CMST 15(2) 151-160 (2009)

DOI:10.12921/cmst.2009.15.02.151-160

COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY **15(2)**, 151-160 (2009)

Visualization of Results Received with the Discrete Element Method

Robert Kostek[1], Antonio Munjiza[2]

University of Technology and Life Sciences, Faculty of Mechanical Engineering
al. Prof. S. Kaliskiego 7, 85-796 Bydgoszcz, Poland
e-mail robertkostek@o2.pl

Queen Mary, University of London, Department of Engineering, 328D
Mile End Road, London, UK, E1 4NS
e-mail: a.munjiza@qmul.ac.uk
Website: http://webspace.qmul.ac.uk/amunjiza

(Received: 03 May 2009; revised: 22 September 2009; accepted: 02 October 2009; published online: 26 November 2009)

Abstract: The output of discrete element simulations includes thousands of time-frames and millions of interacting particles (bodies) in each frame. A single simulation can include terabytes of results, which may lead to very large, very complex data sets. Visualizing them is coupled with difficulties caused by either the number of particles, the number of time-frames or complexities of "system variables". In this work, an attempt has been made to present a data format and graphical template dedicated to discrete element visualization. The article presents a practical approach to the issues, that is the result of programming an open source 3D visualizer, called DEV_KM.

Keywords: data format, DEM, visualization

I. INTRODUCTION

Discrete Element Methods (DEM), sometimes called Molecular Dynamics (MD) or Particle Method (PM), are numerical methods used to solve a wide range of engineering and scientific problems. The fundamental assumption in these methods is that the material consists of large assemblages of separate, discrete particles, e.g. atoms, grains of sand, and stones [1, 2]. The number of the discrete particles can be a few million [3] and even billion [4]. These particles may have different shapes and properties. Complex non-linear interactions between bodies, and within bodies, are simulated with a numerical method. Next, the motion of particles, which is described by differential non-linear equations, is computed. Finally, the motion of a large number of particles, like molecules [5] or grains of sand, is presented as a movie [2, 3, 6-8] and analysed. The methods have significant applications in: mineral processing [1], rock blasting, crushing, powder technology and many more areas. Materials like gases, liquids, solids (powder, sand or rock) can be simulated with the method.

Summarising, the basic idea of modelling bodies, as large assemblages of separate – discrete elements, makes the method very universal; because each body can be modelled as an assemblage of separate atoms, or groups of atoms. A separate discrete element can represent an atom, a group of molecules, a rigid body or FEM primitive. In turn phenomena like: evaporation, laser processing, granular flow of stones, or blow up of building can be modelled.

Visualization is an interdisciplinary field of knowledge concerning presentation of information as images, diagrams, and animations. Visualization plays a key role in DEM, because of a large number of particles. Visualization makes interpretation of the results [9] and verification models [6] easier. Nevertheless, because of the large number of frames, particles (e.g. 4.5 million [3]) and large data sets [9], it becomes problematic [10]. Thus parallel computing is sometimes applied for visualization DEM [4, 9, 10]. The key issues for DEM visualizations are data format, allocation of RAM, and graphical template (colours, lighting, etc.). Thus, there is a need to describe the issues. The article presents practical approach to the discrete element visualization with a standard PC.

II. A NEW DATA FORMAT

The ways of result presentation depend on considered problems. Usually, various problems are considered in different fields, thus visualizations vary [1-21]. In addition, diverse parameters are important in simulations; sometimes scalar [3, 7, 8] vector [3, 8, 11, 12] or tensor fields [13] are presented, while trajectories of bodies [10] or streamlines [10-12 14, 17] can be important in some cases. Various methods of presentation and modelling bodies are applied as well. Bodies can be represented by a number of triangles [2, 8, 15, 16], or spheres [8, 9, 18, 19]. Nevertheless, all visualizers use the same primitives. Thus, the idea of the presented data format is based on the presentation of an individual primitive, and the result received for the primitive. All information is located in different directories, in separate files, according to:

- time,
- kind of information, and
- type of primitive.

That format allows for quick searching of large data sets, because information is clearly stored in separate files. Thus, certain part of data is not analysed at all. In addition to this, a successive presentation of primitives (one by one) gives the opportunity on the presentation of large amounts of data, with low RAM use. This is because a relatively small amount of RAM is allocated to the process. The amount of RAM depends on a number and type of primitives being processed in a certain time. It should be mentioned that the number of primitives being processed in a certain time can be smaller than the number of primitives being presented on a scene. After presentating one group of primitives, the next group is presented. The data format dedicated to the discrete element method is a relatively resent issue. An example of the format is presented below.

Data received from a simulation are located within one folder, for example "**Beam**", in different directories (Fig. 1). The data are divided between "**FRAME**" folders (Fig. 1a), according to time. Each "FRAME" folder contains information about the system saved for a single time instance. The "FRAME" folder contains folders like: "**Geometry**", "**Tensor**", "**Vector**" and "**Scalar**" (Fig. 1b), and others as chosen by the user. Here the data are divided according to the kind of information. If a new data format (kind of information) is expected, a new folder of any name can be created. The "Geometry" folder contains information about the geometry of the primitives, which is located in separate files according to the type of primitives. The "Tensor" folder contains folders like "Stress" and "Strain" (Fig. 1c). All information about tensors is saved in the "Tensor" folder. The "Vector" folder contains folders like "Velocity" and "Acceleration" (Fig. 1c). Thus, information about all vectors is written in the "Vector" folder. The "Scalar" folder contains folders such as: "Temperature", and "Density" (Fig. 1c). In other words, all information about any scalar fields, that is the result of simulation, is contained in the "Scalar" folder. In addition, all information about the density field is contained in the "Density" folder, in separate files according to the type of primitives (Fig. 1d).

