



## 3D Visualization for the MARS14 Code

J.P. Rzepecki<sup>†</sup>, M.A. Kostin, N.V. Mokhov

*Fermi National Accelerator Laboratory P.O. Box 500, Batavia, Illinois 60510*

<sup>†</sup> *Nicolaus Copernicus University, ul. Grudziadzka 5, 87-100 Torun, Poland*

January 9, 2003

### **Abstract**

A new three-dimensional visualization engine has been developed for the MARS14 code system. It is based on the OPENINVENTOR graphics library and integrated with the MARS built-in two-dimensional Graphical-User Interface, MARS-GUI-SLICE. The integrated package allows thorough checking of complex geometry systems and their fragments, materials, magnetic fields, particle tracks along with a visualization of calculated 2-D histograms. The algorithms and their optimization are described for two geometry classes along with examples in accelerator and detector applications.

# 1 Introduction

Nowadays, a good visualization package is a vital part of any Monte Carlo code that involves description of the geometry while simulating particle interactions and tracking. It is mandatory in high-energy physics detector, accelerator and shielding applications. Examples include GEANT [1], MARS [2], FLUKA [3] and MCNPX [4] code Graphical-User Interfaces (GUI). They are used to find geometry misrepresentations and better understand the simulated processes.

The MARS14 two-dimensional Graphical-User Interface, MARS-GUI-SLICE, is based on *Tcl/Tk* and linked to a MARS executable. The interface displays all the details of the coded geometry and its fragments, showing the zone numbers, materials, magnetic fields, color-coded particle tracks and calculated histograms in an arbitrary two-dimensional slice of the system studied. It is a very powerful and user-friendly tool for checking complex geometries and the correctness of particle tracking before executing histories, and for visualization of the results projected onto a corresponding cross-section of the system. An arbitrary 3-D rotation of a slice is possible. Numerous control options are provided. A chosen graphical display panel can be saved as an encapsulated postscript file.

A new package for the MARS code has been created that further extends the power of visualization adding a crucial in many instances three-dimensional view of the system studied. This paper briefly describes the algorithms and optimization methods for both the *Non-standard* and *Extended* geometry classes in MARS along with examples from recent applications in the Fermilab accelerators and experiments.

## 2 3-D Visualization Package

The new package is built on the top of the OPENINVENTOR [5] SGI library. It allows 3-D visualization for the geometry classes available in the MARS14 code, including a recently completely rebuilt the *Extended* geometry mode. The later is based on the predefined shapes such as the box, cone, cylinder and sphere which are used to generate a geometry description utilizing numerous cloning and transformation options [6].

The OPENINVENTOR package is available as a source code (it compiles on Linux and IRIX machines) and also as an RPM package. It is included in most of the Linux distributions, *e.g.* Debian and Red Hat. Since OPENINVENTOR is based on OpenGL [7], the whole rendering is done via OpenGL routines and therefore a user can take advantage of 3-D graphics accelerators which are quite crucial in modern 3-D visualization. A 3-D view panel (Fig. 1) is launched from the MARS-GUI-SLICE window. It gives a user the possibility of defining a part of the mother volume to be rendered with a corresponding scanning step. One can use the OPENINVENTOR 3-D GUI to rotate, zoom in and out and move the scene. Beside those standard OPENINVENTOR functions, other useful features have been added such as changing from a solid to wire-frame rendering, a light editor, and an ability to cut a fraction of the 3-D view, save the scene image to a postscript file and modify the background. One can also plot 3-D particle tracks and modify a color and transparency of materials chosen from the MARS-GUI-SLICE window.

Once in the *Non-standard* geometry mode (see next section), a picture created may look “boxy” when scanned with a large step-size  $\Delta$ . A user can improve the “resolution” by reducing  $\Delta$  for the price of an increased visualization time. A corresponding optimization algorithm is described below.

