 CMES, vol.11, no.3, pp.111-120, 2006

ADVENTURE AutoGL: A Handy Graphics and GUI Library for Researchers and Developers of Numerical Simulations

H. Kawai [1]

Abstract: ADVENTURE AutoGL (pronounced as 'O-te-ga-lu') is a graphics and GUI library, dedicated for simulation-based research and development. It is designed for the simulation users to develop their own data viewers and editors. Currently, the library is used among many researchers and simulation users, mainly in universities and national research centers. Its functionalities and supported platforms are explained. AutoGL applications of various kinds of simulation methods are demonstrated also.

keyword: Numerical simulation, GUI, Graphics, Visualization, Pre- and post- processing.

1 Introduction

A variety of numerical simulation methods, including FDM, FVM, FEM, BEM, mesh-less and particle-based method, as well as MD (Molecular Dynamics), cellular automaton, multi- agent and so on, have been proposed and used among researchers and engineers. Each of them uses its own data structure and requires some special data manipulation. And they have to be visualized, created and modified in a flexible way.

ADVENTURE AutoGL is a simple and handy graphics and GUI (Graphical User Interface) library. Using this library, a researcher or a developer using numerical simulation methods can develop his or her own viewers and editors, namely pre- and post-processors, to visualize and manipulate simulation data.

2 Survey

My motivation to develop ADVENTURE AutoGL is summarized here as its own requirement specification.

* Ease of Use

It has to be designed so that most of researchers in our field can use it easily. I assume that they can write a program either in Fortran, C, C++ or Java to develop their simulation code. However, most of them are not professional programmers, so they are satisfied if they can just use basic computer language constructs such as branch, loop, array and character string. Neither modular programming nor object-oriented programming should be required. Traditionally, GUI and graphics has been regarded as 'dangerous zone' for them. Therefore, special care must be taken to design AutoGL so that its API (Application Programming Interface) is acceptable for them.

* Portability

It has to run on anywhere. This means supporting, not only popular platforms such as Windows, MacOS, Linux and UNIX, but also any special or advanced platforms that some of AutoGL users are interested in for their research purposes, such as virtual reality environment, supercomputer and computational grid, Web, PDA (Portable Digital Assistant) and smart phone. Extensibility is the key for such types of research activities. Also, an extension work should also be simple and easy for them.

* Efficiency

It has to be efficient. Visualization of 3-D objects costs huge amount of computational resources, therefore the implementation must be efficient both in terms of space and time. Actually, some of AutoGL users who want to deal with a finite element analysis of 100M DOFs (Degrees of Freedom) scale, such as ADVENTURE community, as well as users of GeoFEM in RIST, Japan and Salinas in Sandia National Laboratory, U.S. require visualization and manipulation of such a huge data set that no commercial package can support. MD and particle-based simulation users also request handling of huge number of molecules or particles.

Here, the richness of functionality is intentionally excluded because of the diversity of needs required for advanced research activities in the numerical simulation field. Rather than any type of package software specialized for a certain application domain, a kind of development environment such as programming languages, li-

[1] Keio University, Yokohama, Japan

braries and utility commands is suitable to meet the requirement here.

Before developing AutoGL, I surveyed and also used many existing solutions for visualization and manipulation of scientific data, either as commercial products or freely available packages. They are classified roughly into the following categories.

* Libraries for Programming Language

Variety of low-level GUI / Graphics libraries is available. As GUI libraries, there are Motif on UNIX, Gtk+ and Qt on Linux, Cocoa and Carbon on MacOS, WIN32/64, MFC (Microsoft Foundation Class library) and .NET framework on Windows, and AWT (Abstract Windowing Toolkit), Swing and SWT on Java environment. As graphics libraries, OpenGL, Direct3D and Java3D are popular. In terms of efficiency, most of them are sufficient. However, most of the libraries are platform-dependent. Moreover, they are also very difficult to use for most of the researchers assumed here, especially if the performance is the main issue.

VTK (Visualization ToolKit) is a high-level library for data visualization and available through C++, Java, and many other scripting language. It is relatively easy to use, but it is not efficient enough to support a huge-scale analysis.

* Visualization Package

Visualization packages such as AVS/Express and MicroAVS, PV-Wave, OpenDX (IBM Visualization Data Explorer), are popular among the numerical simulation community. Their input files are easy to write, and most of them support some ways to customize file import and visualization. The lack of performance and efficient memory usage against a huge data set prohibit some of the researchers to use them.

Data file format useful for data visualization such as VRML (Virtual Reality Modeling Language) and X3D, and their viewers / editors may also be classified in this category.

* CAE Package

Most of CAD / CAE packages, such as CATIA, I-DEAS, MSC/Patran, FEMAP, ABAQUS, STAR-CD and FIELDVIEW, are for FEM or FVM. Their target users are mainly mechanical engineers who do not write a program, rather than researchers and developers. Their file formats, typically called a neutral file, are often complicated and difficult to read / write. Currently no package can support a huge scale analysis.