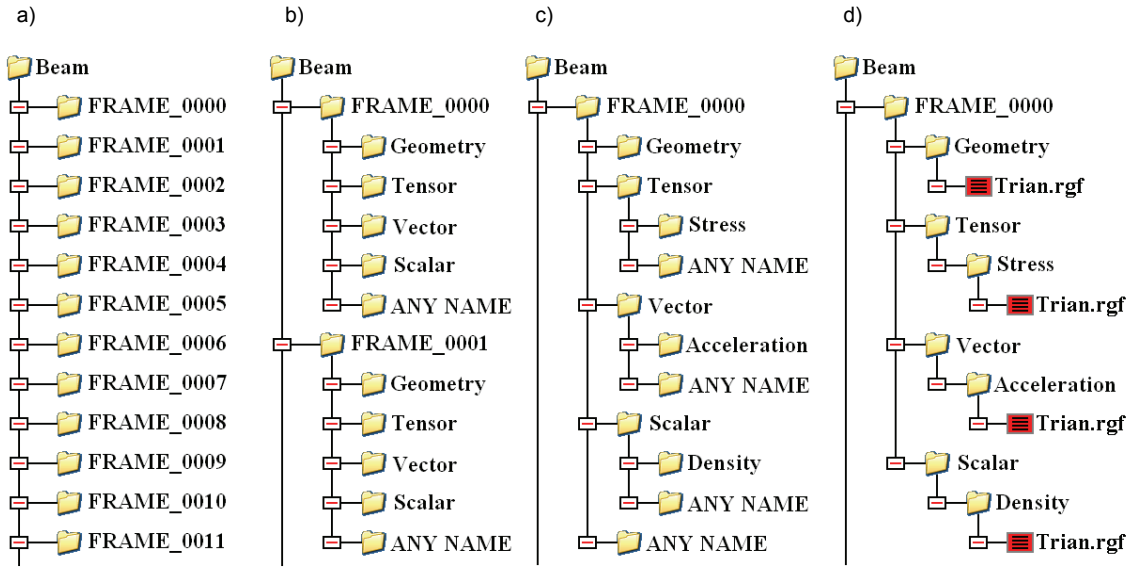


Fig. 1. Structure of the data; (a) FRAME folders, (b) data within FRAME folders, (c) structure of Tensor Vector and Scalar folders, (d) **Trian.rgf** files within folders

Graphics primitives used in the program are: point, line, **triangle**, quadrilateral, polygon 5-99, sphere and tetrahedron. Following the aforementioned idea of the data format, information about **triangles** is located in various directories like (Fig. 1d):

- …\Beam\FRAME_0000**Geometry\Trian**.rgf,
- …\Beam\FRAME_0000**Tensor\Stress\Trian**.rgf,
- …\Beam\FRAME_0000**Vector\Acceleration\Trian**.rgf,
- …\Beam\FRAME_0000**Scalar\Density\Trian**.rgf.

Information describing the other primitives is stored in a similar way:

- …\Beam\FRAME_0012**Geometry\Polygon_006**.rgf,
- …\Beam\FRAME_0012**Tensor\Stress\Polygon_006**.rgf,
- …\Beam\FRAME_0012**Vector\Velocity\Polygon_006**.rgf,
- …\Beam\FRAME_0012**Scalar\Density\Polygon_006**.rgf.

Examples of the files are presented bellow. Geometry of triangles is saved for instance in the directory – …\Beam\FRAME_0012**Geometry\ Trian**.rgf. The structure of the file is shown in Fig. 2. The file contains a few lines of comments followed by the data, obtained from discrete element-based simulation software. The first column contains the ID of a primitive, while the next three columns contain coordinates of nodes of the triangle, followed by normal vectors at nodes. The last column contains some extra information, e.g. the type of the object. Thus information about one triangle takes the three lines. In a similar way data for points, lines, tetrahedrons and other primitives are stored. A tetrahedron is presented as four triangles.

```
! FRAME_0012\Geometry\Trian.rgf
!nn = 1 (elastic), nn = 2 (rigid), nn = 4 (softening)
! ID    X      Y      Z      Nx     Ny     Nz     nn
  0     1e+0   0e+0   0e+0   0e+0   0e+0   1e+0   1
  0     2e+0   1e+0   0e+0   0e+0   0e+0   1e+0   1
  0     3e+0   2e+0   0e+0   0e+0   0e+0   1e+0   1
  1     4e+0   3e+0   0e+0   0e+0   0e+0   1e+0   2
  1     5e+0   4e+0   0e+0   0e+0   0e+0   1e+0   2
  1     6e+0   5e+0   0e+0   0e+0   0e+0   1e+0   2
  2     7e+0   6e+0   0e+0   0e+0   0e+0   1e+0   4
  2     8e+0   7e+0   0e+0   0e+0   0e+0   1e+0   4
  2     9e+0   8e+0   0e+0   0e+0   0e+0   1e+0   4
  ...
```

Fig. 2. **Trian.rgf** file containing geometry data for triangles

An example of Spheres.rgf file which is located in the directory …\Beam\FRAME_0012**Geometry\Spheres**.rgf, is shown in Fig. 3. The first column contains the ID of a sphere, while the next three columns contain information about the position of the centre of sphere. The following column contains the radius of sphere and the last one contains some extra information.

```
! FRAME_0012\Geometry\ Spheres.rgf
!nn = 1 (elastic), nn = 2 (rigid), nn = 4 (softening)
! ID    X      Y      Z      R      -      -      nn
  0     1e+0   3e+0   1e+0   5e+0   0e+0   0e+0   2
  1     2e+0   2e+0   2e+0   2e+0   0e+0   0e+0   2
  2     3e+0   1e+0   3e+0   3e+0   0e+0   0e+0   4
  3     1e+0   3e+0   1e+0   1e+0   0e+0   0e+0   1
  4     2e+0   2e+0   2e+0   2e+0   0e+0   0e+0   2
  5     3e+0   1e+0   3e+0   3e+0   0e+0   0e+0   4
  6     1e+0   3e+0   1e+0   1e+0   0e+0   0e+0   1
  7     2e+0   2e+0   2e+0   2e+0   0e+0   0e+0   2
  8     3e+0   1e+0   3e+0   3e+0   0e+0   0e+0   4
  ...
```

