



Open CASCADE Technology  
6.8.0

Voxel Package

November 7, 2014

## Contents

<b>1</b>	<b>Introduction . . . . .</b>	<b>1</b>
<b>2</b>	<b>Data structure . . . . .</b>	<b>4</b>
<b>3</b>	<b>Algorithms . . . . .</b>	<b>5</b>
<b>4</b>	<b>Visualization . . . . .</b>	<b>6</b>
<b>5</b>	<b>Demo-application . . . . .</b>	<b>8</b>
<b>6</b>	<b>Future development . . . . .</b>	<b>15</b>

## 1 Introduction

A voxel is a sub-volume box with constant scalar/vector value. The object in voxel representation is split into many small sub-volumes (voxels) and its properties are distributed through voxels.

Voxels are used for analysis and visualization of 3D-dimensional distribution of data. Medicine (mainly, tomography), computational physics (hydrodynamics, aerodynamics, nuclear physics) and many other industries use voxels for 3D data visualization and analysis of physical processes.

To produce a voxel representation the 3D space is split by equal intervals along the main orthogonal coordinate axes to obtain  $n_x \times n_y \times n_z$  voxels (small cubes):

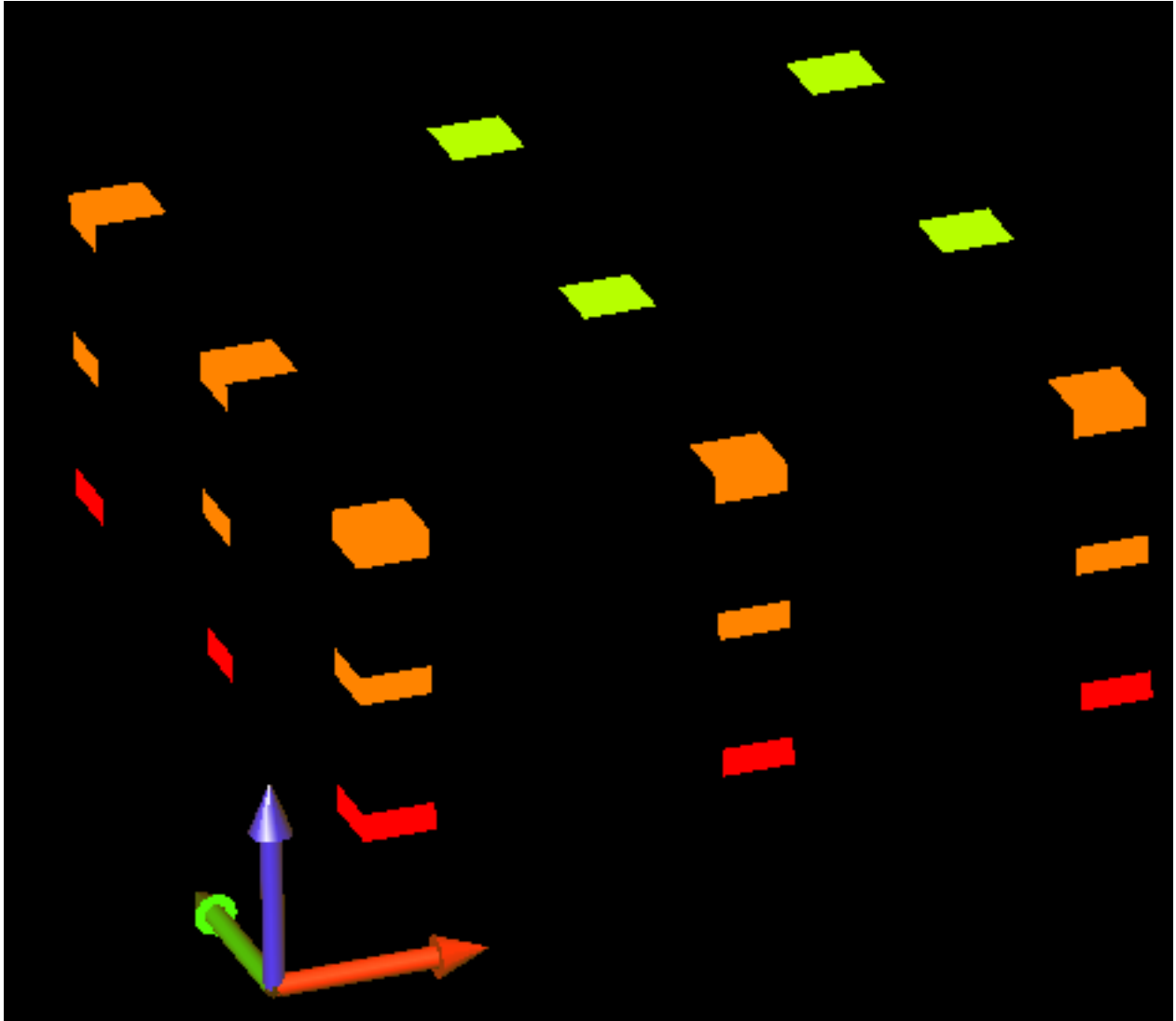


Figure 1: A cube of  $3 \times 3 \times 3 = 27$  voxels.

The data are attached to each voxel and remain the same within the voxel. It means that we obtain the 3D space with discrete data distribution.

The number of voxels used in a calculation can vary. An average model contains several tens of millions of voxels. Such a great amount of data requires special algorithms of computation, data containers keeping data in memory and visualization tools.

Open CASCADE Technology provides several basic data containers for voxels with fast access to the data and optimal allocation of data in memory.

Also, a special visualization toolkit allows visualizing voxels as colored or black/white points and cubes, displaying only the voxels visible from the user's point of view.