Very few packages are available for other simulation methods such as BEM and MD, and they are often expensive. There is virtually no support for more advanced simulation methods, for example, mesh-less methods such as EFGM (Element-Free Galerkin Method) and MLPG (Mesh-less Local Petrov-Galerkin Method) by Atluri et. al (2004a, 2004b), particle-based methods, cellular automaton and multi-agent approaches. The researchers of these methods have to develop their own pre- and post-processors.

3 ADVENTURE AutoGL

ADVENTURE project is an open-source research and development project, starting since 1997, to develop finite element-based large-scale CAE applications. The system called ADVENTURE now consists of about 20 modules. ADVENTURE AutoGL library is contained in ADVENTURE Auto module in ADVENTURE system.

To obtain ADV/Auto module and AutoGL library, a user can download its source code from ADVENTURE home page. The newer versions are also available from the author's home page linked from the member page of ADVENURE.

3.1 Functionality

Currently, AutoGL supports the following visualization capabilities.

- Drawing lines, triangles, quadrilaterals, character strings, spheres, cones, cylinders, etc. Hidden surface removal, lighting, shading and transparency are supported.
- Visualization techniques such as iso-line contour, smooth contour, arrow diagram, deformation plot, iso-surface and arbitrary oriented cut plane.
- GUI components such as buttons, text fields, toggle and choice buttons.
- Event handling, picking and selection of 2-D and 3-D objects.

Figure 1 shows a typical AutoGL application. It has a view window and also one or more panel windows. The view window contains a 2-D or 3-D graphics image, and

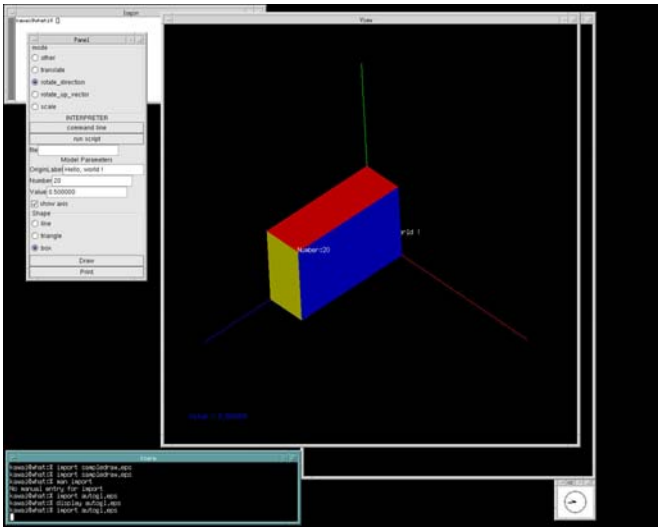


Figure 1 : An example AutoGL application, 'Panel'

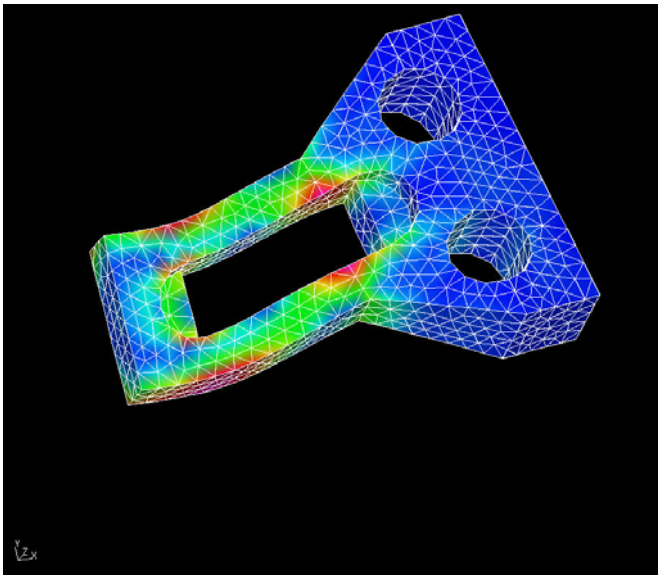


Figure 2 : Contour plot

each panel windows contains GUI components such as buttons and text fields.

Figure 2 and 3 shows some examples of visualization. Figure 2 is a stress contour plot of structural analysis using FEM, and Figure 3 is an iso-surface plot of a scalar field of an FDM result.

Here is an example source code section in C language, written by an AutoGL application programmer.

```
void RedrawView ()
{
  AutoGL_SetColor (1.0,1.0,0.0);
    /* RGB color components */
  AutoGL_DrawLine
    (0.0,0.0,0.0, 3.0,2.0,1.0);
    /* draw line between two 3-D points */
  AutoGL_DrawTriangle
    (0.0,0.0,0.0,
    1.0,0.0,0.0,
    0.0,2.0,0.0);
    /* draw triangle of three points */
}
```