Fig. 3. **Spheres.rgf** file containing geometry data for spheres

Stress tensors for triangles are written in the Trian.rgf file (Fig. 4). This file is stored in the …\Beam\FRAME_0012\ **Tensor\Stress\Trian**.rgf directory. The first column contains the ID of a primitive and the following columns contain tensor data at nodes of the primitive (triangle).

```
! FRAME_0012\Tensor\Stress\Trian.rgf
! Stress tensor
! ID    Sxx     Syy     Szz     Sxy     Syz     Szx
  0    -1e+0    3e+0   -1e+0    0e+0    0e+0    0e+0
  0    -1e+0    3e+0   -1e+0    0e+0    0e+0    0e+0
  0    -1e+0    3e+0   -1e+0    0e+0    0e+0    0e+0
  1     0e+0    0e+0    0e+0    0e+0    0e+0    0e+0
  1     0e+0    0e+0    0e+0    0e+0    0e+0    0e+0
  1     0e+0    0e+0    0e+0    0e+0    0e+0    0e+0
  2     2e+0    3e+0   -3e+0   -2e+0    0e+0    0e+0
  2     2e+0    3e+0   -3e+0   -2e+0    0e+0    0e+0
  2     2e+0    3e+0   -3e+0   -2e+0    0e+0    0e+0
  ...
```

Fig. 4. **Trian.rgf** file containing stress data for triangles

In a similar way the information about velocity for a triangle is contained in file ...\Beam\FRAME_0012\ **Vector\Velocity\Trian**.rgf as shown in Fig. 5. For any other primitive information about velocity is located in a specific file, in directory ...\Beam\FRAME_0012**Vector\ Velocity\Name-of-Primitive**.rgf.

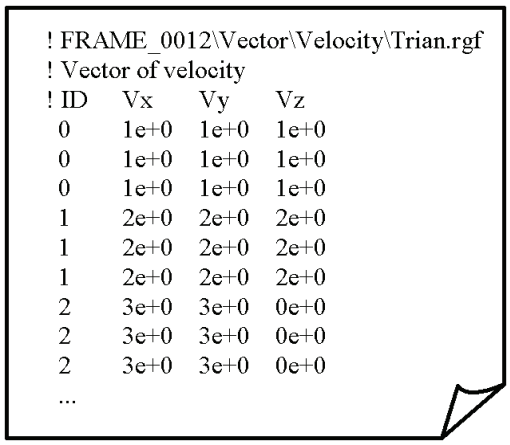


Fig. 5. **Trian.rgf** file containing velocity data for triangles

All scalar fields for presented primitives are stored in specific directories –...\Beam\FRAME_0012**Scalar****Name-of-Scalar****Name-of-Primitive**.rgf. For instance, density of triangles is stored in the Triangle.rgf file, in directory ...\Beam\FRAME_0012**Scalar****Density****Triangle**.rgf (Fig. 6). The file contains a few lines of comments followed by the data, obtained from simulation. The first column contains the ID of a primitive, while the last column contains values of the scalar field, for a particular node of a primitive.

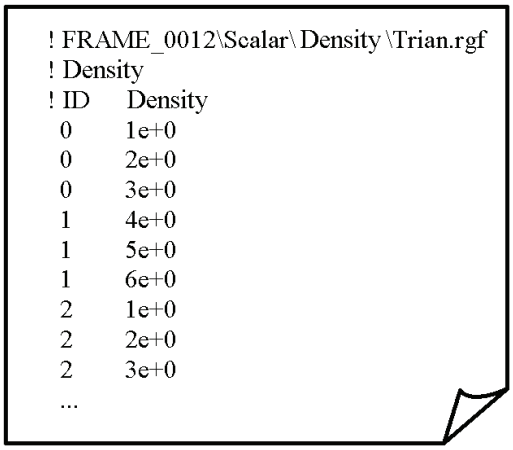


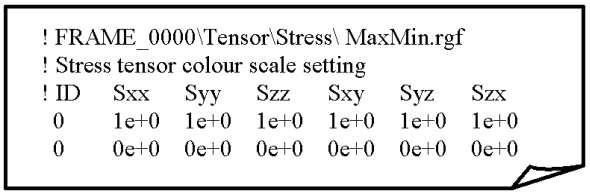
Fig. 6. **Trian.rgf** file containing information about density of triangles

In order to properly apply colour scale, the maximum and minimum values of fields need to be supplied. This is done through MaxMin.rgf files located in the corresponding subfolder of the FRAME_0000 folder:

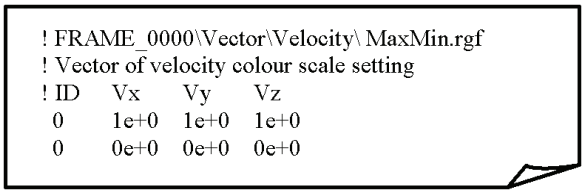
- …\Beam\FRAME_0000**Tensor\Stress\MaxMin.**rgf,
- …\Beam\FRAME_0000**Vector\Velocity\MaxMin.**rgf,
- …\Beam\FRAME_0000**Scalar\Speed\MaxMin** rgf.

