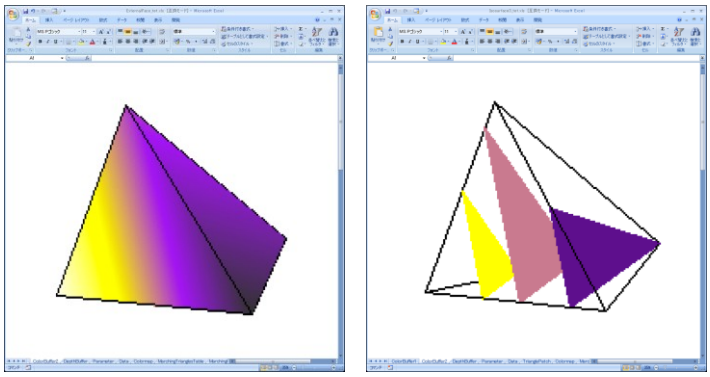
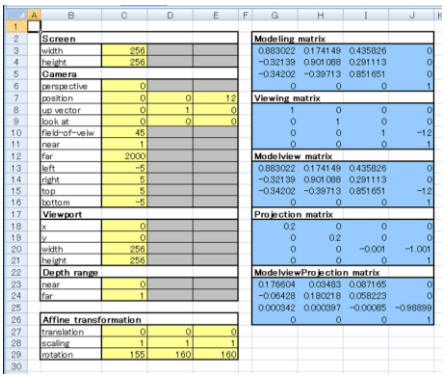
An Educational System of Scientific Visualization Techniques Using Microsoft Excel Spreadsheets

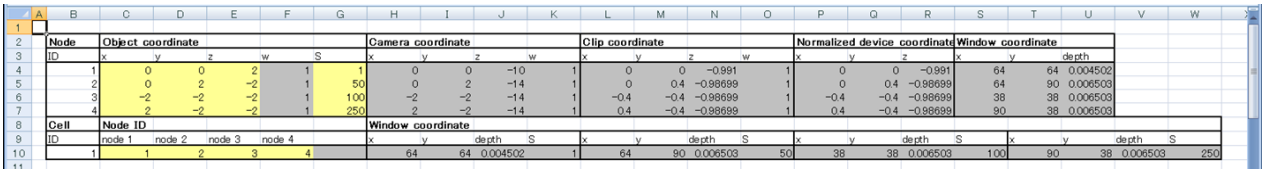
Naohisa Sakamoto and Koji Koyamada
Center for the Promotion of Excellence in Higher Education, Kyoto University



(a) Rendering results on worksheet.
External faces (left) and isosurfaces (right)



(b) Input parameters and transformation matrices.



(c) Vertex transformation.

Figure 1: Excel–based visualization system.

1. Introduction

Microsoft Excel is a popular and widely used spreadsheet application used in various fields. It provides not only statistical process functions but also many numerical functions. Moreover, because original user functions can be implemented by using the VBA (Visual Basic for Applications) program, Excel can be used to learn numerical simulation techniques.

In this paper, we present a new educational system of scientific visualization techniques that uses Microsoft Excel spreadsheets. In our system, the worksheet is used for the rendering window, for calculating the transformation matrices and for vertex transformation (see Figure 1).

The main characteristics of our system are as follows:

- Users can intuitively understand the table-driven algorithm that uses look-up tables, such as in isosurface extraction, by using the worksheet.
- Because the rendering processes can be visually shown on the worksheet in our system, users can learn the rendering pipeline, which is implemented as a fixed processing stage sequence on a GPU (Graphics Processing Unit), and is hard to learn using only graphics APIs (Application User Interfaces) such as OpenGL.
- By using our Excel-based system, users can efficiently learn the basis of computer graphics and scientific visualization techniques by using commodity PCs.

2. Graphics Pipeline

The graphics pipeline in our system is described in the following sections.

2.1 Frame Buffer

A frame buffer can be easily created on a worksheet by treating a cell as a pixel. Then, it is possible to plot a point on the frame buffer by filling in the cell on the worksheet (see Figure 2).

Figure 2: Frame buffer composed of square cells.

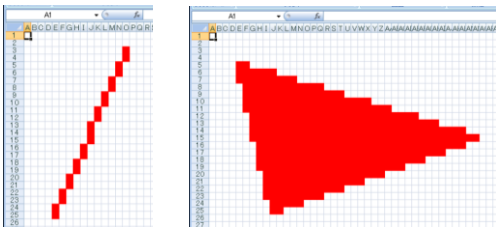
2.2 Vertex Processing

Vertex transformations needed for rendering can be realized on the worksheet. Transformation matrices for the vertex transformation, such as the modelview matrix and projection matrix in OpenGL, can be easily calculated on the worksheet by using only standard worksheet functions, without using VBA (see Figures 1[b] and [c]).

2.4 Fragment Processing

In our system, by filling in the cells, which can be regarded as a fragment in OpenGL, on the frame buffer worksheet using Bresenham's algorithm, it is possible to draw a line on the worksheet (see Figure 3[a]). Polygons can also be drawn on the

worksheet by using a scan conversion algorithm (see Figure 3[b]). In our system, several tests are applied that determine whether the fragment is displayed or not. These processes are performed as VBA functions.



