DEVELOPMENT FOR WEB-BASED CG SYSTEM AND ITS APPLICATION TO MODELING AND ANIMATION SYSTEMS

Yoshinori MOCHIZUKI and Tomoyuki NISHITA
The University of Tokyo
Tokyo, JAPAN

ABSTRACT

Network infrastructure has become an important topic in recent years, and there are now many different services using the World Wide Web. Accesses to Web contents from mobile terminals such as notebook PC's, cellular phones (i-mode, iαppli (NTT DoCoMo, 2002)) have increased. This paper proposes a Web-based CG system. Users specify parameters for the Web client using HTML FORM or Java Applets. The Web server receives these parameters, and executes a rendering process, which requires a high computational cost. We also apply this system to modeling and animation systems.

keywords: computer graphics, Java Applets, World Wide Web

1 INTRODUCTION

Internet-related services, especially the World Wide Web, have become widely used in recent years. The services have a broad range of applications, which include business, various types of entertainment, distance learning, and so on. These services are supported by several different representations and Web technologies such as HTML, Java Applet, DHTML, VRML, XML, and many types of dynamic contents have been used. Computer graphics have become extensively used as a representation technique for Web contents, so the role of computer graphics has become very important.

Meanwhile, the client machines from which users access Web contents have become diversified. Mobile terminals such as notebook computers, cellular phones (i-mode, iαppli) have also proliferated in recent years, but the computational power of these mobile terminals is generally low, so it is very difficult to execute CG applications, which require high computational power.

We propose a new CG system in which the procedure for creating images is shared between the client machine and the Web server. This system creates images and animations using the following steps: Step 1) Users input the necessary information to create the image using Java Applets and HTML FORMs from the Web browser on the client machine. That information is sent to the Web server using CGI (Common Gateway Interface). Step 2) The Web server, which is made up of the information analysis parser and the rendering application, receives that information from the client machine via the information analysis parser. Step 3) The information analysis parser converts the information into data that the rendering application can use, and saves that data in the parameter file. Step 4) The rendering application loads the parameter file which is generated by the information analysis parser, and executes the rendering. Step 5) The rendering results are saved as output files, such as JPEG and MPEG files. An HTML file that includes the rendering results is generated, is sent back to the client machine, and displayed by the Web browser.

There are several advantages to this system: (1) Users can get high-quality rendering results even though the client does not have extensive computer power, for instance, a notebook computer. (2) Users only need the Web browser on the client machine, so it is not necessary to install a special application, and this system is therefore platform-independent. (3) Because this system uses the Web, users can utilize this system wherever there is an internet-capable environment.

2 SYSTEM OUTLINE

2.1 System Constitution

The constituents of this system are shown in Figure 1. The information needed for rendering is generated from HTML FORM and Java Applets on the Web client, and it is notified to the Web server through the CGI. The information analysis parser and rendering applications are in the Web server, and the information analysis parser

receives the information from the Web client. The analysis parser processes the information so that the rendering application can use it. The rendering application generates an image based on these parameters. The generated image is returned to the Web client through the CGI from the Web server.

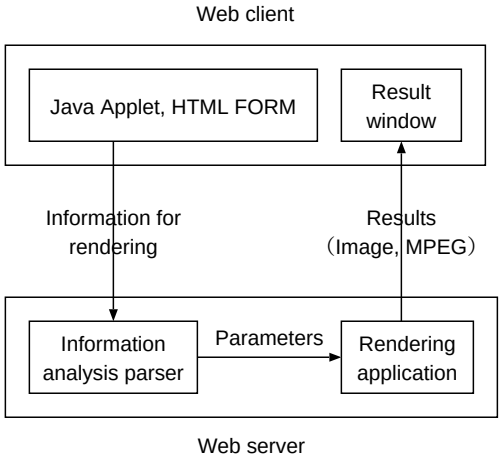


Figure 1: System Constitution

2.2 Processing on the Web Client

First, the information needed for generating the images is specified by HTML FORM and Java Applets in the Web client.