Examples of the aforementioned files for: stress tensor, velocity vector, and speed are shown in Fig. 7. Associated with FRAME_0000 is also the NoFrames.rgf containing the total number of frames (Fig. 8).

a



b



c

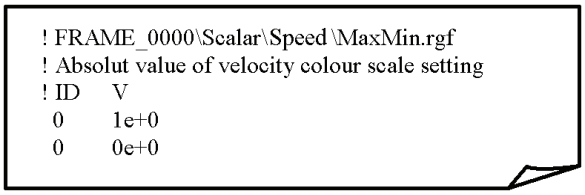


Fig. 7. **MaxMin**.rgf files containing data which describe maximum and minimum values of: a) stress, b) velocity, c) speed

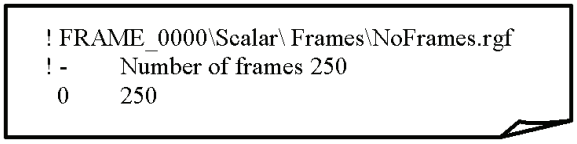


Fig. 8. Structure of a **NoFrames**.rgf file containing information about a number of frames

Summarising, despite huge amounts of information, the structure of data makes searching easy, because information is located clearly in particular directories. The data format is very clear and open. Thus, the approach to the data deposition makes for easy future modification. Users, according to individual needs, can modify the names of folders, the names of files and file format.

The structure of the data gives the opportunity to reduce the amount of RAM allocated to the visualization process, because a limited amount of data is processed in a certain time. Only information about one type of primitive and selected fields received for the primitives is visualized in a certain time.

In contrast to this one, the VTK format allows to divide information according to time; consequently all information about the geometry of the primitives and the results received for the primitives (scalar, vector and tensor fields) is stored in one file [20]. Moreover, information about geometry of a presented system is recorded in three parts: coordinates of nodes, types of primitives, and numbers of nodes associated with a primitive. That forces to store all presented information in RAM. Furthermore, the VTK format does not support spheres; consequently, they are represented by a number of polygons, which is less effective. That leads to problems with visualization of large data sets with PC whose RAM is usualy limited to 3 GB.

III. GRAPHICAL TEMPLATES

The influence of colours on the visual effect is significant. Colours influence both the perception of results, interpretation of results, and also have an aesthetical effect. The proper setting of colours makes the interpretation of a scalar or a vector field easier. Ideally, separate bodies should be contrasted to a background and also to each other. A typical colour scale contains striking colours such as red, yellow, green, blue, and sometimes purple (spectrum scale [21]) (Figs. 9, 10). The colour scale with purple seems to be more convenient for successful visual analysis results because of a wider range of colours. The colour scale makes for the interpretation of information associated with colours. It should contain a short description of the quantity, unit, and the values of the data range (Figs. 9, 10). The colour scale should be continuous, like a spectrum, to allow an interpolation value. Background, in contrast to the colour of the bodies, should be neutral, thus grey is often chosen. In this colour set the white and black colours can be used to present some extra information, e.g. comments, cracks and nodes. In short, the proper setting of colours makes the scene plain and nice looking (Figs. 9, 10).

