE-DIMENSIONAL DRAGON GEOMETRICAL MODELS

A. Lukomski, B. McIntee, D. Moule, E. Nichita

Faculty of Energy Systems and Nuclear Science
University of Ontario Institute of Technology

ABSTRACT

DRAGON is a neutron transport and depletion code able to solve one-, two- and three-dimensional problems. To date DRAGON provides two visualization modules, able to represent respectively two- and three-dimensional geometries. The two-dimensional visualization module generates a postscript file, while the three dimensional visualization module generates a MATLAB M–file with instructions for drawing the tracks in the DRAGON TRACKING data structure, which implicitly provide a representation of the geometry. The current work introduces a new, standalone, tool based on the open-source Visualization Toolkit (VTK) software package which allows the visualization of three-dimensional geometrical models by reading the DRAGON GEOMETRY data structure and generating an axonometric image which can be manipulated interactively by the user.

INTRODUCTION

Currently DRAGON provides two visualization modules, able to represent respectively two- and three-dimensional geometries. The PSP: two-dimensional visualization module images are generated in PostScript format using a colour scheme that suits the requirements intention of the user. The image helps the user differentiate between the make-up of unique regions of a fuel bundle, surrounding components and reflector materials. The TLM: three-dimensional visualization module generates a MATLAB M–file with instructions for drawing the tracks in the DRAGON TRACKING data structure, which implicitly provide a representation of the geometry
[2, 3]. The current work introduces a new, standalone, tool which allows the

visualization of three-dimensional geometrical models by reading the DRAGON GEOMETRY data structure and generating an axonometric image on the computer screen. The screen image can be manipulated interactively by the user.

The software utilized in this investigation is an object-oriented, open source package called *Visualization Toolkit* (VTK) [1]. VTK is capable of running on several different operating systems using different compilers. It is available for free online and can be supplemented with detailed user guides. In this investigation the program is run on Windows XP using the C++ compiler included with Microsoft Visual Studio. Fundamentally, geometric representation in VTK comprises a *graphics model*, which establishes a "stage" for the image to be created on and a *visualization model* which takes information from a source and transforms it into a graphical representation using various types of objects predefined in the VTK libraries [1]. The user is able to interact with the image in the graphics window through basic "interactors" which allow rotation, zoom, and other image manipulations. The underlying DRAGON geometry is not affected by the image manipulation in the viewer. Also, graphics window lighting effects can be modified to improve image quality where required. The ability to manipulate an image's representation in the window gives the user freedom to investigate critical regions of a geometrical model. Furthermore, regions which may be hidden under regular viewing angles can be revealed by using cut-away or transparency, both standard functions in VTK. Because VTK only renders images on a computer screen and does not produce a graphics file, it is up to the user to carry out screen capture, generating a .jpg or other graphics format file and subsequently process it as desired.

Geometric objects used in this work are an extension of basic data objects used in VTK corresponding to basic shapes such as spheres, cylinders and planes which make up the visualization model. Cylinder geometries are created through increases in resolution or increases in the number of geometric edges, beginning with a line of data points. With sufficient resolution, the geometry resembles a perfect cylinder.

METHODOLOGY AND RESULTS

The primary focus of this work was to establish how VTK can be utilized in conjunction with DRAGON to produce desired three dimensional images. In this preliminary stage the code cannot yet handle the entire range of three-dimensional DRAGON geometries but the future outlook of this project entails modifying the code to accommodate all geometries allowed by DRAGON. Two test models were selected for this investigation. The first was a three-dimensional CANDU lattice cell consisting of a 37-element CANDU fuel bundle, pressure tube, calandria tube and moderator. Each fuel ring has a different material and cladding is explicitly represented in the model. In short, this model is an extrusion of a two-dimensional CANDU-lattice cell model into the depth dimension. The geometry dimensions are based on a standard CANDU fuel channel lattice pitch of 28.575 cm and length of 49.53 cm. The second selected model consists of two adjacent three-dimensional cells plus a Zone Control Unit (ZCU) positioned mid-way between the bundles and perpendicular to their axes. All model parameters are read from an ASCII-file representation of the GEOMETRY data structure in DRAGON.

The code, developed in C++, reads the geometry file and eliminates unnecessary data. Information about clusters of fuel pins is grouped together while the other cylinders (calandria tube, pressure tube, etc.) are processed separately. This allows for the different regions to be modified individually, an important feature in instances where certain surfaces are to be (un)capped or made transparent (as discussed later). A Red-Green-Blue (RGB) colour palette similar to the one used in the DRAGON PSP: module was employed. The processing time for displaying each model is only several seconds, which allows a user to invoke the code frequently, if necessary. Each new invocation starts with the same view of the geometry. There is currently no option to save the changes made to the view during a run for use in a subsequent run.