(a) Line (b) Triangle polygon

Figure 3: Line and polygon drawing on the worksheet.

3. Volume Visualization System

Our Excel-based system provides several basic techniques for volume visualization. In this section, we describe the system architecture and the volume visualization functions implemented in our system.

3.1 System Architecture

The system architecture is shown in Figure 4. First, input parameters are specified and used for calculating the transformation matrices on the parameter worksheet (see Figure 1[b]). Then, the vertices of the volume data in the object coordinate system are transformed to the window coordinate system by using the transformation matrices on the vertex processing worksheet (see Figure 1[c]). Finally, the fragment process is executed for the primitive drawing composed of the transformed vertices in VBA, and the rendering result is displayed on the frame buffer worksheet.

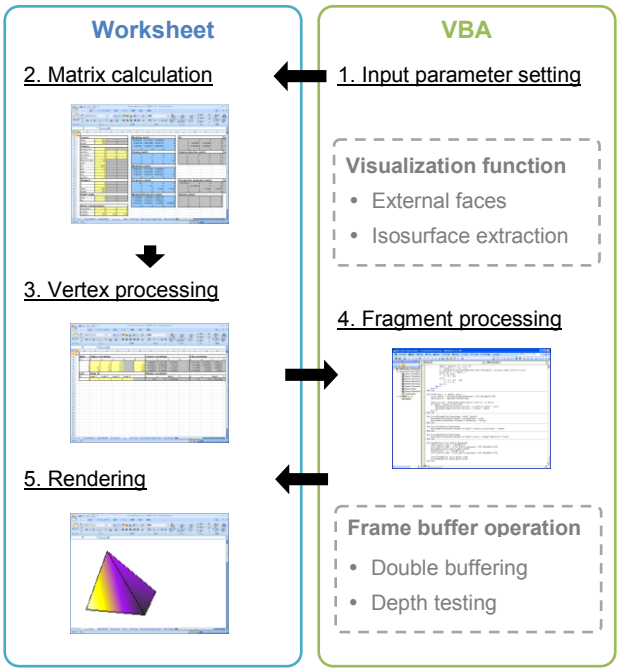


Figure 4: Architecture of our visualization system.

3.2 Visualization functions

Two typical volume visualization functions that are implemented in our system are described next.

External Faces: The external faces of a tetrahedral element can be rendered on the worksheet in 3D by using our system. The rendering results are shown in Figure 1(a) on the left. In our system, a double-buffering process is performed for the animation rendering by switching two color buffer worksheets [1] (see Figure 5 and attached movie file).

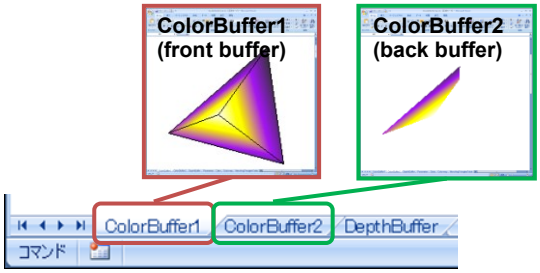


Figure 5: Double buffering using two worksheets.

Isosurface Extraction: Isosurfaces for a tetrahedral element can be extracted by referring to the lookup table on the worksheet. The results are displayed as triangular polygons on the frame buffer worksheet. Figure 1(a) on the right shows the three isosurfaces that are extracted by changing the isolevel. In our system, hidden surface removal can be performed with a Z buffer algorithm implemented in VBA during the rendering process by using the depth buffer worksheet (see Figure 6).

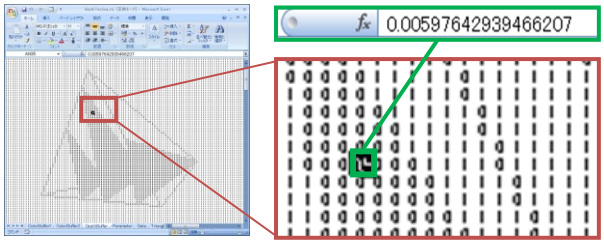


Figure 6: Depth values are stored in each cell on the depth buffer worksheet.

4. Visualization Class with Excel

We provide a class with Excel to teach the basic techniques of scientific visualization (see Figure 7). Students can learn the basic principles of computer graphics and visualization techniques by using our system.

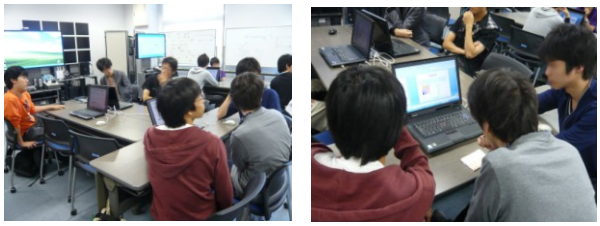


Figure 7: Visualization class using Excel.

5. Summary

We have developed an educational system of scientific visualization techniques that uses Microsoft Excel spreadsheets. In the near future, we expect to implement volume-rendering functions in our system.

References

[1] P. Rakos, Microsoft Excel: Revolutionary 3D Game Engine. http://www.gamasutra.com/view/feature/3563/microsoft_excel_revolutionary_3d_.php