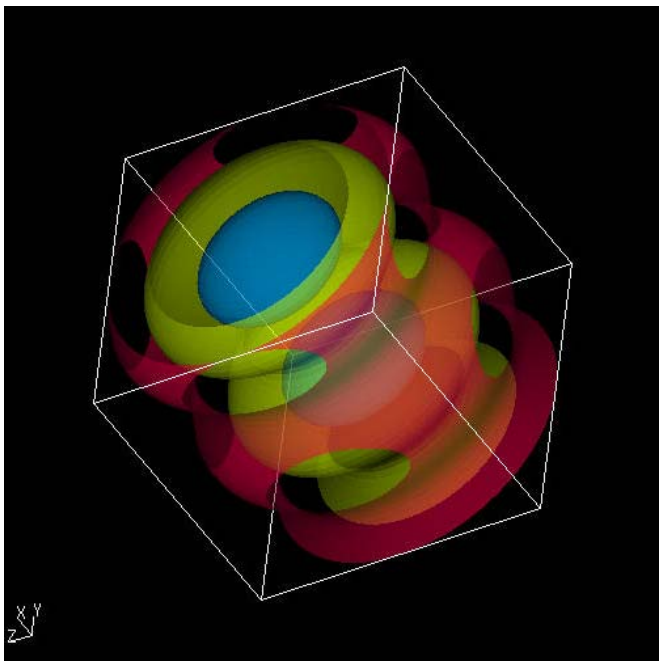


Figure 3 : Iso-surface plot

The current color is specified as yellow (red = 1, green = 1, blue = 0), and a line segment and a triangle are rendered into the view window. They are defined in 3-D world coordinate system. The triangle is lighted and shaded automatically. It is like a classic BASIC language, although it can also handle 3-D graphics.

```
double Parameter;
void Print (void)
{
  printf (" Hello World \n");
  printf (" Parameter is %f \n", Parameter);
```

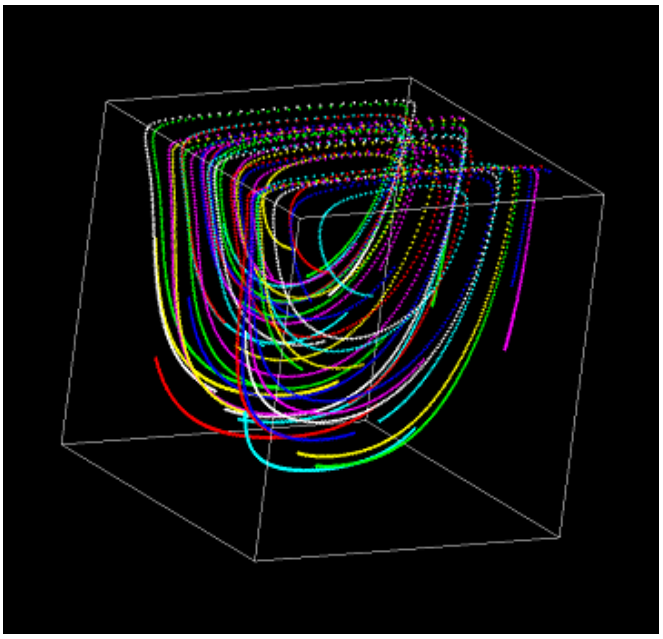


Figure 4 : Particle trace of cavity flow

```
}
void AutoGL_SetUp (int argc, char *argv[])
{
  AutoGL_SetViewRedrawCallback
    (RedrawView);
  AutoGL_AddReal(&Parameter, "Parameter");
  AutoGL_AddCallback (Print, "Print");
}
```

In an AutoGL application, the programmer cannot define the main function, which is already defined inside the AutoGL library. Instead, the programmer has to define AutoGL_SetUp function, which is called once at the initialization phase. Fortran77 API is also defined.

In this code section, there is a global variable, Parameter, of double precision real number type, and a function, Print, defined by the programmer. When the AutoGL initialization function, AutoGL_SetUp, is called, the variable, Parameter, and the function, Print, are registered into AutoGL. The variable, Parameter, appears in a panel window as a text field control, which accepts a real number. The function, Print, is represented as a button on the panel window. When an AutoGL application user modifies the value of the text field control labeled "Parameter", it is reflected into the variable, Parameter, immediately. Then the AutoGL application user pushes the button labeled "Print", and the function, Print, is invoked. Two message lines containing "Hello World" and the current value of the variable, Parameter, is shown on the terminal.

AutoGL_SetViewRedrawCallback registers a view redraw callback function. In this case, it is the function RedrawView, already provided by the programmer. The function is called every time the view is redrawn or a mouse or a key event happens.

Combining AutoGL low-level visualization functionalities, users can implement more advanced visualization techniques by themselves. They can also support their own special numerical simulation scheme and its original data structure.

For example, capability to generate particle traces, shown in Figure 4, is calculated by a fluid dynamics code, ADV/Fluid module, developed in ADVENTURE project. Also, in the research by Okada et. al (2004), two finite element analysis results of different scales are superposed to form a final contour plot required for mesh superposition method, shown in the Figure 5.

