Iso-Surface Extraction and Visualization in Modular Visualization Environment

Marek Krejza[1], **Václav Skala**[2]

Department of Computer Science and Engineering
University of West Bohemia
Univerzitní 8, Box 314, 306 14 Plzeň
Czech Republic
mkrejza@students.zcu.cz skala@kiv.zcu.cz
http://iason.zcu.cz/~{mkrejza| skala}

Abstract

The Marching Cubes and Marching Tetrahedra algorithms are widely used to generate iso-surfaces of a 3D scalar fields. These methods are very popular for their good efficiency and because they produce triangular mesh that represents extracted iso-surface. This triangular mesh can be processed and rendered by standard graphics pipeline using hardware accelerators and OpenGL API. This approach enables to reach maximal speed of rendering.

This paper describes iso-surface extraction and rendering modules and interaction between them under the Modular Visualization Environment (MVE).

1. Introduction

1.1. About MVE

Modular Visualisation Environment (MVE) brings together dynamic computational and rendering tools in a programmable modules that enables users to rapidly create visualizations of highly complex data from disparate sources. Visualization is an often-critical component of deep computing analysis, combining with powerful computers and advanced algorithms to solve complex business and scientific problems. MVE consists of two parts: MVE-Editor and MVE-Runtime.

MVE defines program API, which allows programmers to make variety of modules. An application can be compiled from these modules in MVE-Editor, simply by associating inputs and outputs of the modules according to data types supported by these inputs and outputs. As the result we do not obtain executable application, but a scheme of the application, that can be executed via MVE-Runtime.

Modules are intended to work as threads. This make resulting application work better on computer systems with more than one processor - parallel.

2. Making modules

Module is simply DLL library that fills some requirements. In one DLL library can be more than one module. The name of the module is then passed as a prefix to each function to which it belongs. For each module must be specified following functions:

- `XXX_MAIN_MODULE_FUNC`
- `XXX_SETUP_FUNC`
- `XXX_FREE_DATA`
- `XXX_FREE_SETUP_DATA`
- `XXX_FREE_STATE`

[1] This work was supported by the Ministry of Education of the Czech Republic - project VS 97155
[2] This work was supported by the Academy of Sciences of the Czech Republic project - A2030801

where *XXX* represents module name. First function is the main module function. It take one input of any MVE supported type and produce one output of other type supported by the MVE. *SETUP_FUNC* is used to make some options before the main algorithm is started. Module which compute isosurface from the volumetric data uses this function for example to define starting treshold.

The last three methods are used to free memory previously allocated by module. There is information of size and type included in every data structure produced by modules. The module can use two special pointers to memory, where current state of algorithm and initial values are stored.

The DLL library has except this functions, another two:

- Get_Modules
- `Free_DLL_Descr`

This functions help MVE to register modules all modules in DLL library.

3. Modules

3.1. Data types currently supported

For computing iso-surface from volume data we need some data structures. The structures supported by MVE are:

- Common (basic data types)
- Volume types:
  used for working with volumetric data (CT, MRI ...)
- Triangle mesh
  This type can be passed for example as input to triangle viewer. And also this type is used as output of all modules computing iso-surfaces.

3.2. Loaders

Loaders are used by many other modules to load data from disk to the memory. There are two types of loaders finished yet.

The first one called Volume_Loader is used to open volumetric data file and load it to the MVE Volumetric structure in memory. Structure of file containing volume data is not so difficult.

There are several types of volume data files, according to the way they were acquired. We differentiate **CT** (Computed Tomography), **SPECT** {Single Photon Computed Tomography) and finally MR (Magnetic Resonance) volumetric data. We use special file (VIF - volume info file) together with data file, in witch we have described type of the file and information about object.

Second module can be used for reading triangles from file to the memory data structure. This module can read two types of files:

- TRI file format - specially designed for the MVE environment
- ASCII STL file format - one of the standard file format for triangles
- Binary STL file format – binary version of previous triangle format

The TRI file format has one advantage. When saving triangles to disk, first we have to save their vertices and then triangles represented by indexes to the array of previously saved vertices. This method saves memory and simplifying reading of the triangles to the triangle data structure in MVE.

3.3. Generating iso-surface from volume data

There are three main possibilities to display volume data. Display only orthogonal projection, generate a set of triangles that approximately represents the iso-surface of viewed object or make voluminous model of object for example by methods based on ray-tracing.

