Voxelizer plug-in for Blender

Release 1.00

Roman Grothausmann[1,3], Julien Gout[2] and Mark Kühnel[3]

November 9, 2012

[1]roman.grothausmann@helmholtz-berlin.de, Helmholtz-Zentrum Berlin, Germany
[2]julien.gout@uni-potsdam.de, University of Potsdam, Germany
[3]grothausmann.roman@mh-hannover.de and kuehnel.mark@mh-hannover.de,
Institute for Functional and Applied Anatomy, Hannover Medical School, Germany

Abstract

This document describes a plug-in for Blender (www.blender.org) that allows to rasterize 3D mesh objects into 3D voxel data, i. e. it allows to voxelize Blender meshes. In 2D, this process can be compared to rasterization of vector graphics into pixel graphics. The voxelization is done by VTK (www.vtk.org) functions. A simple GUI allows to choose the type of voxelization and to specify necessary parameters. Depending on the type chosen, only the surface of the mesh objects is voxelized or the enclosed volume, i. e."filled". Beside the size of the output, one can specify if the result should be "anti-aliased" or not.

This paper is accompanied with the source code, input data, parameters and output data that the authors used for validating the script described in this paper.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3391]

Contents

1 Introduction

Blender, being a very sophisticated 3D modeling software that allows handy mesh generation, deformation, editing and processing mechanisms, can be easily used to generate test data. Up to now, the 3D data could only be used and exported as vector data with e. g. the VTKBlender Module[1]. The described Blender plug-in enables to export this 3D mesh data (a continuous geometric representation) as discrete 3D pixel data or *voxel data*. This way it is possible to generate specific voxel data sets for testing image filters and analysis tools. The process can be regarded as 3D vector drawing followed by 3D rasterization.

2 Software Requirements

You need to have the following software installed:

- Blender (`www.blender.org` <2.50)
- Visualization Toolkit (> 5.4) (VTK, `www.vtk.org`) compiled with Python wrappers
- VTKBlender Module[1]

Optionally, for filled anti-aliased voxelization:

- TetGen[2]
- vtkPartialVolumeModeller[3]

Optionally, the accompanying patch can be applied to make vtkImplicitModeller behave as expected by the script.

In order to make VTK wrap vtkPartialVolumeModeller for Python, copy vtkPartialVolumeModeller.cxx and vtkPartialVolumeModeller.h into VTK/Graphics/, add `vtkPartialVolumeModeller.cxx` under `SET( Kit_SRCS` in VTK/Graphics/CMakeLists.txt and reconfigure, compile and install VTK.

We also intended to create a plug-in for Blender (> 2.50), however this was not possible because Blender (> 2.50) needs Python 3 while wrapping VTK with Python 3 is not yet possible.

3 Installation and Usage

If the script is copied into the script directory of blender `.blender/scripts/` it can be called within Blender from the Object menu (see fig. 1). The VTKBlender Module[1] should be placed in the same directory.

If VTK was compiled with /opt/python-2.6.2/ and installed in /opt/vtk/ the following environment variables should be set before starting blender (e.g. inside bash):

```
PYTHONPATH=/opt/vtk/lib64/python2.6/site-packages/
PYTHONHOME=/opt/python-2.6.2/
LD_LIBRARY_PATH=/opt/vtk/lib/vtk-5.10/
PATH=/opt/python-2.6.2/bin/:$PATH
```