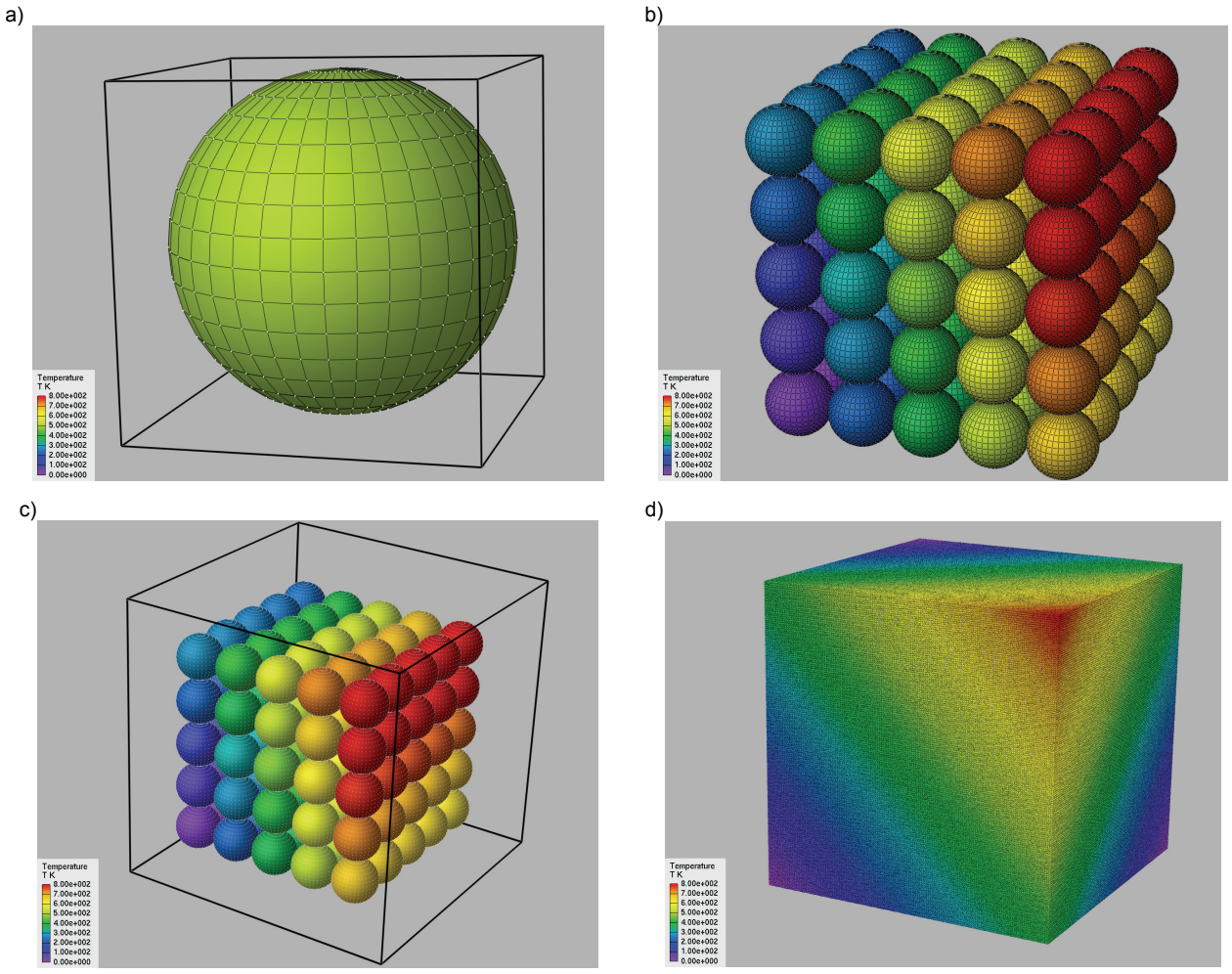


Fig. 9. The use of perspective projection (a-d), shading (a-d), presentation of mesh (a,b) and nodes (a-c), for enhancing the perception of the 3D shapes, of particle-based models, (d) 4×10^6 spheres – 2.6×10^9 quadrangles

The next very important factor is the presentation of shape. Proper lighting plays an important role in this, see [9]. Illumination effects like chiaroscuro (Figs. 9, 10) can enhance the perception of complex shapes. Presenting nodes (Fig. 9a-c) or mesh (Fig. 9ab), on the surfaces of bodies, and the rotation of scene also reveals a shape. Perspective projection, as the most natural projection for people, plays an important role in the process as well (Fig. 9). If the aforementioned issues are neglected, then the scene looks flat, and the interpretation of results becomes difficult. In other words, all the aforementioned details are essential for successful shape interpretation.

Proper lightning, shading and colour selection improve the quality of a picture; however, the aim is not to make photorealistic pictures, but to present results. Thus, quality should be matched to the aim. For instance, advanced effects like reflections may lead to misinterpretation. A white particle may appear pink if it reflects light from a nearby red particle [9]. Legible pictures or movies should not contain too much information which can lead to misinterpretation and cluttering of the view space. For example, visualization vectors or streamlines can lead to easier (Fig. 10) or more difficult interpretation of the results. These and other factors are taken into account when results are visualized. Some examples of these are shown in Figs. 9, 10.

IV. COMPARISON OF VISUALIZATION RECEIVED WITH ParaView AND DEV_KM

In this section a number of spheres has been visualized with ParaView and DEV_KM. The following PC computer has been used: system WinXP, processor AMD Athlon 64 2800+, 1 GB RAM, graphic card ATI Radeon 9550. Previously it has been mentioned that the VTK format does not support spheres, thus they are represented by a number of polygons. In turn, the *.vtk files are larger than the *.rgf