This paper discuss the second possibility - computing iso-surface, transform it to net of polygons (triangles) and display to the screen. The advantage of this method is, that there is a possibility to use hardware accelerators, for rendering generated polygons and of course data type on output of every method for generating iso-surface must be compatible with MVE data structures.

There are many methods to generate iso-surface from volume-data, but we will explain only three basic methods which are already implemented into the MVE. The methods are Marching Cubes, Marching Tetrahedra and Body-Centered tesselation scheme.

Three modules for computing iso-surface from the volumetric data are finished now. The modules are Marching Cubes, Marching Tetrahedra 5 and Marching Tetrahedra 6.

3.4. Marching Cubes

Marching Cubes is known and very popular method for generating iso-surfaces from a 3D scalar fields – volumetric data.

Advantage of this method is that is simple and fast. Produce acceptable amount of triangles in compare with the next method Marching Tetrahedra. Disadvantage is, that this algorithm can produce triangle mesh with holes. It is because of only eight values are used for interpolating iso-surface in the cell. Triangle mesh can have holes for example in case, that if two complementary cells are abreast. This is a special case, that triangles can share only vertices and not edges. The second disadvantage is, that the Marching Cube method can produce redundant triangles, degenerated to lines or points.

Output of this algorithm is iso-surface represented by set of triangles. The triangles can be rendered using hardware accelerators to optimize for speed. As rendering engine can be used for example OpenGL, Mesa, DirectX, or other.

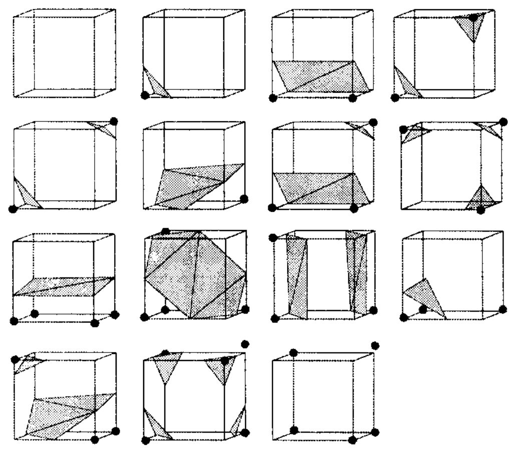


Figure 1: Fundamental cases in MC method

3.5. Marching Tetrahedra 5,6

Methods Marching Tetrahedra 5 and 6 are trying to minimize disadvantages of method Marching Cubes. Especially holes in the triangle surface.

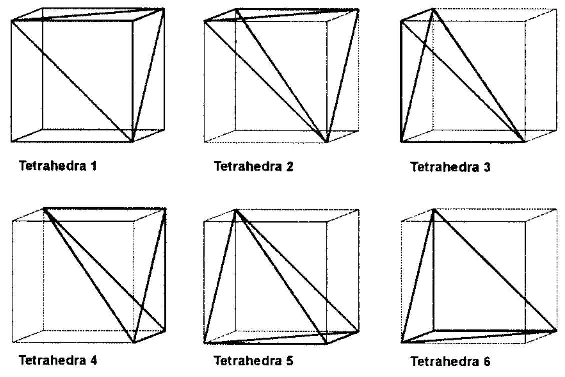


Figure 2: Possible decomposition to 6 tetrahedra

Instead of computing triangle vertices on edges of cell (as Marching Cubes does), Marching Tetrahedra divide the cell to 5 or 6 tetrahedra - MT5 (Marching Tetrahedra 5) or MT6 (Marching Tetrahedra 6). Triangles vertices are then computed on edges of this tetrahedra.

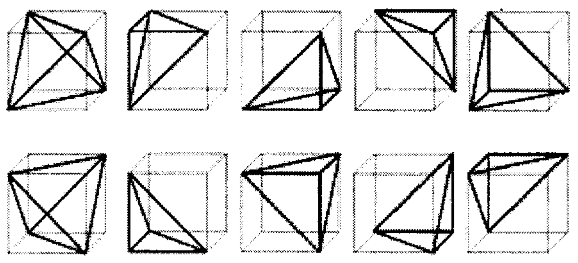


Figure 3: Two decompositions to 5 tetrahedra

3.6. Body centered scheme

The body-centered scheme use special way to divide cell to 24 tetrahedrons as shown on picture 1. This method has some advantages on Marching Tetrahedra algorithm.