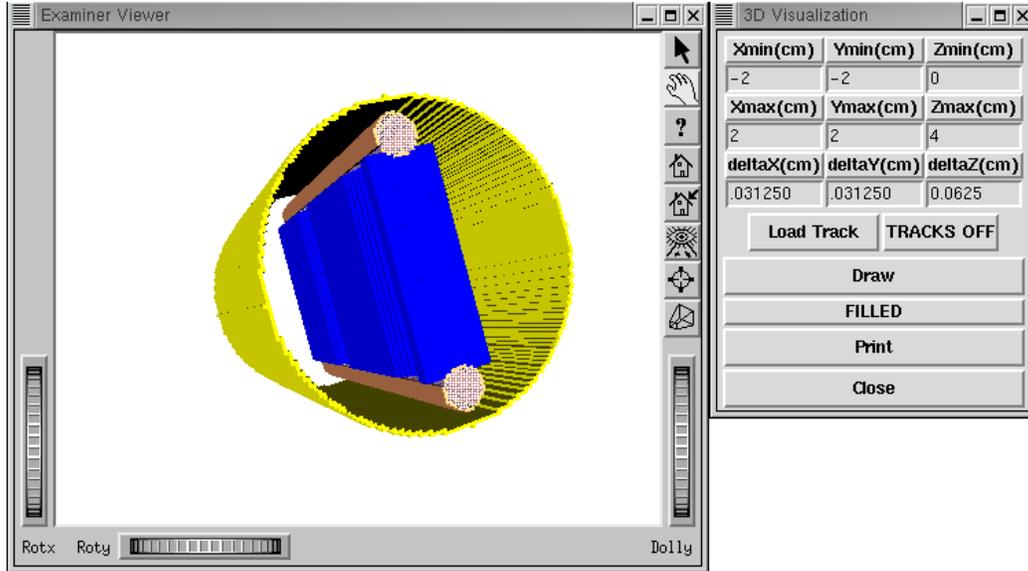


Figure 1: 3-D GUI window displaying a segment of the NuMI target.

### 3 Visualization for Non-Standard geometry

#### 3.1 Algorithm

Typically, it is the *Non-standard* geometry class that is used in most cases with the MARS code. This mode is based on a geometry representation through a set of user *Fortran* or *C* subroutines (*REG1*, *VFAN* and so on) [2]. Almost any arbitrary complex geometry can be precisely described this way. The price a user pays for that is a lot of coding. The visualization is absolutely vital for debugging the above routines and checking the entire geometry description.

The basic idea of creating a 3-D scene for this type of geometry is to scan the entire mother volume for zone numbers and materials with some steps  $\Delta_x$ ,  $\Delta_y$  and  $\Delta_z$ . Then every scanned point is replaced with a parallelepiped  $\Delta_x \times \Delta_y \times \Delta_z$ , with a color corresponding to the given material.

In order to illustrate the algorithm, let us consider a 2-D example with a user-defined geometry to be a simple circle. The result of scanning the entire mother volume with a coarse step-size is given in Fig. 2(left). It is rather hard to say that the central region is a circle. The picture on the right is closer to the reality because the scanning step-size is decreased by a factor of two. Although the resolution is better there is a trade-off. The number of boxes is increased by a factor of four in a 2-D world and by a factor of eight in

a 3-D case. This makes the scanning process slower and requires more computer memory. For example, if a 3-D scene is just 100 units long in all the dimensions and  $\Delta$  is equal to five, then the total number of parallelepipeds is 8000. Rendering 8000 parallelepipeds is a rather slow process. Therefore, a better way is needed to provide a relatively good resolution with a smaller number of objects. If one takes a look at Fig. 2 again, one could notice that a number of small boxes can be replaced with bigger ones without any loss of detail.