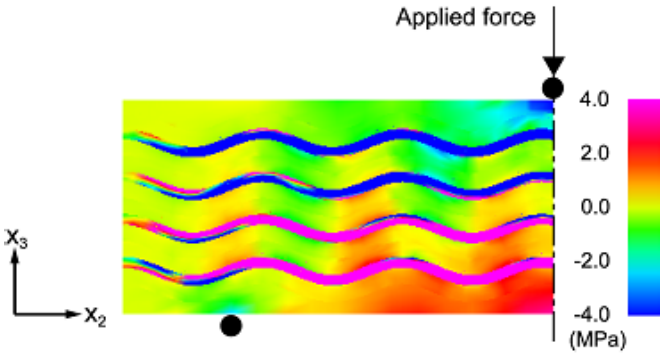
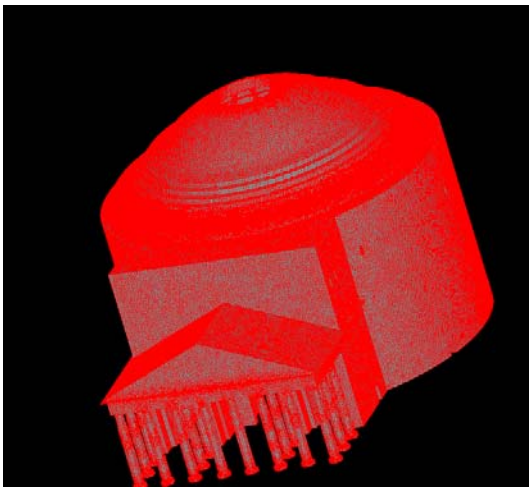


Figure 5 : Mesh superposition method

3.2 Platform

AutoGL supports the following computer platforms.

- Client environments using OpenGL library: Linux, FreeBSD, MacOS (using Gtk+ and GtkGLArea libraries), Windows (using WIN32 library) and some commercial UNIX environments (using Motif library).
- Server environments using off-line rendering: PC clusters, SGI Altix, Hitachi SR8000 and The Earth Simulator.



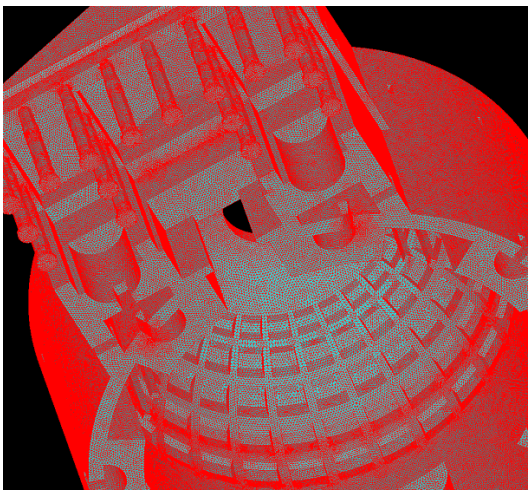


Figure 6 : Mesh of Pantheon (45M DOFs)

- Web-application: As browsers, Internet Explorer, Netscape Navigator and Mozilla. Also, PDA terminals such as NTT DoCoMo i-mode portable phone.

On a typical desktop or notebook PC equipped with a commodity-level video card, interactive operations to a finite element model of 100M DOFs are supported. Figure 6 shows visualization of the mesh of an ancient architecture, Pantheon in Roma, with 45M DOFs.

Without modifying its source code, an AutoGL application is available through a Web browser as a Web-application. Image data are generated from the Web server side and downloaded to the browser side. Figure 7 shows a sample application 'Panel' already demonstrated in Figure 1, this time running on Mozilla.

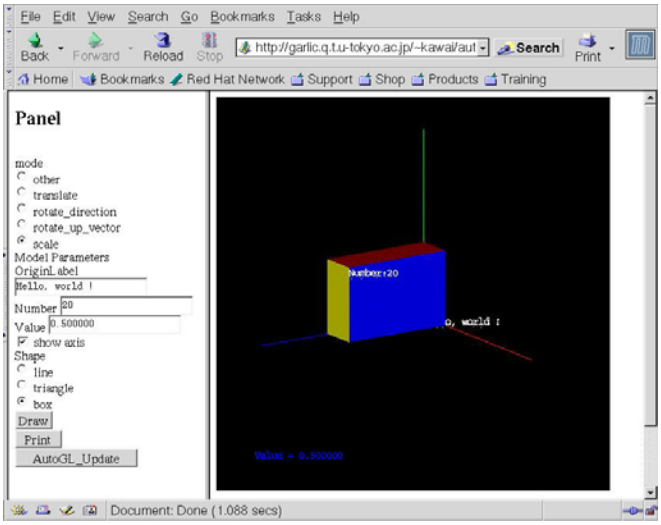


Figure 7 : Web-application 'Panel' browsing on Mozilla

It is implemented as a CGI (Common Gateway Interface)-based Web application. An AutoGL application binary executable, created by a programmer and written in C, C++ or Fortran, runs on the server side as a "daemon" process, which is independent of the Web server process. The AutoGL application process communicates with a CGI script.

First, the programmer runs the AutoGL application binary executable on the Web server machine. Next, a user of the AutoGL-based Web application on the client side terminal, which may be HTML, Java applet or i-mode Java application, operates through the terminal, and invokes CGI calls. Every time a CGI call happens, the CGI script program invoked from the Web server process sends a message to the AutoGL application process, already running on the Web server machine as a "daemon" process, mainly through file I/O. The CGI script program writes an AutoGL command script file. The AutoGL application finds the script file, reads it, generates a view image through its off-line rendering capability and writes a JPEG file. Finally, the image file is sent back to the client terminal by the Web server process.