- All tetrahedra obtained this way have the same shape although their orientations differ
- Decomposition of a cube to tetrahedrons is symmetrical.

- Every tetrahedron is shared between two neighbouring cells.

Improvement of this method consist in special dividing cube into tetrahedrons, which ensures better traceability of produced triangles and of course resulting triangles better interprets the iso-surface – are more regular.

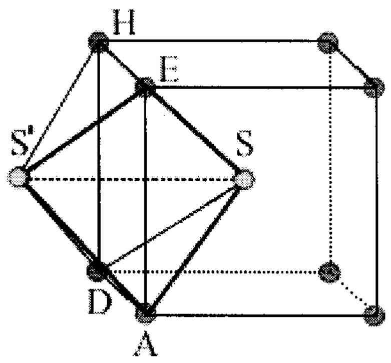


Figure 4: This picture shows how is a cube divided to 24 tetrahedrons by method body-center

4. Rendering

One of the most helpful module in the MVE environment is Triangle_Renderer. It is designed to view triangle data with all, what belongs to it. Renderer use OpenGL as a graphic engine. Now it has many options and is optimized for speed as much as it is possible for the structures in MVE.

Some of the options are:

- Gouraud/Constant shading
- Support for OpenGL List function
- Parallel/Perspective projection switching
- Triangle/Line/Points switch
- Lights on/off
- Axis on/off
- Changing normal direction of data
- Turn on/off computing unit normals
- Optional use of OpenGL CullFace parameter
- Optional use of OpenGL SingleSide parameter
- Color options
- Stereo viewing on card hardware witch supports stereo mode (tested Intergraph Intense/Wildcat)
- Export single BMP files, export set of BMP files - animation
- Export TRI and STL file formats data
- Textures mapping
- Environmental texture mapping
- Some informational functions

5. Results

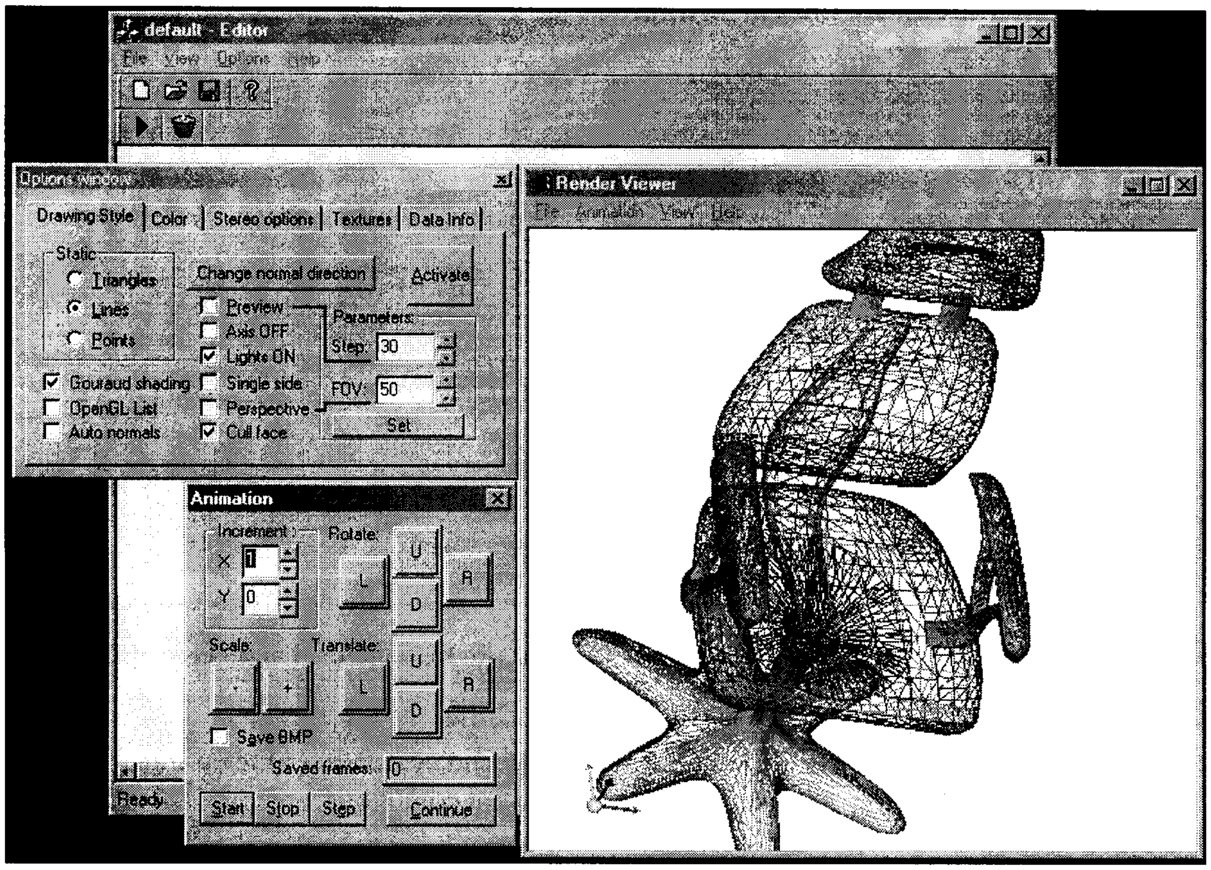
The memory and speed requirements of algorithms for generating of iso-surfaces are high. It is because of greater size of input and output data. Also for using special extends of OpenGL – for example stereo projection or vertex array list, we need hardware that support OpenGL v1.2 specification. Since most of systems does not support all of the 1.2 specification, can be these features turned off.