In the case of HTML FORM, the information necessary to create images, such as the kind and color of the object are selected from the range of information that was prepared. The connection with the Web server is achieved by calling the CGI. The call for the CGI to run is initiated when the SUBMIT button for FORM was pushed on the Web browser, and the information that was selected is sent to the Web server. When sending information through the CGI, although it needs to encode the information using certain rules, the Web browser usually does it automatically.

It is possible to specify the viewpoint position and the object location etc. interactively via a Java Applet. The connection with the Web server is carried out by calling the CGI using a `URLConnection` class. Unlike HTML FORM, encoding of the information is not carried out automatically, so it needs to be encoded using the `encode` method of the `URLEncoder` class. After sending the necessary information to the CGI, the Web client waits until the Web server generates the image. The rendering results are saved as output files, such as JPEG and MPEG files, in the Web server. An HTML file including the rendering results is generated, is sent back to the Web client, and is displayed in the Web browser.

2.3 Processing on the Web Server

2.3.1 Information Analysis Parser

The information analysis parser in the Web server receives the information that was given to the CGI from the Web client. Because the information is given as a character string, the information analysis parser is described by the `perl` language suitable for character string operation. The information analysis parser that received the information also decodes it. The decoded information is processed as a parameter file that the rendering application can use. A parameter file is saved once at the Web server. The form of the parameter file changes with different rendering applications.

2.3.2 Rendering Application

The rendering application is described by C, C++, or Java, etc., and starts as a call for the external application from the information analysis parser written in `perl`. *POV-Ray* (Hallam Oaks Pty., 1995-2000), *Radiance* (Larson and Shakespeare, 1998), etc., which are mentioned later, are used for the rendering application.

The active rendering application loads the parameter file generated by the information analysis parser, and performs the rendering.

3 APPLICATIONS OF WEB RENDERING

To give examples of the applications of this system, a metaball editing system, a lighting design system and a morphing system were mounted. The details of each system are described below.

3.1 Metaball Editing System

A metaball system has the capacity to smoothly fuse computer-generated balls to render an organic form. Since remarkable processing capability is required to achieve this rendering, we built a system that uses the Web to perform the display and editing of the metaballs.

3.1.1 Functional Outline

This system equips the Web client side with a metaball editor via a Java Applet. Metaballs are edited with the editor and the information about their locations, radii, viewpoint position, etc. is passed on to the Web server. Image generation is performed by the Web server once it

has received the information, and the result is displayed by the Web client.

3.1.2 Metaball Editor

The operation window of the metaball editor is shown in Figure 2. The editor is mounted by Java Applets. It consists of an edit screen on the right-hand side, an operation panel and a displaying status area on the left-hand side.

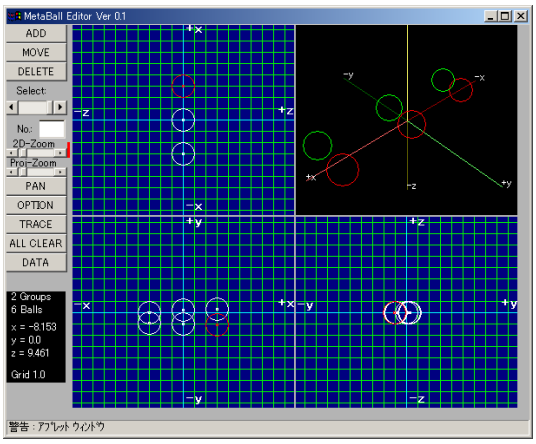


Figure 2: Metaball Editor

The upper right of edit screen is a perspective view, and the lower right, the upper left and the lower left serve as yz, zx, and xy plane views respectively. The arrangement of the current metaball is expressed in perspective as a wire frame, and it is possible to change the viewpoint interactively by dragging with a mouse on the perspective view area. The button and slide bar for each function are in an operation panel, and are used for performing addition of a metaball, movement, deletion, etc. User can specify parameters of each metaball such as the location, radius, concentration, color, reflection, attributes of penetration.