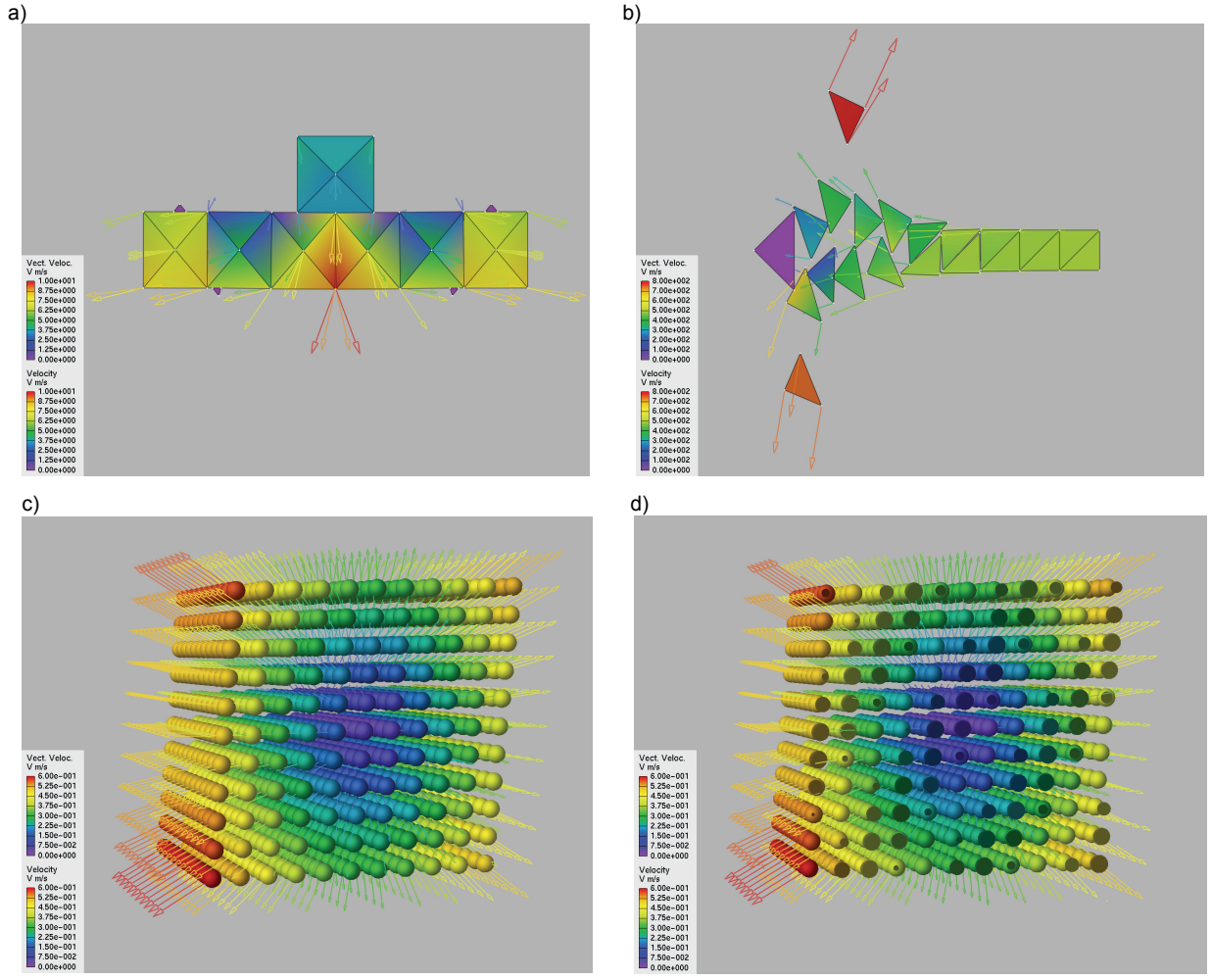


Fig. 10. Results of simulation with the discrete element method a) collision of elastic bodies, b) collision of rigid particles, c) explosion of a number of spheres, d) a cross section of the spheres

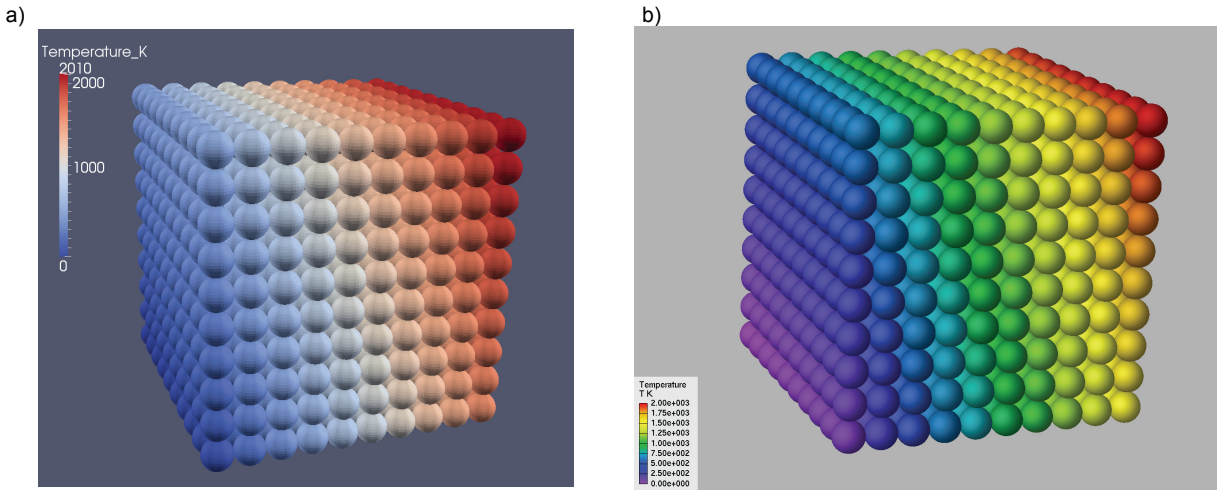


Fig. 11. Visualization of 1000 spheres received with ParaView 3,6,1 (a) and DEV_KM (b)

files (Tab. 1). Each sphere has been represented by 648 quadrangles, both with ParaView and DEV_KM, to enable the comparison of quality offered by the two programs and time of visualizations. An analysis of the received results tends to tends to a conclusion that for the considered examples the visualization with DEV_KM is more effective than with ParaView (Tab. 1). Moreover, ParaView is not able to present 17 496 000 quadrangles with the PC. The quality of visualization offered by the two programs is satisfying (Fig. 11); however, DEV_KM visualization seems to be more legible.

Table 1. Comparison of *.vtk and *.rgf files, and time of their processing with ParaView and DEV_KM

Visualized primitives		Size of ASCII files		Time of rendering, s	
Number of quadrangles ***.vtk**	Number of spheres ***.rgf**	***.vtk**	***.rgf**	ParaView 3,6,1 ***.vtk**	DEV_KM ***.rgf**
648 000	1 000	66.1 MB	64 KB	21	2
5 184 000	8 000	550 MB	525 KB	168	10
17 496 000	27 000	1.85 GB	1.76 MB	∞	34

V. CONCLUSIONS

A novel data (graphical) format dedicated to discrete element methods has been presented. The basic novelty of the format is breaking the geometry into primitives, and attaching various fields to each of the primitive types. In addition, the temporal discretisation of the data is employed through a large number of time instances being recorded. This enables the creation of virtual movies. All data are located within separate files, in a well-structured directory hierarchy. This makes it transparent, easy to search, modify and adapt to any kind of discrete element simulations. The data structure gives the opportunity to reduce the amount of RAM allocated to the visualization process. It is particularly convenient when large numbers of particles are presented. The issue of graphical templates was considered as well, because illumination and colours influence both the perception and interpretation of results.

It is worth noting that discrete element simulation of systems containing millions and even billions of particles is a relatively recent phenomenon. This paper is not offering a definitive solution to the presented issues; however, the authors hope that it can add some contribution to the debate of this important topic.

The open source 3D visualizer, which is called DEV_KM, is intended to be freely available in the Open Source format on the Internet.