Hardware that looks to work properly is listed below.

- PC workstation HP Vectra XU, dual Pentium Pro 200 MHz, 128 MB RAM, 4 GB HD – without hardware accelerated OpenGL
- PC workstation HP Vectra XW, Pentium Pro 200 MHz, 128 MB RAM, 4 GB HD 5 – without hardware accelerated OpenGL
- Intergraph TDZ-2000 GL2, PentiumII 400 MHz, 512 MB RAM, HDD 9.1 GB Ultra Wide SCSI, RealiZm II VX113 and Vildcat graphics adapters
- Dell Precision - PentiumIII 450 MHz, 256 MB – 1GB RAM, 6.4 GB HD, Diamond Viper 770

For stereo viewing are used 3D glasses Crystal Eyes with emitter. The fastest graphics accelerator seems to be Intergraph Vildcat.

5.1. Pictures of MVE environment



Static: Triangles [✓ Lines] Points

[✓] Gouraud shading
[ ] OpenGL List
[ ] Auto normals
[ ] Preview
[ ] Axis OFF
[✓] Lights ON
[ ] Single side
[ ] Perspective
[✓] Cull face

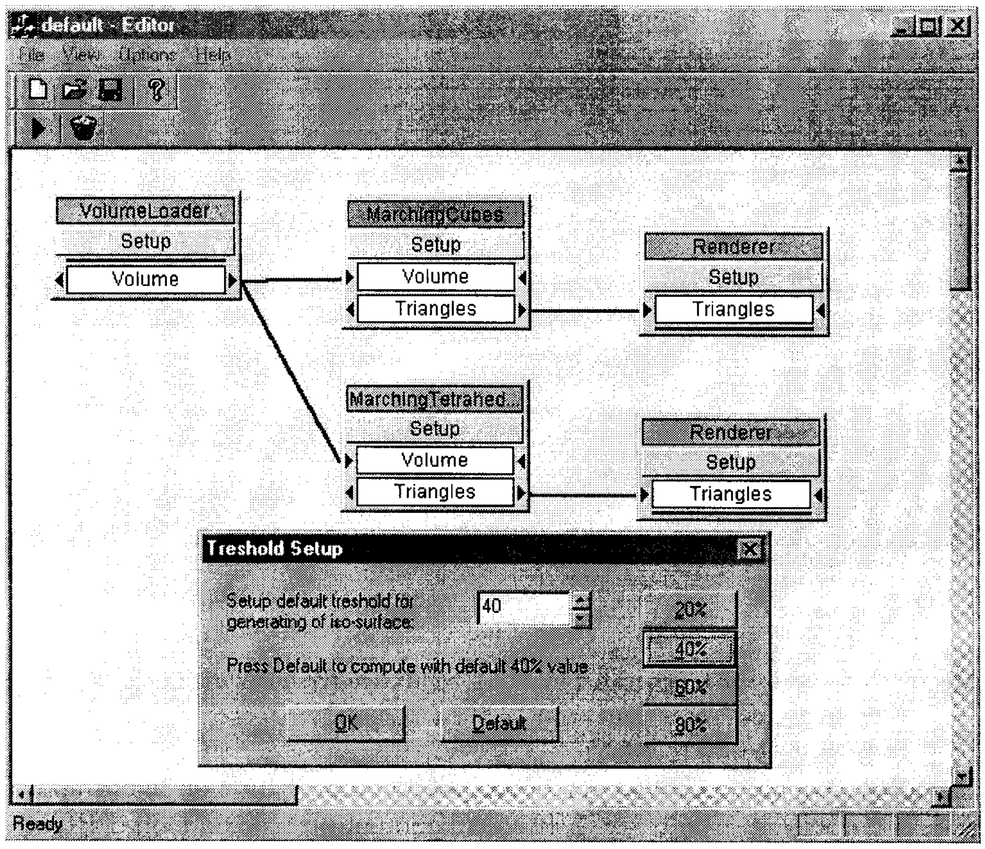
Parameters: Step: 30 FOV: 50 Set

Increment: X 1 Y 0

[ ] Save BMP

Saved frames: 0

Figure 5: Renderer - model of chair rendered as line model

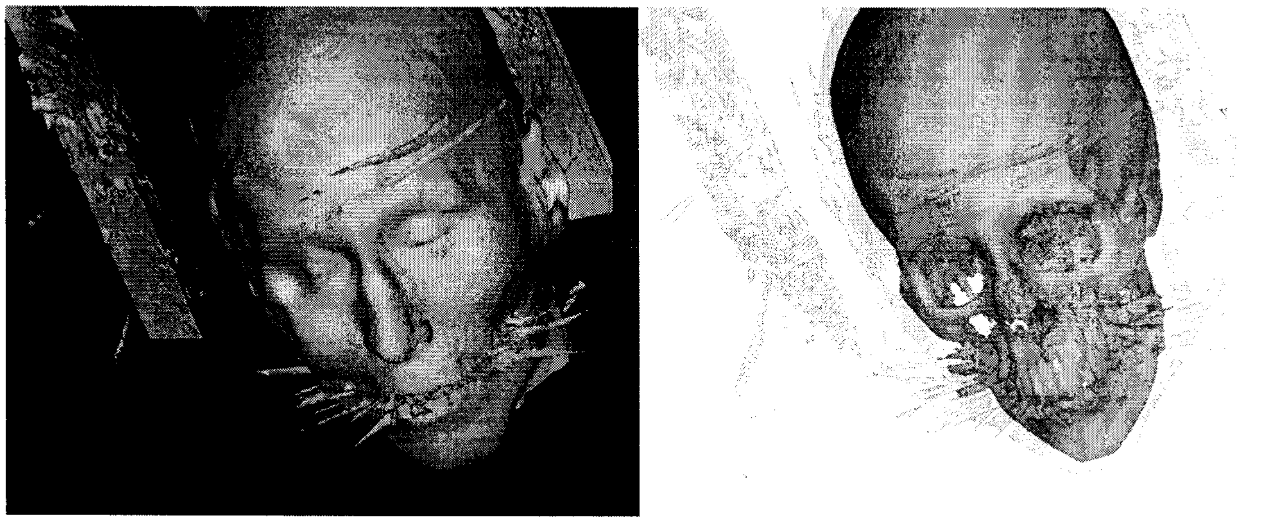


Setup default treshold for generating of iso-surface: 40

Press Default to compute with default 40% value

Figure 6: Integration volume modules into MVE environment

5.2. Output from Marching Cubes algorithm



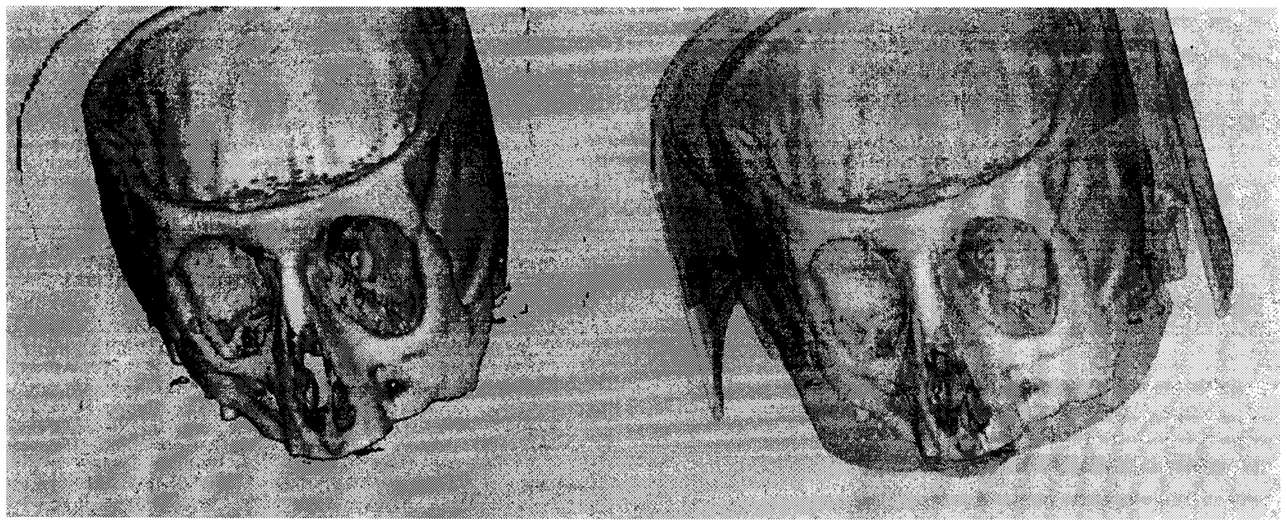
a) b)