3.1.3 Preview

The Web client is also equipped with a Java Applet that gives a preview of how the arranged metaballs appears. It is useful for checking whether the fusion of the metaballs is performed as intended.

We adopted the method of Nishita and Nakamae (1994) using the Bezier Clipping method for preview. Although this method can perform rendering of metaballs at comparatively high speed, effects of reflection, refraction, shadowing, etc. is omitted. The Web server performs these effects.

3.1.4 Flow of the Rendering Process

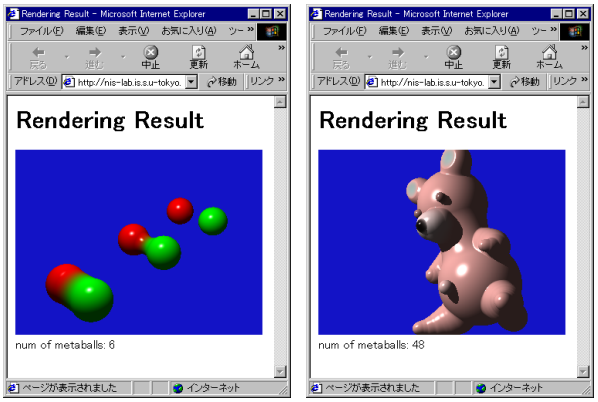
The rendering of metaballs is performed by a ray tracing application called *POV-Ray*. In *POV-Ray*, information, such as the viewpoint position, the location and other attributes of the objects are loaded from a parameter file called the scene file, and rendering is performed based on them.

Attribute information, such as the location, radius and concentration of each metaball, and viewpoint position are passed to the CGI, and the Web server then receives information from the editor. In the Web server, the information received is analyzed by the information analysis parser and a scene file is generated based on it. *POV-Ray* is called up as an external application, which then loads the scene file generated by the information analysis parser and performs the rendering.

3.1.5 Examples of Output Image

Example images generated by the Web server and shown in the Web browser are shown in Figure 3.

Number of metaballs is 6 in (a), and number of metaballs is 48 in (b) with shadowing.



(a) (b)

Figure 3: Result of Rendering Metaballs in Web Browser

3.2 Lighting Design System

The conventional interior lighting design problem considers both direct lights and interreflections of lights, and for which the ray-tracing method and the radiosity method are indispensable (on the ray-tracing method, see Whitted (1980), and on the radiosity method, see Cohen and Greenberg (1985) and Nisita and Nakamae (1985)). The number of light sources, luminous intensity distribution, light source arrangement, and light source color, etc. are the quantities to be optimized.

Because radiosity method requires high computational power, our proposed system is conceivable as usefulness. So we developed a new lighting design system using Web.

3.2.1 Functional Outline

The proposed system adopts the approach of latter which dosen't need trial and error. Also, our system assumes that all light sources are white. The positions user can put the light sources are specified in advance. We cannot put multiple light sources at the same position. In the optimization process, the system decides where to put a light source and which type of light source to use.

First, the user determines a viewpoint, and specifies illumination value in several points in a room using Java Applet. Information, such as a specified viewpoint position and illumination values, is passed to the Web server.

The Web server specifies the positions and the types of light sources, which fulfill the illumination value of each point passed from the Web client in the given viewpont position, by optimization. According to the specified position and type of light sources, a final image is generated and it is sent to the Web client.

3.2.2 Calculation and Optimization of Illumination Distribution

When the space is lit by light sources, the illuminance values in the specified regions increase. To get these values, the Web server renders images for each light source. We use the rendering application called *Radiance*. In each image, only one light source is turned on. We call these images intermediate images. By using these images, we can get the illuminance values of the specified regions lit by each light source.

The Web server measures the extent to which the calculated solution satisfies the illuminance condition specified by the user as follows. First, the Web server calculates the square of the difference between the calculated illuminance and the user specified illuminance at all positions where the user specified illuminance values which are sent from the Web client. Next, the Web server sums up the squares. We call this value the objective function. This objective function is the difference between the calculated solution and the specified illuminance condition. Therefore, by minimizing this objective function, we can get the optimum solution.