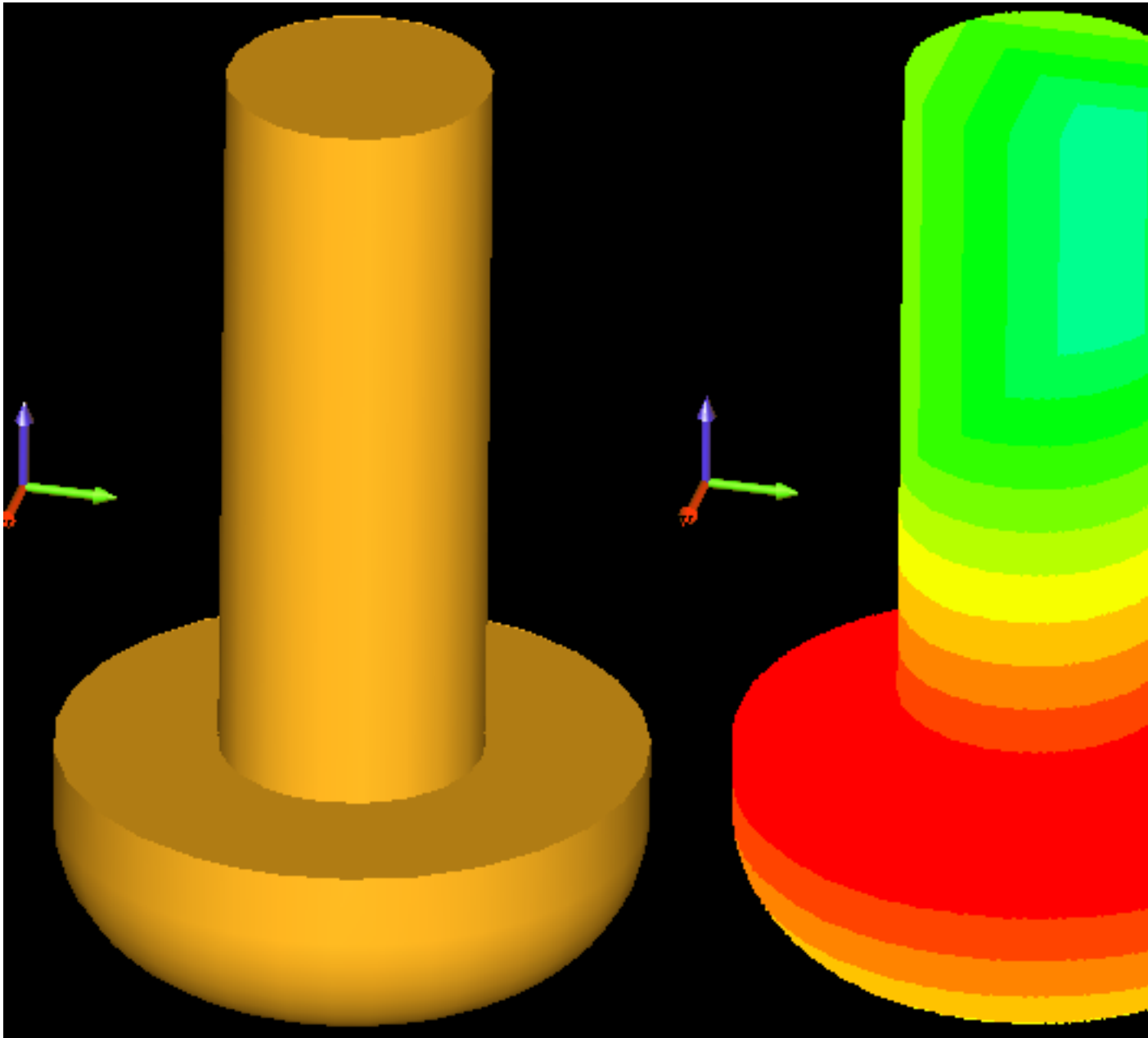


Figure 2: A shape and its voxel representation

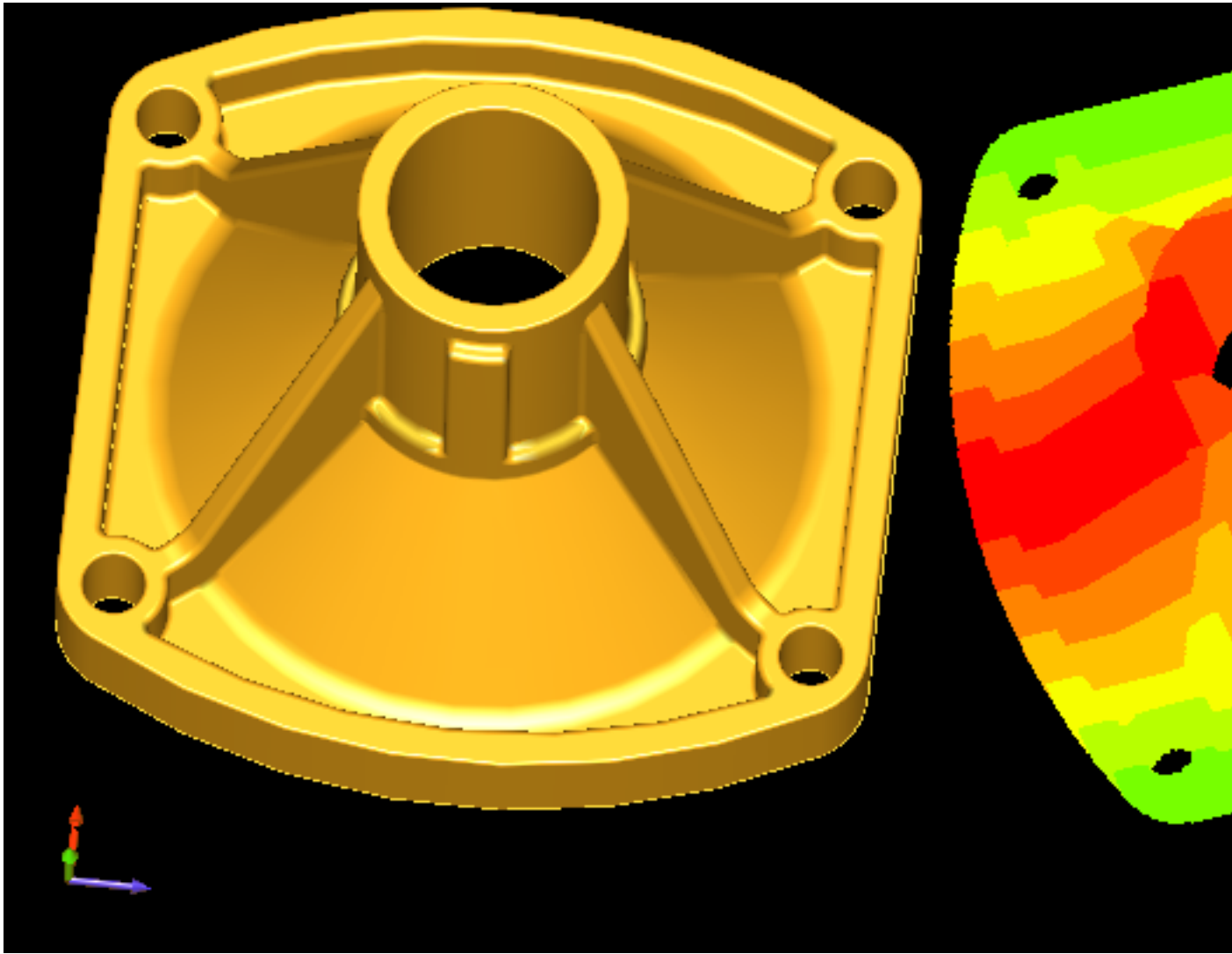


Figure 3: A shape and its voxel representation

In these images a boundary representation is displayed to the left. In the center and to the right there are 3D discrete representations (or 3D discrete topology). Any solid shape can be translated into a voxel representation.