Not only on the Web server, off-line rendering capability is also useful if simulation data become very huge. Visualization images can be generated on a computational server. Rendering is performed based on software emulation. Figure 8 shows the animation of a dynamic analysis of an ABWR (Advanced Boiled Water nuclear Reactor). A finite element simulation is carried out on The Earth

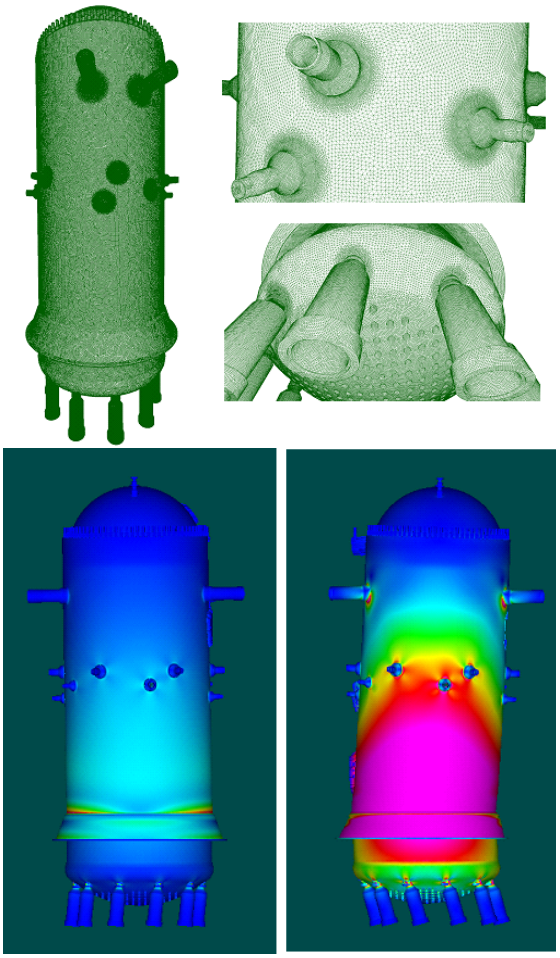


Figure 8 : An off-line rendering of ABWR (35M DOFs) dynamic analysis

Simulator using ADV/Solid module. It is performed by Ogino and Shioya et. al (2005).

4 AutoGL Applications

Since 2002, ADV/Auto module and AutoGL are downloaded from ADVENTURE home page and used among various kinds of researchers and engineers. The followings are some of their applications.

Until now, AutoGL has covered the following application domains.

- Mechanical engineering
  - Turbine blades and pumps
  - Nuclear power plant vessels
  - Portable information devices
  - Sports dynamics
  - Fluid-structure interaction
  - Virtual manufacturing and prototyping
- Architectural, civil and environmental engineering
  - Ancient and legacy architecture
  - City traffics
  - Pollution of ocean and lake
  - Heat island problem
  - Earthquake and health monitoring
- Nano-level material science
  - Neutron irradiation damage
- Biology
  - Biomechanics
  - bio-informatics and DNA search

These applications are re-classified according to numerical methods, physics and disciplines as follows.

- Finite element method
  - Solid mechanics and fracture mechanics
  - Fluid dynamics
  - Electro-magnetics
  - Design optimization
  - Multi-scale analysis and homogenization
- Finite difference method
  - Fluid dynamics, ocean and climate modeling
- Finite volume method
  - Fluid dynamics, architectural environmental study
- Mesh-less and voxel-based method
- Molecular dynamics
- Dislocation dynamics
- Bio-informatics
- Multi-agent approach

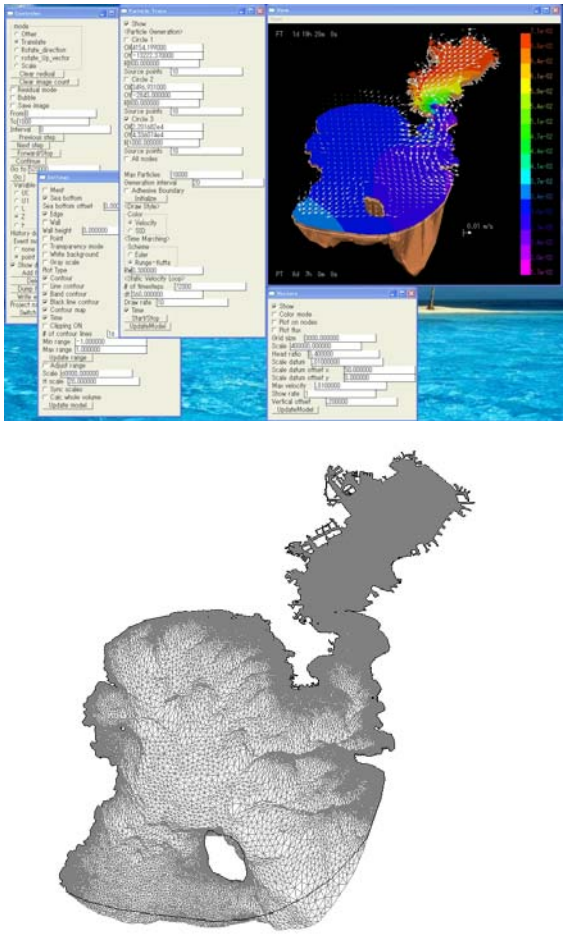


Figure 9 : A shallow water analysis of Tokyo bay for environmental assessment

– Traffics simulation

Figure 9 shows a screen image of the editors and the viewers specialized for an environmental simulation of lakes and coastal regions by Bunya and Yoshimura, et. al (2005). Using this graphical editor, coastal line data from GIS (Geographical Information System) database can be imported, edited and simplified for the analysis purpose. A finite element analysis is performed to solve the shallow water equation of the problem. Animation of the analysis results, such as contour and height plot, as well as particle plot can be generated.