References

[1] R. Kostek, *Simulation of the granular flow within an impact crusher*. Inżynieria i Aparatura Chemiczna 48(2), 74-75 (2009).
[2] J.P. Latham, A. Munjiza, X. Garcia, J. Xiang and R. Guises, *Three-dimensional particle shape acquisition and use of shape library for DEM and FEM/DEM simulation.* Minerals Engineering 21(11) (2008).
[3] T. Tsuji, K. Yabumoto and T. Tanaka, *Spontaneous structures in three-dimensional bubbling gas-fluidized bed by parallel DEM–CFD coupling simulation*. Powder Technology 184(2), 132-140 (2008).
[4] ParaView, http://www.paraview.org
[5] Wm. G. Hoover, *Computational Physics with Particles – Nonequilibrium Molecular Dynamics and Smooth Particle*

Applied Mechanic. Computational Methods in Science and Technology 13(2), 83-93 (2007).
[6] F.Y. Fraige, P.A. Langston and G.Z. Chen, *Distinct element modelling of cubic particle packing and flow*. Powder Technology 186(3) 224-240 (2008).
[7] W.R. Ketterhagen, J.S. Curtis, C.R.Wassgren and B.C. Hancock, *Modeling granular segregation in flow from quasi-three-dimensional, wedge-shaped hoppers*. Powder Technology 179(3), 126-143 (2008).
[8] The Scientific Computing and Imaging (SCI) Institute web site http://www.sci.utah.edu
[9] C.P. Gribble, C. Brownlee and S.G. Parker, *Practical global illumination for interactive particle visualization*. Computers & Graphics 32(1), 14-24 (2008).
[10] M.L. Sawley, J. Biddiscombe and J.M. Favre, *Advanced visualization of large datasets for discrete element method simulations*. Discrete Element Methods (DEM) '07, Brisbane, Australia, 26-29 August 2007.
[11] S. Weyna, *Microflown based identication of vortex shadding in the space of real acoustic flow fields*. The Twelfth International Congresson Sound and Vibration, CSV12, Lisbon, Portugal, 690, 2005.
[12] S. Weyna, *Image of acoustic energy field radiated in 3d space by electrodynamic loudspeaker*. 19th International Congress on Acoustics Madrid, 2007.
[13] M. Hlawitschka and G. Scheuermann, *HOT Lines: Tracking lines in higher order tensor fields*. 16-th IEEE Visualization (VIS 2005), 27-34 (2005).
[14] M. Schirski, T. Kuhlena, M. Hoppb, P. Adomeit, S. Pischinger and C. Bischof, *Virtual Tubelets – efficiently visualizing large amounts of particle trajectories*. Computers & Graphics, 29(1) 17-27 (2005).
[15] F. Goes, S. Goldensteina and L. Velhob, *A simple and flexible framework to adapt dynamic meshes*. Computers & Graphics 32(2) 141-148 (2008).
[16] Y. Xi and Y. Duan, *A novel region-growing based iso-surface extraction algorithm*. Computers & Graphics, doi: 10.1016/ j.cag. 2008.09.007.
[17] A. Wiebel, C. Garth and G. Scheuermann, *Computation of localized flow for steady and unsteady vector fields and its applications*. IEEE Trans. Visualization and Computer Graphics 1(8) (2002).
[18] P.W. Cleary and J. Ha, *Three-dimensional SPH simulation of light metal components*. J. Light Metals 2(3), 16-183 (1993).
[19] P.W. Cleary and J. Ha, *Modelling the high pressure die casting process using SPH.* Material Forum 25, 1-29 (2001).
[20] Visualization Toolkit (VTK), http://www.vtk.org
[21] Rheingans P., *Task-based color scale design.* Proceedings of Applied Image and Pattern Recognition '99, SPIE, 35-43 (1999).

APPENDIX

Structure of the VTK files

The VTK files consist of five basic parts

1. The first part is the file version and identifier. This is a single line, e.g.
   # vtk DataFile Version 2.0
2. The second part is the header; string – 256 characters maximum, terminated by end-of-line character\n. The header describes the file.
   3D Simulation of DEM granular flow2009/10/10 \n
3. The next part describes the file format, ASCII or BINARY.
   ASCII
4. The fourth part is the dataset structure. The geometry part describes the geometry and topology of the dataset. This part begins with a line containing the keyword **DATASET** followed by a keyword describing the type of dataset. Then, depending upon the type of dataset, other keyword/data combinations define the actual data.

DATASET STRUCTURED_POINTS
DIMENSIONS $n_x\ n_y\ n_z$
ORIGIN $x\ y\ z$
SPACING $s_x\ s_y\ s_z$

STRUCTURED_GRID
DIMENSIONS $n_x\ n_y\ n_z$
POINTS n *dataType*
$p_{0x}\ p_{0y}\ p_{0z}$
$p_{1x}\ p_{1y}\ p_{1z}$
...
$p_{(n-1)x}\ p_{(n-1)y}\ p_{(n-1)z}$

UNSTRUCTURED_GRID
POINTS n *dataType*
$p_{0x}\ p_{0y}\ p_{0z}$
$p_{1x}\ p_{1y}\ p_{1z}$
...
$p_{(n-1)x}\ p_{(n-1)y}\ p_{(n-1)z}$
CELLS n *size*
$numPoints_0$, $i, j, k, l,...$
$numPoints_1$, $i, j, k, l,...$
$numPoints_2$, $i, j, k, l,...$
...
$numPoints_{n-1}$, $i, j, k, l,...$

CELL_TYPES n
$type_0$
$type_1$
$type_2$

...
$type_{n-1}$

POLYDATA POINTS *n dataType*
$p_{0x}\ p_{0y}\ p_{0z}$
$p_{1x}\ p_{1y}\ p_{1z}$
...
$p_{(n-1)x}\ p_{(n-1)y}\ p_{(n-1)z}$