## 2 Data structure

The data structure to store the voxels data is a special class which gives fast access to the data of each voxel and allocates the data in an optimal way in the memory of a computer.

Fast access to the data is provided by means of bit-wise operators on the indices of internal arrays.

The optimal data allocation is reached through division of the whole data set into data subsets and keeping only non-zero pieces of data in memory.

A voxel can contain different data types, but presently Open CASCADE Technology implements only several basic ones:

- 1 bit or Boolean data type – a voxel contains a flag: 0 or 1 (false or true).
- 4 bits or Color data type – a voxel contains a value occupying 4 bits. It is an integer in the range of 0 .. 15. The data can be divided into 16 subsets and displayed by Color-voxels.
- 4 bytes or Float data type – a voxel contains a floating-point data type.

In addition, the data structures provide methods for calculation of a center point by voxel indices and a reverse task – fast search of a voxel by a point inside the cube of voxels.

### 3 Algorithms

There are two service classes implemented for data structures of voxels:

- Boolean operations – provides simple boolean operations on cubes of voxels (fuse and cut).
- Voxelization – the conversion of a geometrical model into its voxel representation.

#### Boolean operations

Fusion and cutting of two cubes of voxels are performed the class *Voxel\_BooleanOperations*. The cubes should have the same size and be split into voxels in the same way.

- *::Fuse()* summarizes the values of the corresponding voxels and limits the result by the upper limit (if succeeded).
- *::Cut()* subtracts the values of the corresponding voxels and limits the result by zero.

#### Voxelization

A class *Voxel\_Convert* converts a *TopoDS\_Shape* into one of the voxel data structures filling the solid shape by non-zero values.

The algorithm of voxelization generates only 1-bit or 4-bit voxels. Other data types may be obtained by conversion of voxels from one type to another.

Voxelization of a shape is performed by means of computation of intersection points between lines filling the volume and triangulation of the shape. The lines are parallel to main orthogonal axes and can intersect the shape from different sides: along +X, +Y and/or +Z axes.

The algorithm can run in multi-threaded mode (the number of threads is unlimited). The user can see an integer value indicating the progress of computation.

## 4 Visualization

Visualization of voxels is not a simple task due to a great amount of data used for 3D analysis.

Open CASCADE Technology allows visualization of a cube of voxels in two modes:

- Points – the centers of voxels as 3D points.
- Boxes – the voxels as 3D cubes of adjustable size.

A degenerated mode displays only the points (boxes) visible from the point of view of the user for transformation operations (zoom, pan and rotate).

To focus on a particular part of the model non-relevant voxels can be erased. The displayed region is defined by six co-ordinates along X, Y and Z axes .

It is possible to display the voxels from a particular range of values (iso-volume):