Figure 10 shows traffic simulator based on multi-agent techniques, called MATES (Multi-Agent based Traffic and Environment Simulator) by Yoshimura (2005). It is written in C++. The simulation is performed to assess extension of LRT (Light Rail Transit) lines in Okayama city, Japan.

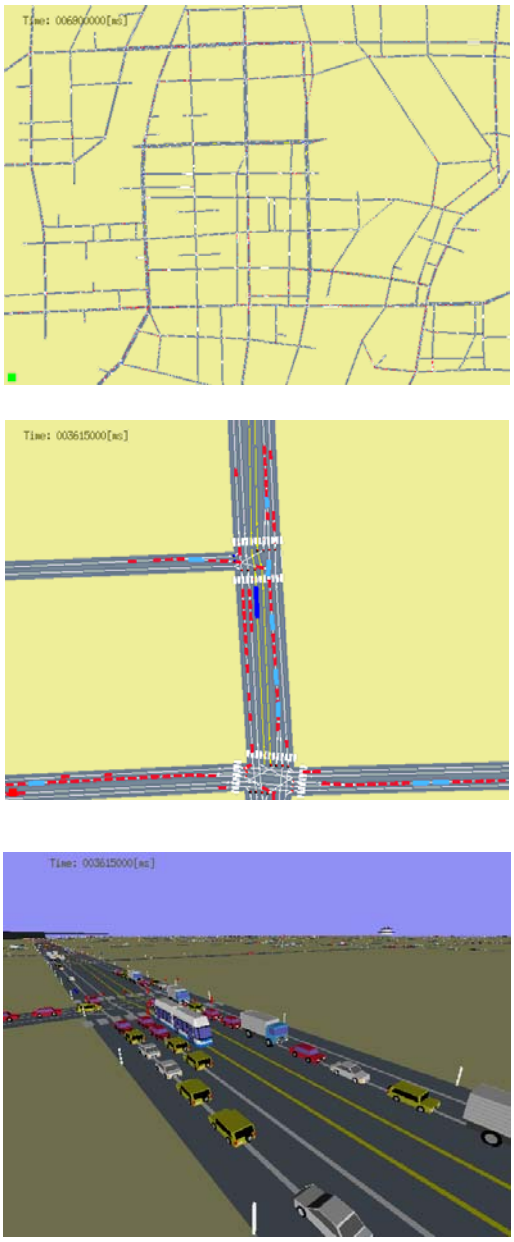


Figure 10 : Traffic simulation of Okayama

Figure 11 shows the results of an FDM-based CFD analysis of the heat island problem of Tokyo in Japan, developed in our 21st Century COE Program (Murakami laboratory). The upper left side picture shows a screen shot of the viewer specialized for the problem and simulation codes. It is written in Fortran77. Using the viewer, GIS data of land usage and building information can be superposed into CFD analysis results to show their co-