The VTK representation was achieved by adding instances of called objects and orienting them appropriately with respect to the global reference frame (axes). The built-in *addCylinder* function was used to create the 37 fuel pins of each fuel bundle and the additional regions of the fuel channel. As with the PSP: module, colour-code designations were used to represent different material regions. It is important to note that surface rendering was used throughout the project and hence cylinders were represented using shells and not solid blocks. The potential exists for volume rendering to be implemented into the viewer; however, this possibility was not studied so far. A capping parameter exists which allows the ends of a cylinder to be either added or removed to reveal internal components and sub-layers which may not be otherwise visible. In Fig. 1, the non-fuel cylinders are uncapped allowing for a clear depiction of the capped fuel pins which would otherwise be covered. Extending the capping function to the ends of the non-fuel cylinders would create unwanted overlap where the different-mix regions would not be properly differentiated.

Some of the surfaces were also made semi-transparent, thus allowing the user to see beneath the surface into a cell volume. Variable transparency was implemented on different surfaces of the model, including the calandria tube, pressure tube and fuel cladding, to expose the fuel pins beneath.

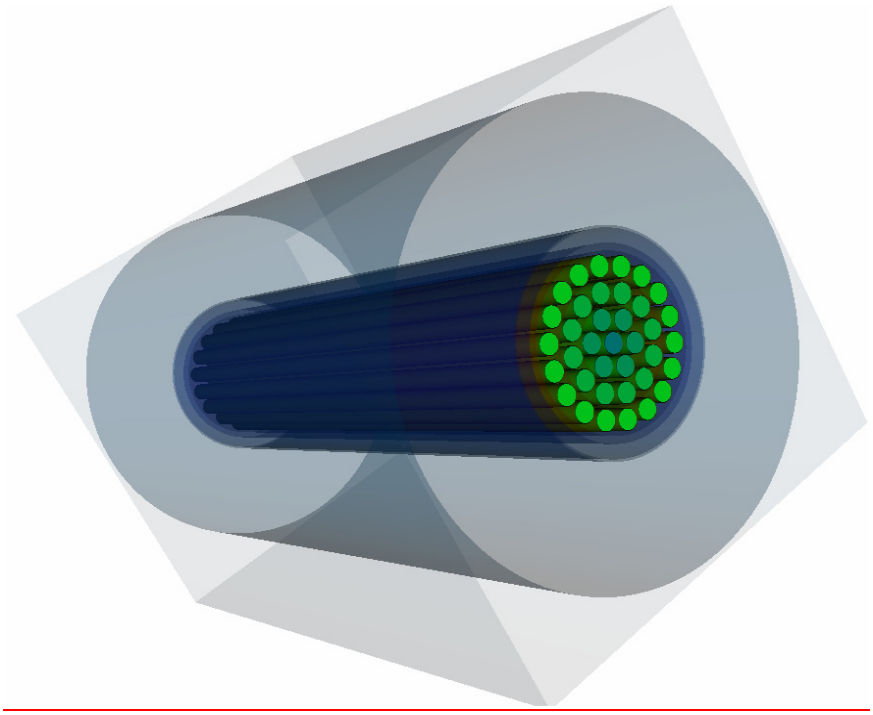

Figure 1: View of a 3D CANDU Lattice Cell

In the case when three-dimensional models are extruded versions of two-dimensional ones, it is considered desirable for consistency reasons to have the axial view of the three-dimensional model match the PSP:-generated view of the corresponding two-dimensional model. Figure 2 shows the axial view of the three-dimensional lattice cell as well as the view of the corresponding two-dimensional cell generated by the PSP: module. It can be seen that due to the transparency options and to the perspective representation, the VTK image is not identical to the one generated by PSP:. For example, the image's apparent additional square boundary is actually the boundary of the lattice cell on the back plane. The two images could be made identical by not using transparency and using full capping in the three-dimensional axial view. However, those options would be undesirable as they would prevent proper lighting of the inner regions resulting in dark areas of the three-dimensional model when viewed from different angles.