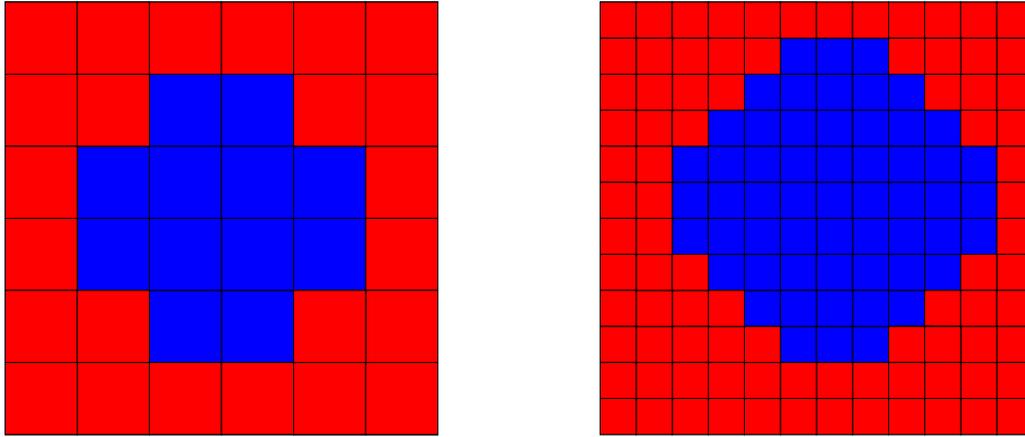


Figure 2: A “circle” created with a scanning step-size  $\Delta=\Delta_x=\Delta_y$  (left) and  $\Delta/2$  (right).

### 3.2 Optimization

Fig. 3 shows three steps of creating a bigger box. The general idea is to check out whether all the small boxes, next to the expanding one, are of the same material. First, the boxes on the right side are tested, then those on the bottom. If they are made of the same material as the expanding box, then the box is expanded in both directions (first two pictures in Fig. 3). If the material of the candidate boxes on the bottom is not the same as the one of the expanding box, then only the right direction is used (third picture in Fig. 3). In a 3-D case, the third direction is also used.

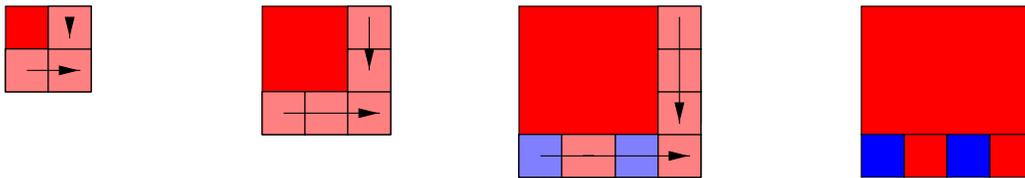


Figure 3: Three steps and result of optimization.

Fig. 4 shows the “circle” from Fig. 2 after the above optimization. The number of objects is reduced from 144 to 23. In a 3-D case, applying the above algorithm results in

even a larger decrease in the number of objects and substantial CPU-time saving. Once initialization is complete, an interactive manipulation is quite fast.

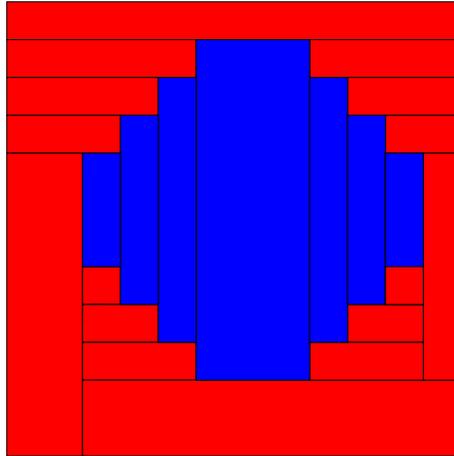


Figure 4: The “circle” from Fig. 2 (right) after optimization.

### 3.3 Non-Standard geometry examples

Figs. 5-7 present three visualization examples created with the described method. They show the Fermilab Booster and NuMI beamline systems coded within the MARS14 framework. The models include many geometry zones, materials and magnetic fields. The particle track visualization provides a “bubble chamber” regime, especially helpful in debugging beamline models and analysis of cascade processes in complex cases.

## 4 Extended Geometry

Geometry representations through the predefined shapes is used in several Monte Carlo programs such as GEANT [1], FLUKA [3] and MCNPX [4]. Earlier versions of the MARS code also allowed such an option in a rather simple form. Such an approach can be very powerful when a set of the predefined shapes is sufficient to describe the geometry under consideration. Visualization of the *Extended* geometry shapes in MARS14, such as the box, cone, cylinder and sphere is trivial: the shapes are directly represented by OPENINVENTOR. Note that the geometry classes in MARS14 can be mixed up together because the *Extended* geometry mode does not override the others. Fig. 8 shows an example of an *Extended* geometry simplified model of the CDF detector used to study backgrounds due to beam loss in the detector vicinity.