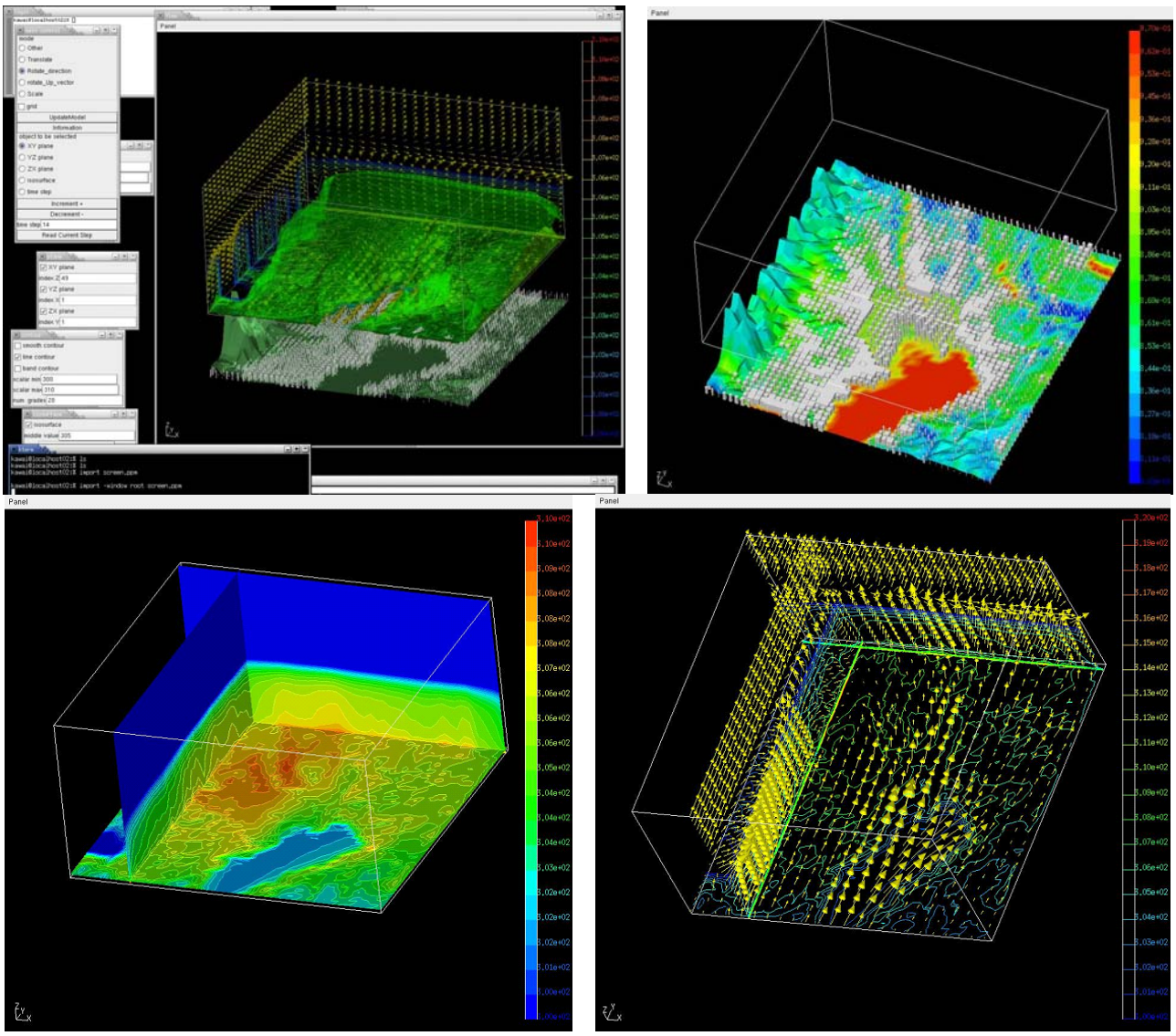


Figure 11 : A meteorological analysis over Tokyo for heat island problems

relation. The 2nd picture shows land and building information. The 3rd picture shows temperature distribution. The 4th picture shows velocity distribution of airflow.

Figure 12 shows an example of a 2-D finite element post-processor developed in our 21st Century COE Program (Noguchi laboratory), as a post-processing component of our finite element development and benchmarking framework. It is written in Fortran90. It took just a few days to develop. The framework also contains mesh generation components, benchmark verification components and interface to ABAQUS.

The examples, shown in Figure 13 and 14, are performed by Takahashi, et. al (2006). These simulation results are based on MD and dislocation dynamics, respectively. They are tools to enable a multi-scale simulation of material embrittlement by neutron irradiation damage.

5 Summary

ADVENTURE AutoGL library is introduced. Its capabilities and supported platforms are explained, followed by its applications to a variety of numerical simulations.

Acknowledgement: This work was financially supported, initially by The JSPS "Research for the Future" program of "Computational Science and Engineering", then followed by The 21st Century COE Program, 'Sys-