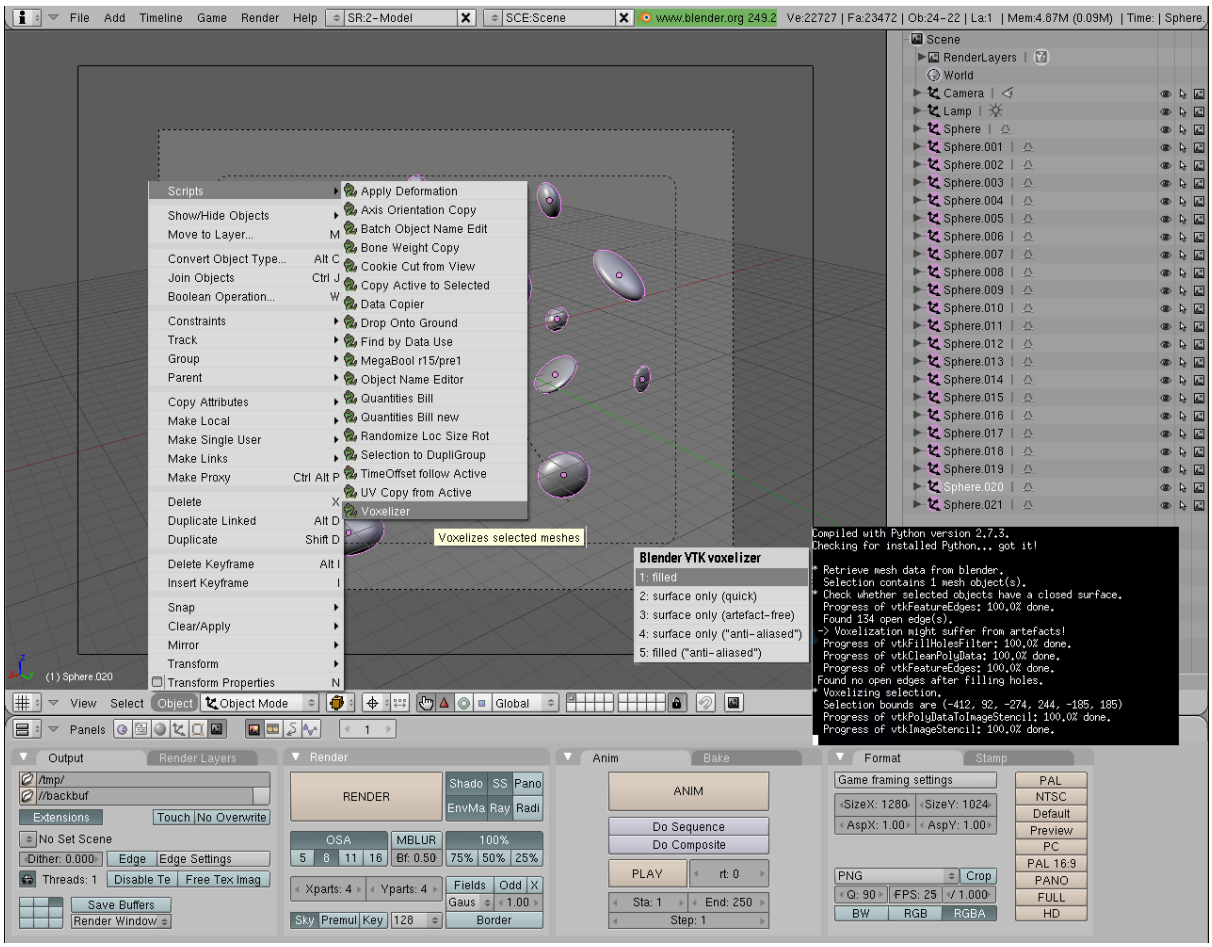

Figure 1: Ater installation, the voxelizer plug-in can be found under *Object→Scripts→Voxelizer*. The output file name of the MetaImage (*.mhd), the scale and the optional surface check can be specified in the parameter window of the plug-in. The voxelization of all selected mesh objects is started with the OK-button.

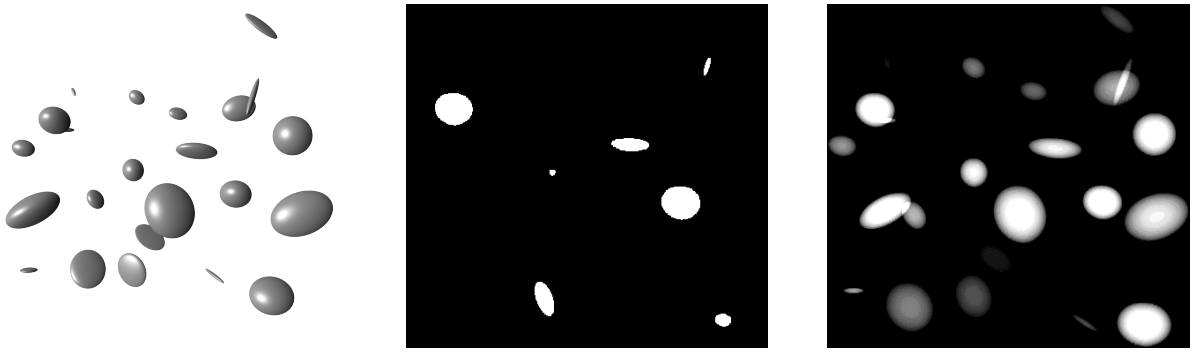

Figure 2: Figure showing the original Blender mesh objects, a slice and a z-projection of the voxelizer result.