## 5 Acknowledgments

Thanks to O.E. Krivosheev for useful discussions.

## References

- [1] <http://wwwinfo.cern.ch/asd/geant4/geant4.html>.
- [2] N.V. Mokhov, "The MARS Code System User's Guide", Fermilab-FN-628 (1995);  
N.V. Mokhov, O.E. Krivosheev, "MARS Code Status", Proc. Monte Carlo 2000  
Conf., p. 943, Lisbon, October 23-26, 2000; Fermilab-Conf-00/181 (2000);  
<http://www-ap.fnal.gov/MARS/>.
- [3] <http://pcfluka.mi.infn.it>.
- [4] <http://mcnpx.lanl.gov>.
- [5] <http://www.openinventor.com>.
- [6] M.A. Kostin, N.V. Mokhov, to be published;  
see also <http://www-ap.fnal.gov/MARS/>.
- [7] <http://www.opengl.org>.

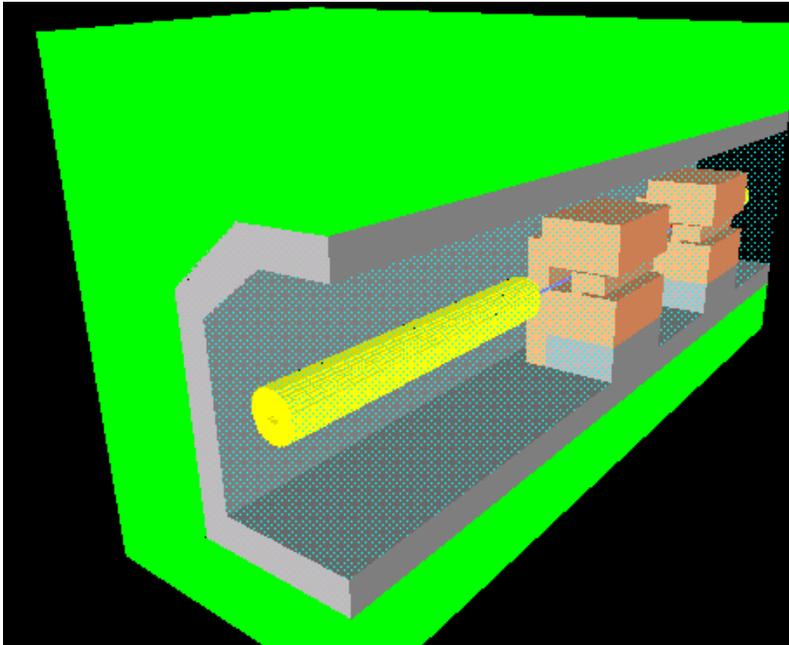


Figure 5: The Fermilab Booster collimation section.

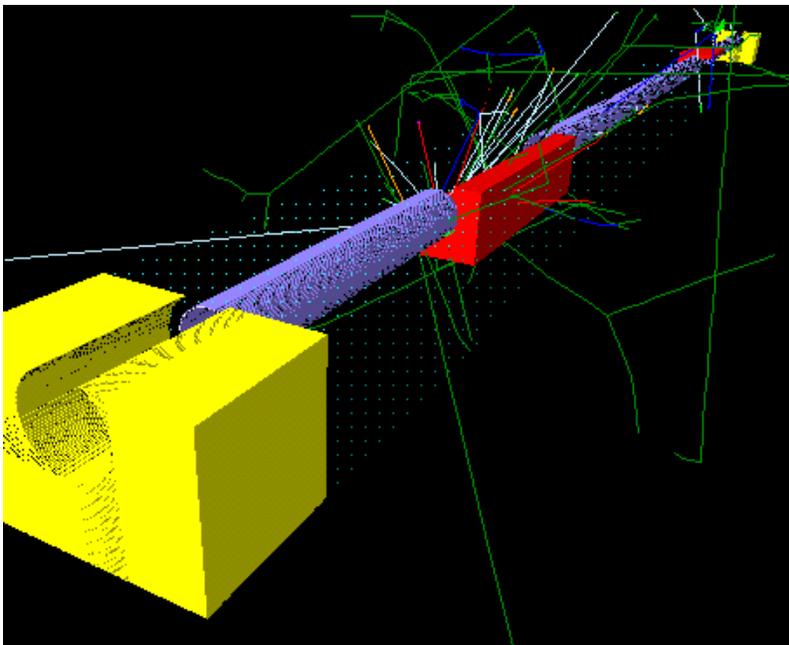


Figure 6: Details of the Fermilab Booster collimation section with “L” shaped collimators and particle tracks shown.