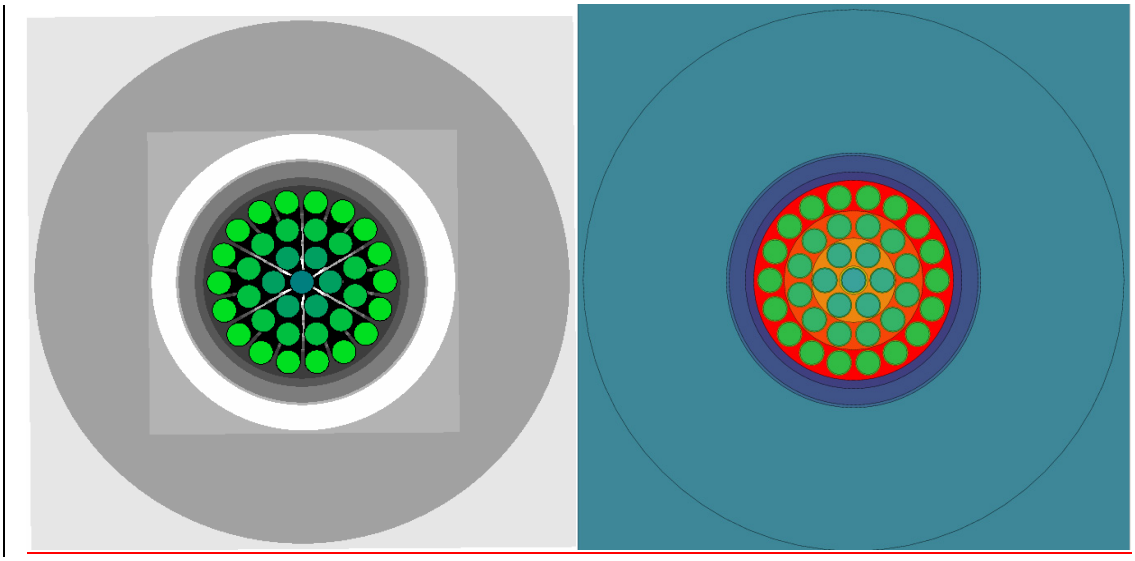

Figure 2: 3D Lattice Cell Axial View (left) and 2D Lattice Cell View (right)

An alternative to using transparency is to use a wireframe representation, as illustrated in Fig. 3. The wireframe representation is not effective when the model includes a large number of sub-geometries.

One of the advantages of using VTK is the ability to manipulate the image through simple predefined "interactor" functions which allow rotation and zoom, as well as more complicated manipulations. There is no need to re-execute the code for different orientations and almost all manipulation can take place within the graphics window.

For example, when viewing the image from the initial zoom condition it is difficult to distinguish between the different coolant-material regions due to the lack of end capping. Increasing the zoom on the image better defines the boundaries as can be seen in Fig. 4.

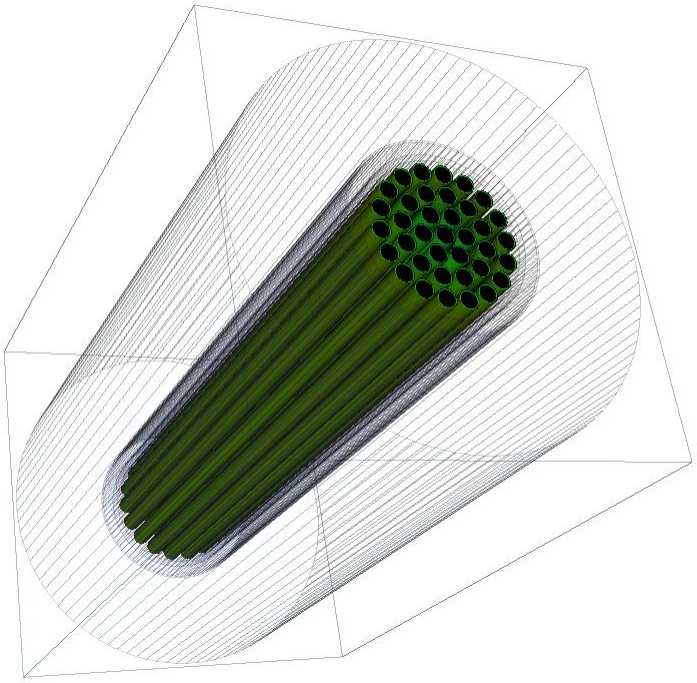

Figure 3: Wireframe Representation of a 3D Lattice Cell

The zoom function also assists in differentiating the colour of the mixtures as slight variations are hard to distinguish. The colour palette was chosen to be as consistent as possible with the two-dimensional image generated by the PSP: module.

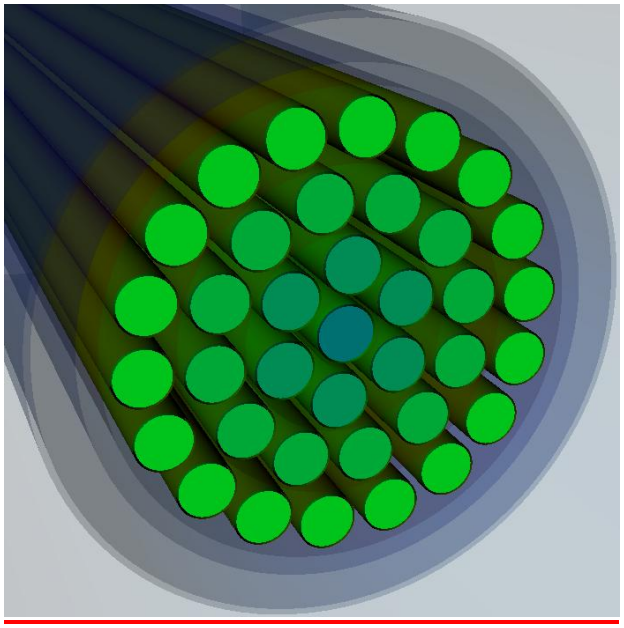

Figure 3: Close-up View of Fuel Bundle End

An undesired consequence of the on-the-fly image manipulation, is that images cannot be saved upon execution and so the user must re-orient and zoom the image to the desired position at each run and use screen capture to save it.