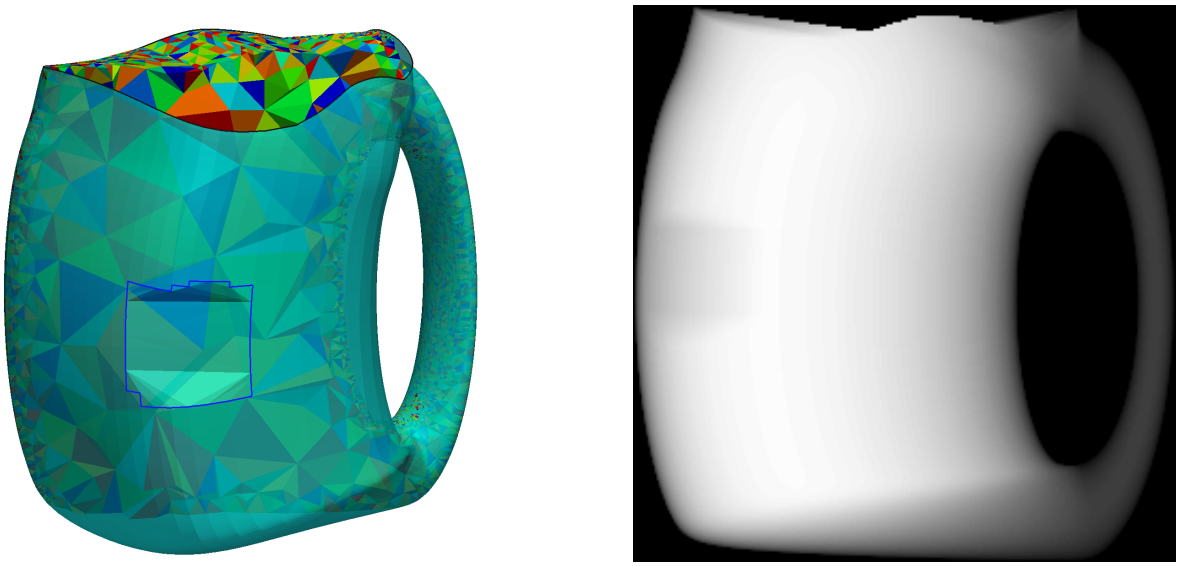

Figure 3: A “torus cup” with an additional hole (filled with $r \approx 1$, blue edges). The torus cup can be voxelized filled if the cup is closed as well ($r \approx 2$, black edges). The z-projection or the result is shown on the right. Using TetGen, the “handle” remains open. The quality tetrahedralization is partially rendered in the left image (tetrahedra colored arbitrary).

All selected mesh objects will be included in the voxelization process. The dimensions of the output will be taken from the dimensions within the blender scene and scaled by the specified scale factor. Optionally the meshes can be checked if they represent closed surfaces. This however, is no necessity for the chosen voxelization method. The progress of the script is reported on the command line as well as by the Blender progress bar.

For the demonstration we crated some UV-spheres and randomized their location, size and rotation with the Blender plug-in *Object→Scripts→Randomize Loc Size Rot*. Then applied the voxelizer plug-in *Object→Scripts→Voxelizer* with a scale of 20, see Fig. 1. A slice and a z-projection of the result are shown in Fig. 2.

4 Usage notes

In general, a closed, manifold mesh is needed for the creation of a volume or “filled” voxelization. A mesh which is not closed can be reagrded as an incomplete mesh, i. e. as a mesh with holes. The vtkFillHolesFilter can be used to close holes in none-convex meshes, e. g. a torus missing mesh cells or a mesh resembling a cup with a handle, see fig. 3.

Filled but not anti-aliased voxelizations (menu entry #1, see fig. 4) can be generated with vtkPolyDataToImageStencil. It does not need a closed mesh but open meshes are likely to cause artifacts in the result. We therefore included an option to check if the mesh is closed (with vtkFeatureEdges), i. e. has no open edges. The plug-in offers the option to close these with the vtkFillHolesFilter. During our test even fully closed meshes have shown artifacts in the voxelization. In this case it often helped to increase the output size (by increasing the scale value).

A filled voxelization can also be created with the itkTriangleMeshToBinaryImageFilter. We did not use itkTriangleMeshToBinaryImageFilter because this would make the plug-in dependent on ITK (wrapped for Python) and in our test itkTriangleMeshToBinaryImageFilter was much slower than the VTK method.