VERTICES *n size*
$numPoints_0, i_0, j_0, k_0, ...$
$numPoints_1, i_1, j_1, k_1, ...$
...
$numPoints_{n-1}, i_{n-1}, j_{n-1}, k_{n-1}, ...$

LINES *n size*
$numPoints_0, i_0, j_0, k_0, ...$
$numPoints_1, i_1, j_1, k_1, ...$
...
$numPoints_{n-1}, i_{n-1}, j_{n-1}, k_{n-1}, ...$

POLYGONS *n size*
$numPoints_0, i_0, j_0, k_0, ...$
$numPoints_1, i_1, j_1, k_1, ...$
...
$numPoints_{n-1}, i_{n-1}, j_{n-1}, k_{n-1}, ...$

TRIANGLE_STRIPS *n size*
$numPoints_0, i_0, j_0, k_0, ...$
$numPoints_1, i_1, j_1, k_1, ...$
...
$numPoints_{n-1}, i_{n-1}, j_{n-1}, k_{n-1}, ...$

RECTILINEAR_GRID
DIMENSIONS $n_x\ n_y\ n_z$
X_COORDINATES n_x *dataType*
$x_0\ x_1 ...\ x_{(nx-1)}$
Y_COORDINATES n_y *dataType*
$y_0\ y1 ...\ y_{(ny-1)}$
Z_COORDINATES n_z *dataType*
$z_0\ z_1 ...\ z_{(nz-1)}$

FIELD *dataName numArrays*

arrayName0 numComponents numTuples dataType
$f_{00}\ f_{01} \cdots f_{0(numComponents-1)}$
$f_{10}\ f_{11} \cdots f_{1(numComponents-1)}$
...
$f_{(numTuples-1)0}\ f_{(numTuples-1)1} \cdots f_{(numTuples-1)(numComponents-1)}$

arrayName(numArrays-1) numComponents numTuples dataType
$f_{00}\ f_{01} \cdots f_{0(numComponents-1)}$
$f_{10}\ f_{11} \cdots f_{1(numComponents-1)}$
...
$f_{(numTuples-1)0}\ f_{(numTuples-1)1} \cdots f_{(numTuples-1)(numComponents-1)}$

5. The final part describes the dataset attributes. This part begins with the keywords POINT_DATA or CELL_DATA, followed by an integer number specifying the number of points or cells, respectively. (It does not matter whether POINT_DATA or CELL_DATA comes first.) Other keyword/data combinations then define the actual dataset attribute values (i.e., scalars, vectors, tensors, normals, texture coordinates, or field data) [20].

The example presents a cube represented by six polygonal faces. Scalar and vectror fields have been associated with the polygons and nodes.

```
# vtk DataFile Version 2.0
Cube example
ASCII
DATASET POLYDATA

POINTS 8 float
0.0 0.0 0.0
2.0 0.0 0.0
2.0 2.0 0.0
0.0 2.0 0.0
0.0 0.0 2.0
2.0 0.0 2.0
2.0 2.0 2.0
0.0 2.0 2.0

POLYGONS 6 30
4  0 1 2 3
4  4 5 6 7
4  0 1 5 4
4  2 3 7 6
4  0 4 7 3
4  1 2 6 5

POINT_DATA 8
SCALARS density float 1
LOOKUP_TABLE default
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0

FIELD FieldData 1
Wektor-1-x-y-z 3 8 float
1 2 3
2 3 4
3 4 5
4 5 6
5 6 7
6 7 8
7 8 9
8 9 10

CELL_DATA 6
SCALARS Temperatura-prostokatow int 1
LOOKUP_TABLE default
```

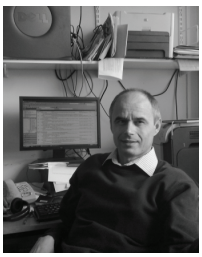
```
0
2
4
6
8
10

NORMALS normalne-prostokatow float
 0  0 -1
 0  0  1
 0 -1  0
 0  1  0
-1  0  0
 1  0  0

FIELD FieldData 3
Skalar-1-x 1 6 int
1
2
3
4
5
6
Wektor-2-x-y 2 6 float
1.0 2.0
2.0 3.0
3.0 4.0
4.0 5.0
5.0 6.0
6.0 7.0
Wektor-3-x-y-z 3 6 float
1.0 2.0 3.0
2.0 3.0 4.0
3.0 4.0 5.0
4.0 5.0 6.0
5.0 6.0 7.0
6.0 7.0 8.0
```

For more details please visit Web page [21].

ROBERT KOSTEK received the M.Sc. degree from the University of Technology and Agriculture in Bydgoszcz (1998) and the Ph.D. degree from the Szczecin University of Technology (2005). From 2006 to 2007 he worked at Queen Mary, University of London in the field of visualization. Currently he is employed at the University of Technology and Life Sciences in Bydgoszcz. His present research activities are focused on modelling, simulation, identification, numerical methods and non-linear dynamics.



PROFESSOR MUNJIZA has pioneered the combined finite discrete element method. He is one of the founding members of Computational Mechanics of Discontinua. He has written over 150 papers including 2 books, of which one is a monograph of his own research results. He is credited with the invention of NBS contact search algorithm and also MR search solutions. He has developed original fracture solutions for both 2D and 3D dynamic simulations. His recent work has centered on nano-scale applications, including molecular dynamics and the next generation of fracture solutions. He is also an accomplished software designer with both commercial codes and open source codes attached to his name. It is worth mentioning that he started his career as structural engineer and bridge designer with 14 highway bridges attached to his name. He has been a part to large scale discontinua simulations involving millions and even billions of particles.