Figure 7: CTHead.vol (256x256x108) generated by MC – right: two tresholds used together with alpha channel

Numbers of triangles for data on Fig.7.a

- 578,296 triangles (Marching Cubes) – 9,10 second computing time on Intergraph (PII)
  7,7 second computing time on Dell (PIII)
- 1,428,630 triangles (Marching Tetrahedra 5)
- 1,785,076 triangles (Marching Tetrahedra 6)

For data on Fig.7.b approx. Two times more triangles were obtained due to two iso-surfaces displayed



a) b)

Figure 8: CTMayo.vol (128x128x128) generated by MC – right: two tresholds used together with alpha channel

Numbers of triangles for data on Fig.8.a:

- 196,206 triangles (Marching Cubes) – 3,14 second computing time on Intergraph (PII)
  2,4 second computing time on Dell (PIII)
- 487,946 triangles (Marching Tetrahedra 5)
- 603,074 triangles (Marching Tetrahedra 6)

6. Conclusion

Modules for generating iso-surfaces Marching Cubes and Marching Tetrahedra optimized to work under MVE environment was presented.

With other helpful modules such as Triangle_Renderer can be simply made application for rendering volumetric data to the screen. The user of this software can include his own modules between Volume_Loader and Marching_Cubes modules in the application scheme, which use for example FFT (Fast Fourier Transformation) to adjust volume data before MC