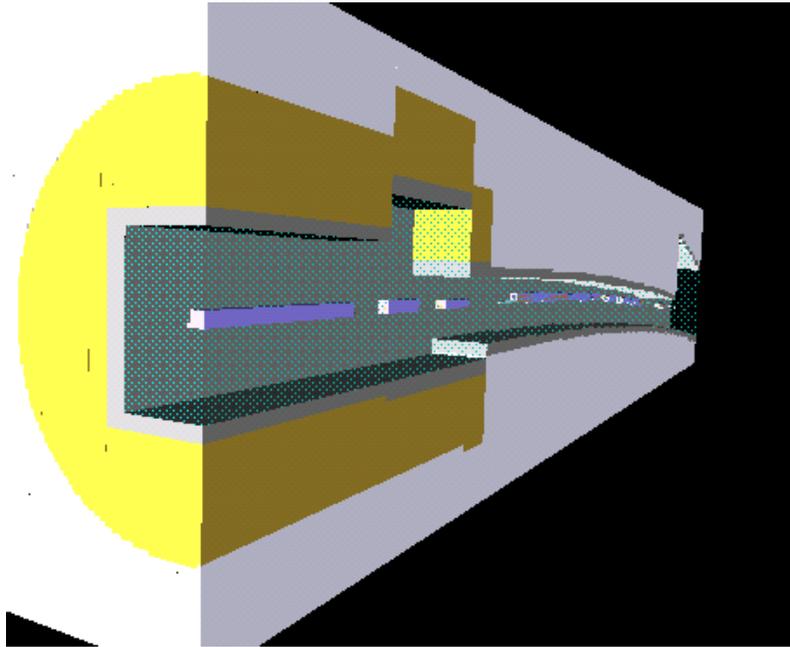


Figure 7: A fragment of the NuMI beamline and tunnel.

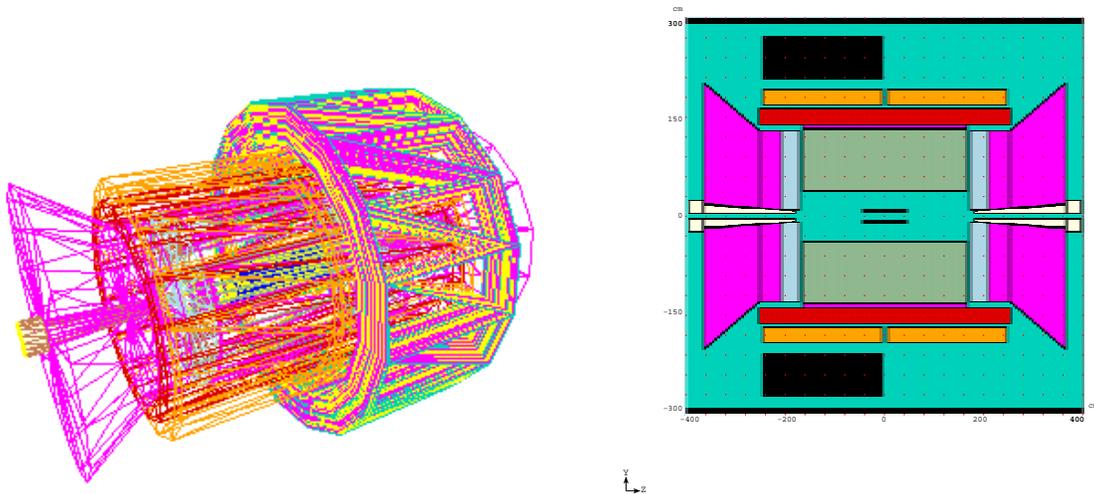


Figure 8: 3-D wire-framed (left) and 2-D (right) representations of a simplified model of the CDF detector.