To further test the viewer, a supercell model consisting of two standard 37-element CANDU fuel bundles and a ZCU was used. The resulting view of the model is shown in Fig. 5. The ZCU is positioned at the origin, with the fuel bundles positioned one half-lattice-pitch away on each side. The bundle axes are perpendicular to the ZCU axis. Although the fuel bundles are geometrically identical, each is represented individually. This is evidenced by the slightly different colours employed on each of the fuel bundles. The lattice cell boundary is not represented.

The same transparency level of the calandria and pressure tubes as in the single-bundle model was used. For the ZCU shell, a high level of transparency was used in order to reveal the detailed inner components of the device. The generated image presents the geometric details with a high degree of clarity with the exception of the lack of capping on the fuel bundles as previously discussed. The VTK "interactor" functions were used to remove some surfaces, thus revealing inner components; this is especially true for the ZCU.

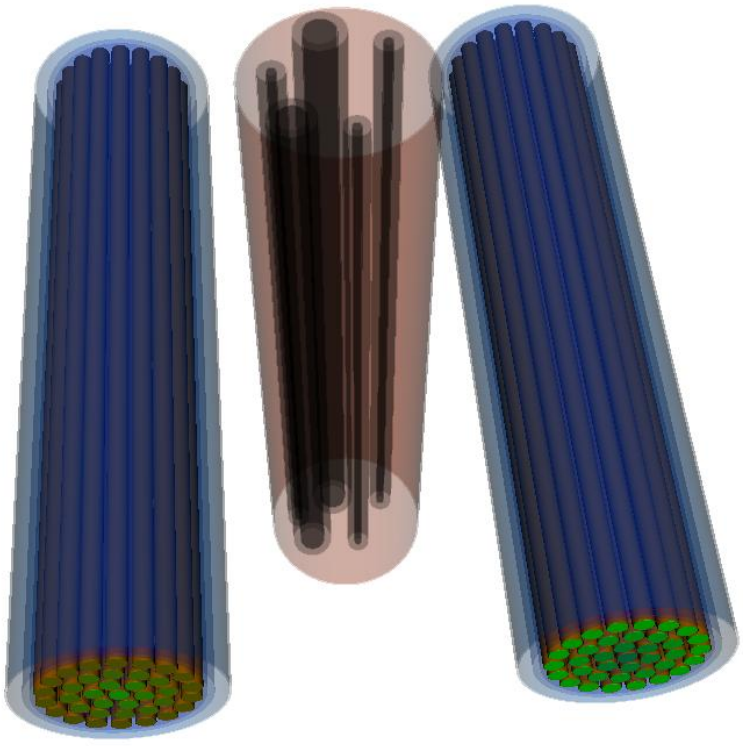

Figure 5: View of 3D Supercell Consisting of Two Fuel Bundles and ZCU

An approximate top view of the model is shown in Fig. 6.

Figure 6: Top View of 3D Supercell

CONCLUSION AND FUTURE WORK

A stand-alone three-dimensional viewer was developed for the transport code DRAGON using the open-source VTK software package. Preliminary tests show the viewer to work correctly on simple models and to provide a useful tool for DRAGON input preparation and verification. The viewer allows the user to manipulate interactively both the image as a whole and individual parts, and thus to focus on and analyze model components.

Future work will focus on fine-tuning the code to achieve better representation of complicated models. In particular, volume-rendering coupled with a cutting plane will be studied, as well as ways to add and remove caps and colours interactively. Co-ordinate axes will be added to the view to facilitate orientation. The code will also be modified to allow the representation of any DRAGON geometry. In the more distant future, a visual editor allowing the interactive modification of DRAGON geometries will be considered.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. G. Marleau of École Polytechnique de Montréal for providing the DRAGON input file used to generate the ZCU test geometry.

REFERENCES

[1] "The VTK User's Guide," *Kitware, Inc., (2006)*

[2] C. Plamondon, "Verification des lignes d'integration et illustration des geometries DRAGON", *Technical Report, IGE-290, Ecole Polytechnique de Montreal (2006)*

[3] A. Zkiek, G. Marleau, "Verification of DRAGON: The NXT Module.", *28th CNS annual conference, St. John, NB (2007)*