method is called. Also another module can be used, which reduce number of triangles on output of triangle mesh generators and then smaller amount of triangles is passed to the Triangle_Renderer – result is better speed and less memory requirement.

7. Acknowledgement

The authors would like to thanks to all who contributed to this work, especially to MSc. and PhD. students, who prepared others modules and use the MVE, at the University of West Bohemia in Plzen, to all who have stimulated this work and development. Theirs invaluable critical comments and suggestions improved the manuscript significantly.

References

[1] Bartos,P.(supervisor Skala,V.): Volumetric Data and Surface Models (in Czech), MSc. Thesis, Univ.of West Bohemia, Plzen, Czech Republic, 1999.

[2] Baxa,P., Skala,V., Moucek,R.: Error Estimation for Iso-surfaces, In Proceedings of COMPUGRAPHICS '97, 1997.

[3] Zara, J.: Moderni pocitacova grafika, Computer press, 1998

[4] Csebfalvi,B.: An Incremental Algorithm for fast Rotation of Volumetric Data, TR-186-2-99-07, 1999

[5] Chan, S., Purisima, E.: A New Tetrahedra Tesselation Scheme for Isosurface generation, Computers&Graphics, Vol.22, No.1., pp.83-90, 1998

[6] Treece, G., Prager, R., Gee, A.: Regularised Marching Tetrahedra: improved iso-surface extraction, Computers&Graphics, Vol.23, pp.583-598, 1999