Filled anti-aliased voxelizations (menu entry #5) can be created with vtkPartialVolumeModeller[3]. It

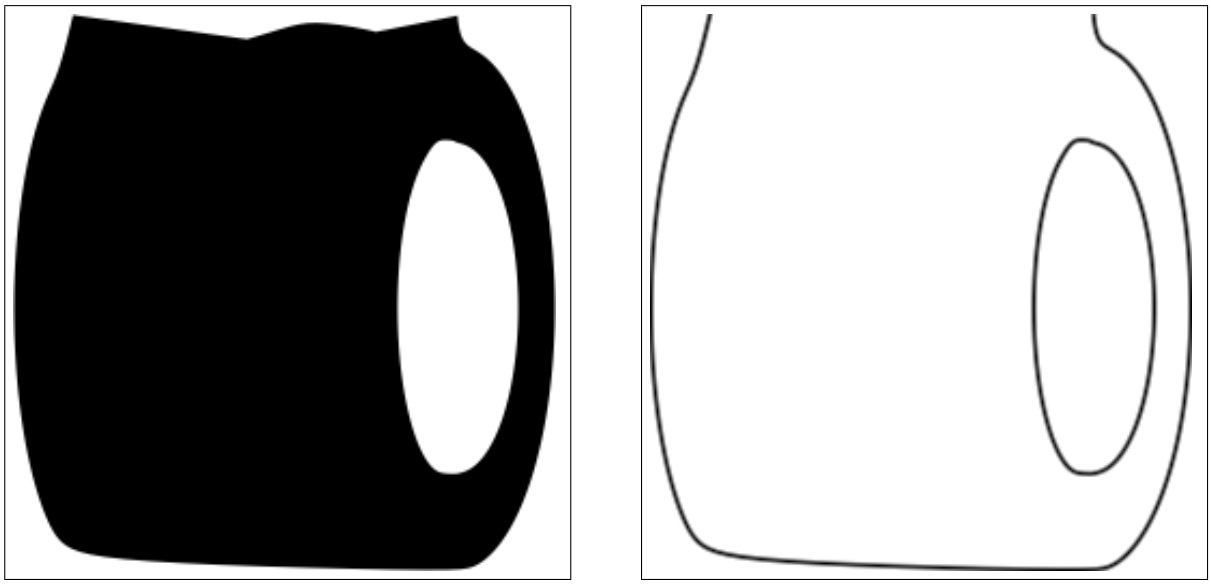

Figure 4: Slices of the “anti-aliased” voxelization of the “torus cup” from fig. 3. Left: volume voxelization, i. e. filled, right: surface voxelization.

needs a “volume mesh” consisting only of tetrahedra. This type of mesh cannot be directly created with blender since blender does not know volume elements. The necessary conversion from a surface representation (stored as vtkPolyData) to a tetrahedral representation (stored as vtkUnstructuredGrid) can be achieved with vtkDelaunay3D. However, vtkDelaunay3D only uses the vertices of the input mesh not the topology implied by the edges and triangles of the mesh. Therefore, vtkDelaunay3D does not preserve domain boundaries and would for example close the inner hole of a torus. TetGen[2] preserves domain boundaries and does not close the inner hole of a torus. Since TetGen is not available as a Python plug-in, we call TetGen (instead of vtkDelaunay3D) as an external program in this plug-in. TetGen can read and write vtkDataSet (`/tmp/voxelizer-tetgen_in.vtk`, `/tmp/voxelizer-tetgen_in.1.vtk`) which can then be used directly for the vtkPartialVolumeModeller (which does not report continuous progress information). Instead of using vtkPartialVolumeModeller (menu entry #5), combining the results of #1 and #4 can be quicker.

Voxelizing only the surface of a mesh (see fig. 4) can be achieved with vtkImplicitModeller or vtkVoxelModeller. In this case it does not matter if the mesh is not closed. Using vtkImplicitModeller yields a kind of “anti-aliased” result (#4). For vtkImplicitModeller a ‘wall thickness’ can be chosen. Its ‘per voxel mode’ is multithreaded and reports computation progress information, but has shown voxelization artifacts for large ‘wall thickness’. The plug-in interprets a negative ‘wall thickness’ to be in voxel units of the output. Positive values are taken to be in the mesh-units of the input. The plug-in expects the accompanying patch to be applied to vtkImplicitModeller. It works also without applying the patch but lead to unexpected results during testing.