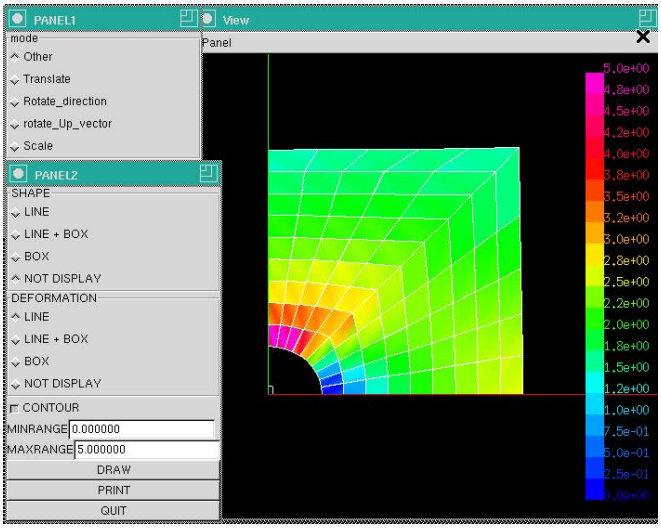


Figure 12 : 2-D post-processor in a finite elements benchmark framework

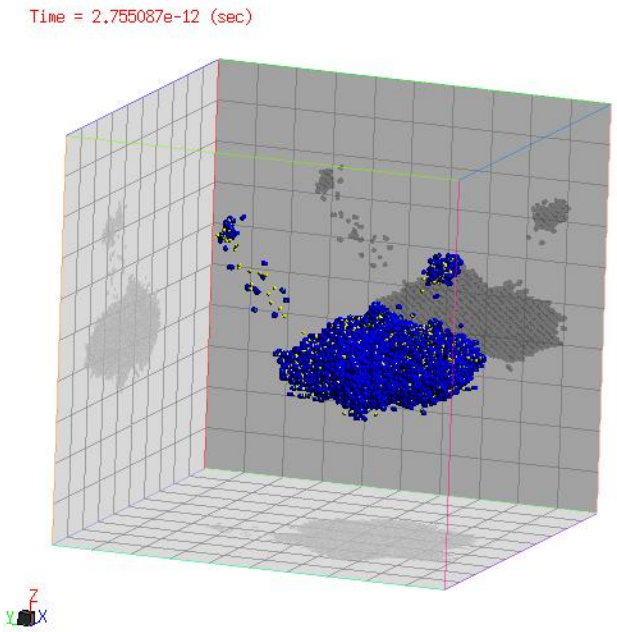


Figure 13 : Molecular dynamics

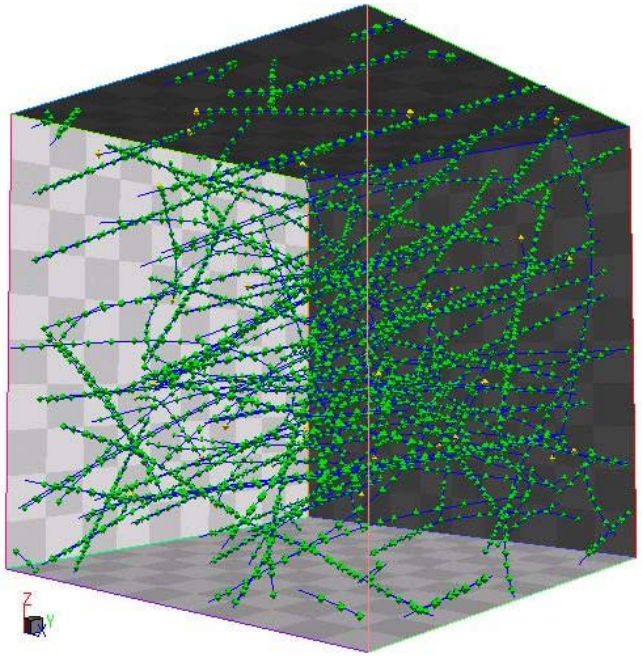


Figure 14 : Dislocation dynamics

tem Design: Paradigm Shift from Intelligence to Life', of the Ministry of Education, Science and Culture of Japan. Also, special thanks to 'hard-core' AutoGL users, including Prof. Murakami, Prof. Noguchi and Mr. Tanaka in Keio Univ., Prof. Kikuchi, Prof. Wada and Dr. Takahashi in Tokyo Univ. of Science, Prof. Kanayama, Prof. Shioya and Dr. Ogino in Kyushu Univ., Prof. Yoshimura, Prof. Soneda, Dr. Yamada, Dr. Bunya, Mr. Fujii and Mr. Nakama in Univ. of Tokyo, Prof. Okada in Kagoshima Univ., Prof. Miyamura in Nippon Univ., Prof. Mimura in Tsuruoka Univ., Prof. Nakabayashi in Toyo Univ., Prof. Ishihara in Kyushu Institute of Technology, and their laboratory members, as well as ADVENTURE community members.

References

Bunya, S.; Westerink, J. J. and Yoshimura, S. (2005): Discontinuous Boundary Implementation for the Shallow Water Equations. *Int. J. Numer. Meth. Fluids*, Vol. 47, pp. 1451-1468.

Han, Z. D. and Atluri, S. N. (2004): Meshless Local Petrov-Galerkin (MLPG) approaches for solving 3D Problems in elasto-statics. *CMES: Computer Modeling in Engineering and Sciences*, Vol. 6, pp. 169-188.

Okada, H.; Liu, C. T.; Ninomiya, T.; Fukui, Y. and Kumazawa, N. (2004): Analysis of Particulate Composite Materials Using an Element Overlay Technique. *CMES: Computer Modeling in Engineering and Sciences*, Vol. 6, pp. 333-348.

Ogino, M.; Shioya, R.; Kawai, H. and Yoshimura, S. (2005): Seismic Response Analysis of Nuclear Pressure Vessel Model with ADVENTRUE System on the Earth Simulator. *Journal of The Earth Simulator*, Vol.2, pp.41-

54.

Sladek, J. ; Sladek, V. and Atluri, S. N. (2004): Meshless Local Petrov-Galerkin Method in Anisotropic Elasticity. *CMES: Computer Modeling in Engineering and Sciences*, Vol. 6, pp. 477-490.

Takahashi, A.; Soneda, N.; Kikuchi, M. (2006): Computer Simulation of Microstructure Evolution of Fe-Cu Alloy during Thermal Ageing. *Key Engineering Materials*, Vol. 306-308, pp.917-922.

Yoshimura, S. (2005): MATES: Multi-Agent based Traffic and Environment Simulator – Theory, Implementation and Practical Application. *CMES: Computer Modeling in Engineering and Sciences*, Vol.11, No.1, pp.17-26.

ADVENTURE home page,

http://adventure.q.t.u-tokyo.ac.jp/

GeoFEM home page, http://geofem.tokyo.rist.or.jp/

VTK home page, http://www.vtk.org/

OpenDX home page, http://www.opendx.org/