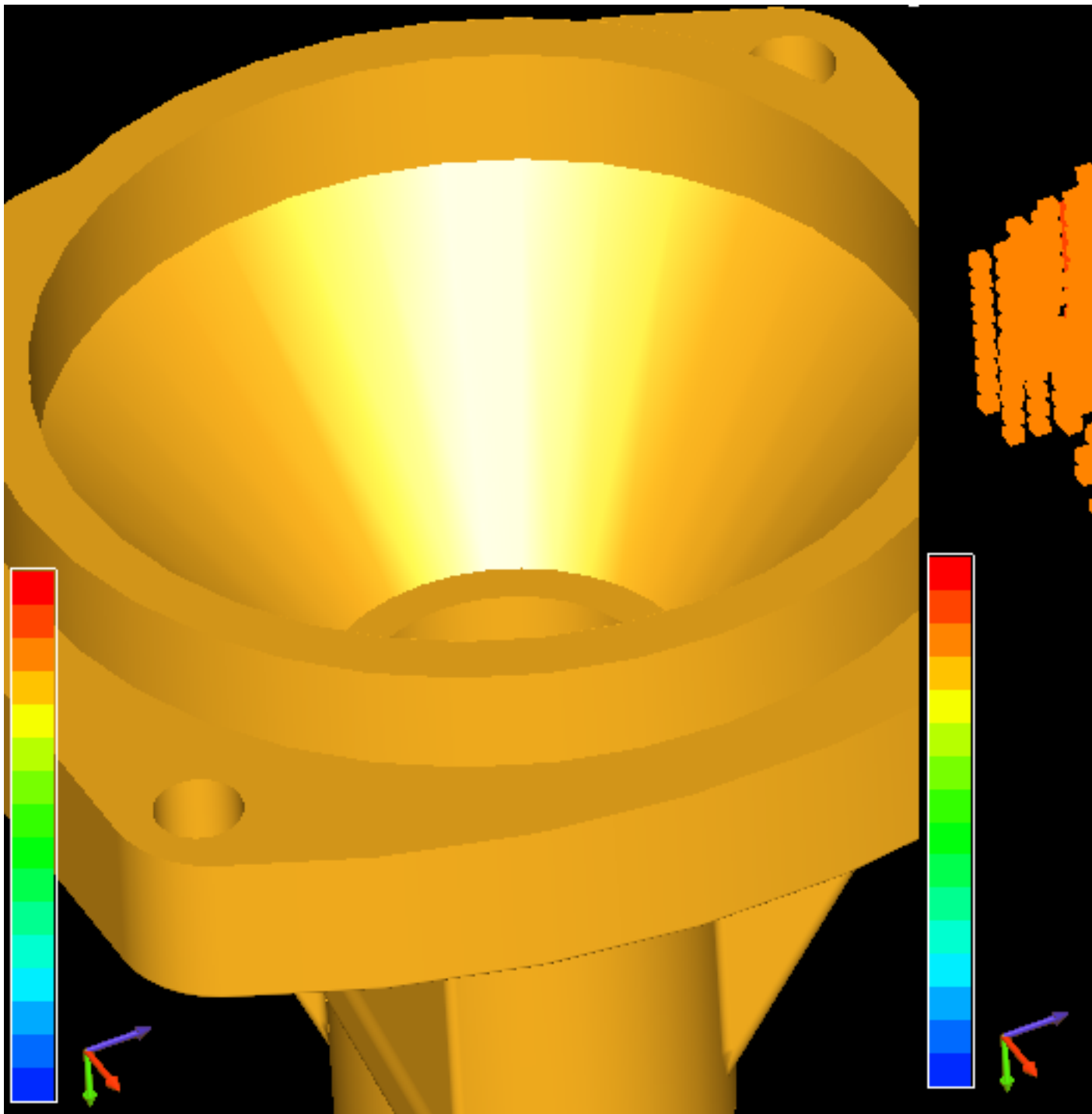


Figure 4: Iso-volume of a shape

The voxels are displayed by means of "direct drawing in Open GL" technology or "user draw" technology. Therefore, some visualization files are compiled within Open CASCADE Technology, but the files of "direct drawing" are compiled by the end-user application.

It is necessary to include the files *Voxel\_VisData.h*, *VoxelClient\_VisDrawer.h* and *VoxelClient\_VisDrawer.cxx* into the visualization library of the application (containing all files of *OpenGL* package) and call the method *Voxel\_VisDrawer::Init()* from the application before the visualization of voxels.

## 5 Demo-application

A demonstration application has been created to show OCCT voxel models. This is a test demo application because it includes a set of non-regression tests and other commands for testing the functionality (accessible only through TEST pre-processor definition).

The *File* menu allows creation of canonical shapes (box, cylinder, sphere, torus) or loading of shapes in BREP format:

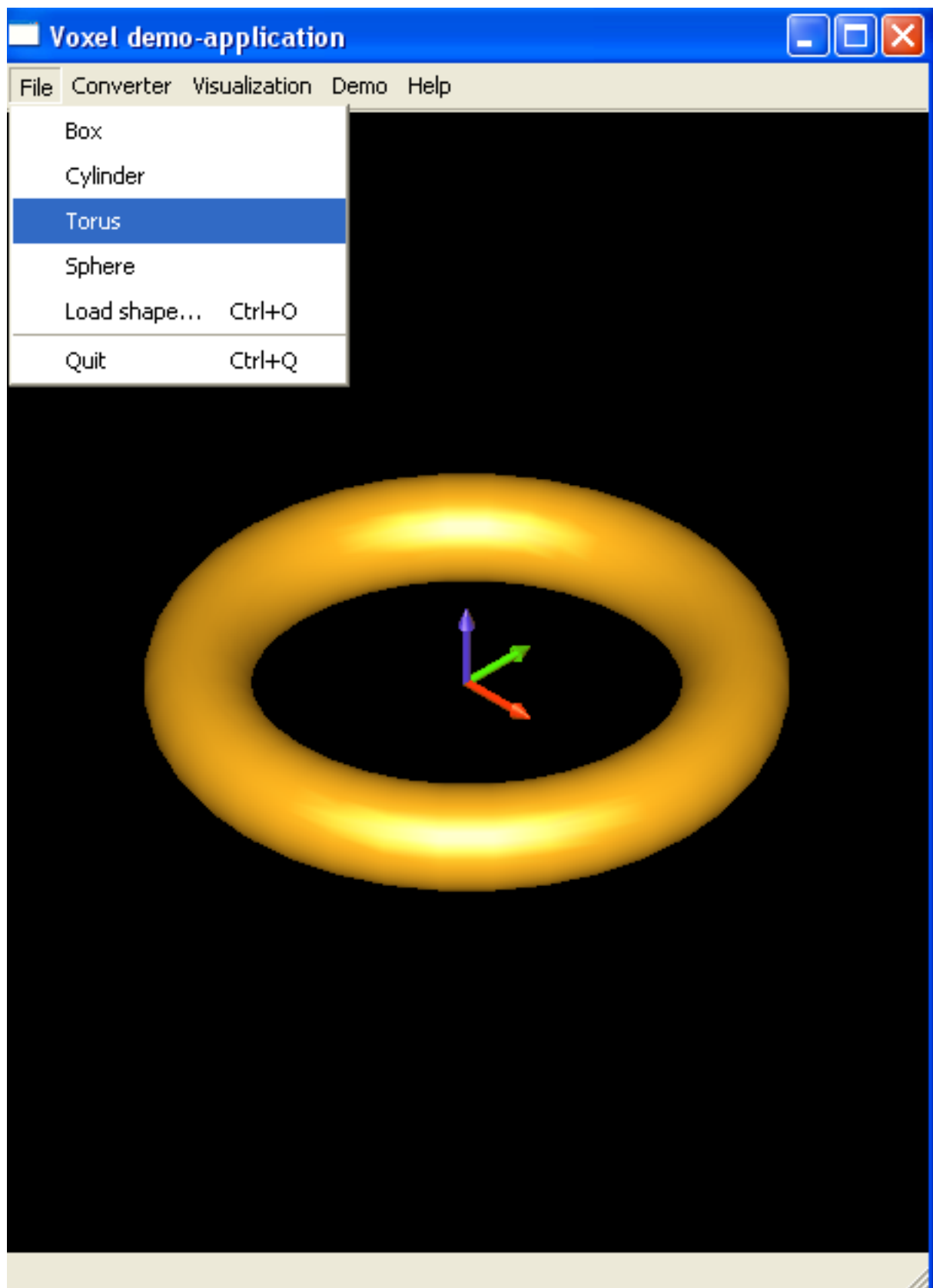


Figure 5: Demo-application. Creation or loading of a shape

The menu *Converter* voxelizes the shape. Two types of voxels can be obtained: 1-bit or 4-bit voxels.

- 1-bit voxels are displayed in white color on black background.
- 4-bit voxels use 16 colors filling the model in a special way for demonstrative purposes:

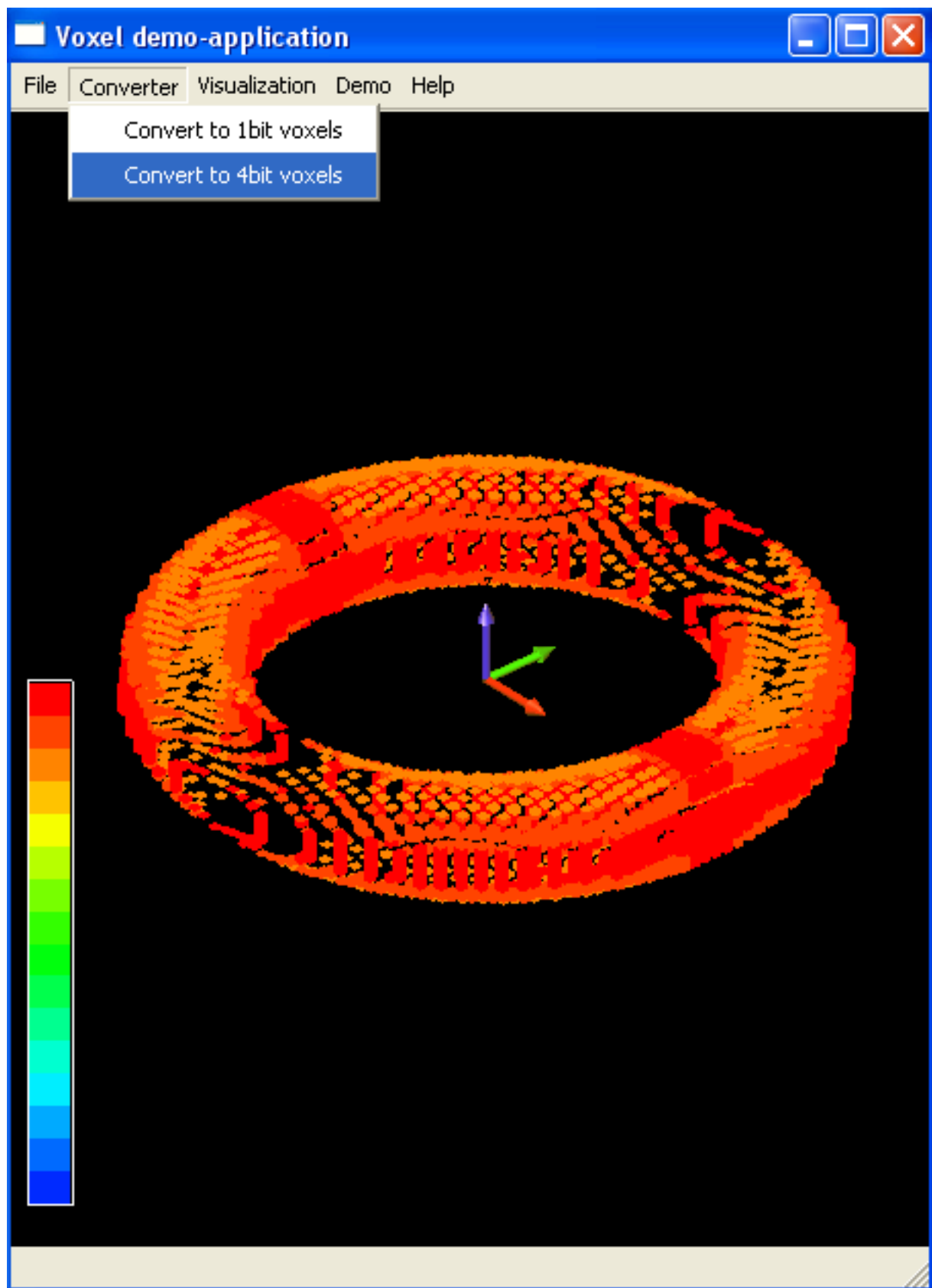


Figure 6: Demo-application. Voxelization

The converter uses two threads (two processors, if available) to perform voxelization.

The menu *Visualization* offers two modes of visualization: Points and Boxes, allows defining the size of points and boxes (quadrangles), the minimum and the maximum displayed color, and the boundaries of the bounding box for displayed voxels:

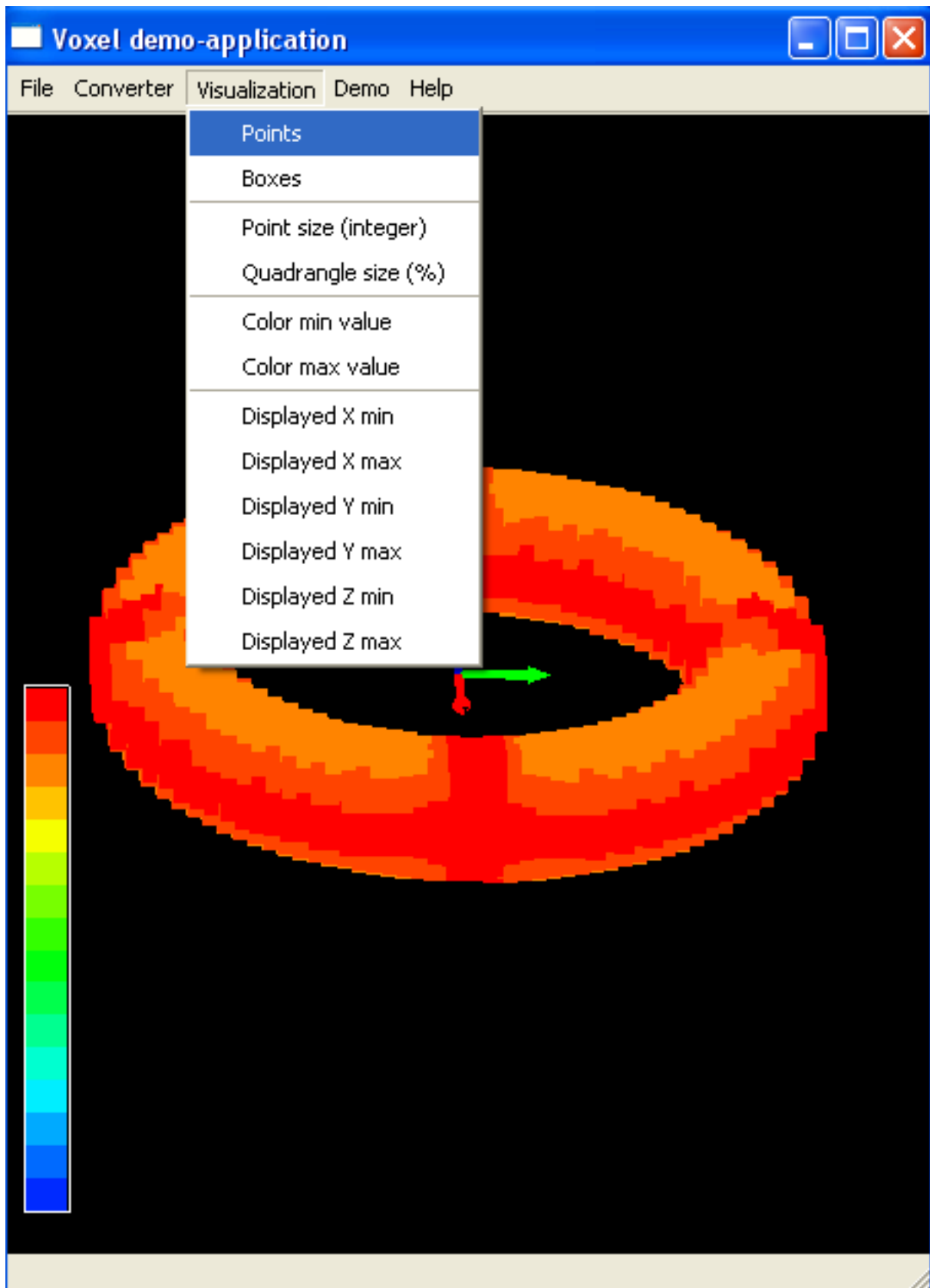


Figure 7: Demo-application. Visualization

The last menu, *Demo* contains a demo-command for running waves of 4-bit voxels:

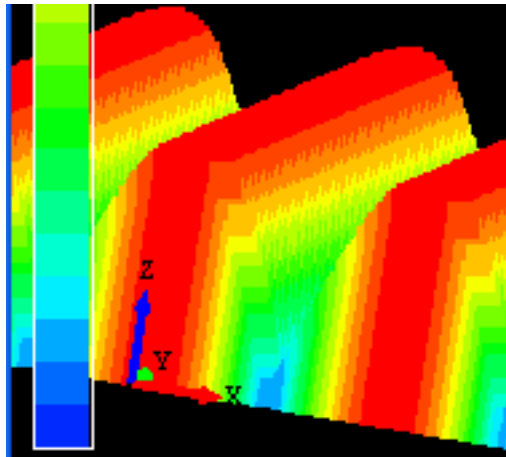


Figure 8: Demo-application. Running waves

## 6 Future development

In the future OPEN CASCADE plans to develop the platform of voxels in the following directions:

- Data structure:
  - Extension of the list of basic data types.
  - Development of a deeper hierarchy of voxels (for example, octree – division of a voxel into 8 sub-voxels).
  - Development of a doxel (4D voxels where the fourth co-ordinate is the time, for example).
- Algorithms:
  - Conversion of a voxel model into a geometrical model (a reversed operation to voxelization).
- Visualization:
  - Optimization of visualization (mainly, the speed of visualization).
  - New shapes of voxel presentation in the 3D Viewer and new approaches to visualization.
  - Selection of voxels.