A not anti-aliased surface voxelization can be achieved by thresholding the result of vtkImplicitModeller (#2) or with the vtkVoxelModeller (#3). The vtkVoxelModeller is much slower than vtkImplicitModeller in the tests. It is not multithreaded and does not report computation progress information, but has not shown any voxelization artifacts.

C++ errors cannot be caught within Python with a vanilla VTK installation. This can cause failure of the Python script which in turn crashes blender. This can happen if e. g. image.AllocateScalars() runs out of memory. There have been attempts to pass C++ errors to Python (see e. g. Ref. 4). This problem could be circumvented by calling a separate executable written in C++ that does the voxelization. This however would need the VTK mesh to be stored in a temporary file and would make both anti-aliased and not anti-aliased filled voxelization dependent on external programs.

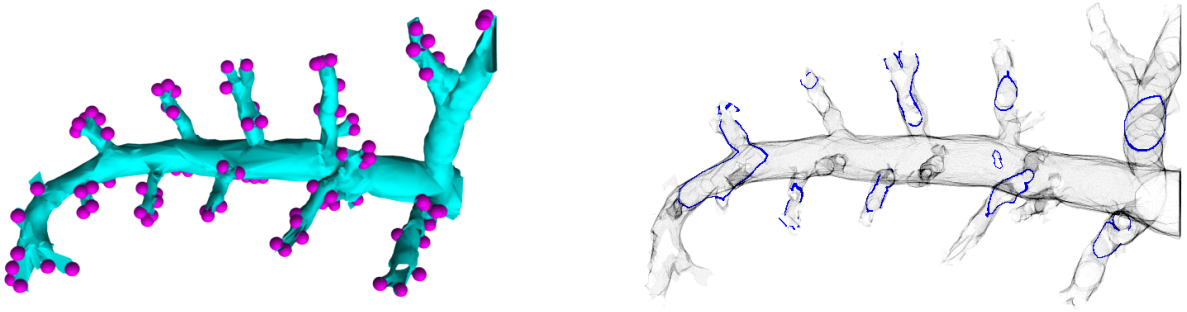

Figure 5: A Blender render of a mouse airway tree is shown in cyan on the left. The magenta spheres mark transition regions. The image on the right is a composite of a slice (blue) and the z-projection of the surface voxelizer result. Some openings are visible in the blue slice.

5 Application example

Fig. 5 shows a vector-graphic (mesh) model of the airways in a mouse lung. The magenta spheres mark the transition from air conducting structures to gas exchange structures. Constrictions in the transition region effect the lung functionality. Blender and the voxelizer plug-in allow to convert the original data for further analyses which depend on voxel data (in this case surface voxelization).

6 Performance

On an i3-2100 CPU @ 3.10GHz the voxelization of the torus cup (holes closed, scaled by 10, xyz: 26x27x20 voxel) took the following approximate execution times: #1: 9s; #2: 7s; #3: 123s; #4: 4s; #5: 463s.

7 Conclusions

The described plug-in for Blender allows easy voxelization of mesh objects created and modified within Blender. One can choose between a volume (filled) or surface (not filled) voxelization, each either anti-aliased or not. This allows easy creation of test voxel data sets. With the plug-in it is also possible to convert model (mesh data) from other programs into voxel data which then allows further processing with voxel filters or analyses.

8 Acknowledgment

We thank Cory Quammen for the vtkPartialVolumeModeller (his efforts to make VTK wrap it for Python and adding report of progress) and Hang Si for TetGen and their help to integrate TetGen and vtkPartialVolumeModeller into this project. We also thank the members of the vtk-users mailing-list for their help and support.

References

[1] Chris Want and Fritz Mielert. VTKBlender Module. open source, 1.19. URL http://www.ualberta.ca/CNS/RESEARCH/Vis/VTKBlender/. 1, 2, 3

[2] Hang Si. TetGen: A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator. open source, 1.4.3 (2011/01/19). URL http://tetgen.org. 2, 4

[3] Cory Quammen and Russell M. Taylor II. Grid voxelization with partial volume effects in VTK. *The VTK Journal*, (792), March 2011. URL http://hdl.handle.net/10380/3254. 2, 4

[4] Charl P. Botha. Patch for turning (almost) all VTK errors into Python exceptions. open source, 2006/08/08. URL http://vtk.1045678.n5.nabble.com/Re-patch-for-turning-almost-all-VTK-errors-into-Python-exceptions-IMPROVED-td1251918.html. 4