Although the method which used Hopfield Neural Network (Takahashi et al., 1993), the method using the genetic algorithm (Dobashi et al., 1998), etc. were in the optimization technique, we adopted the chaotic neural network (Aihara et al., 1990). The chaotic neural network is superior to the Hopfield neural network because it has the ability to escape from local minima. If we use the Hopfield neural network, we must choose by trial and error the initial conditions in order to prevent the network from falling into a local minimum. In contrast, if we use the chaotic neural network, the trial and error is not required and we have more possibilities of finding the optimum solution because of the chaotic neural network's ability to escape from local minima.

Since the position and type of light source which should be turned on are determined by optimization, the Web server generates image using *Radiance*. The generated image is sent to the Web client.

3.2.3 An Example of Output Image

An example image rendered by the Web server and shown in the Web browser is shown in Figure 4. The light source position was limited to 12 places, and types were limited to four types, 60W electric bulb, 100W electric bulb, 60W fluorescent light, and 100W fluorescent light.

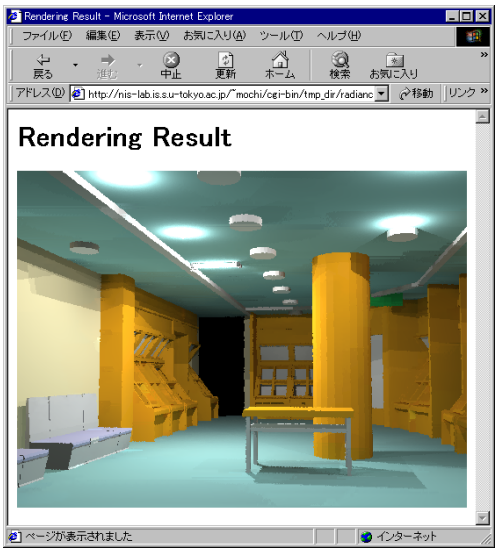


Figure 4: An Example of Lighting Design in Web Browser

3.3 Morphing System

Morphing is a technique that makes an animation that changes smoothly between two input images (calles the source image and the target image). For making animation, user specifies the correspondence relationship

between the source image and the target image, and then two images are interpolated and several intermediate images are generated. We construct such a system that the Web server generates an animation using morphing.

3.3.1 Functional Outline

In this system, the user specifies the correspondence relationship between the source image and the target image by using Java Applet on the Web client, and the information about the correspondence relationship is passed to the Web server. The Web server performs generating animation as an MPEG file once it has receives the information, and the result is displayed by the Web client.

3.3.2 Java Applet for Correspondence Relationship Specification

The appearance of the Java Applet for correspondence relationship specification is shown in Figure 5. The correspondence-related specification changes with different morphing techniques. Here, the user can make a choice between the mesh warping (Wolberg, 1990) and the field morphing (Beier and Neely, 1992).

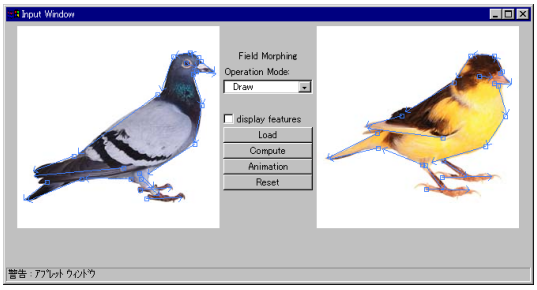


Figure 5: Java Applet for Correspondence Relationship Specification

First, the source image and the target image are loaded and the appropriate morphing technique is chosen. In the mesh warping, a mesh is displayed on the source image and the target image, the control point is dragged with a mouse, and a correspondence relationship is specified. The number of divisions on the mesh can also be selected. In the field morphing, a correspondence relationship is specified by the set of the line segments. The user can perform addition, movement, and deletion of a line segment. Moreover, the preset example correspondence-related data to the source image and the target image are available. Once the correspondence relationship of the image is decided, the information will be sent to the Web server.

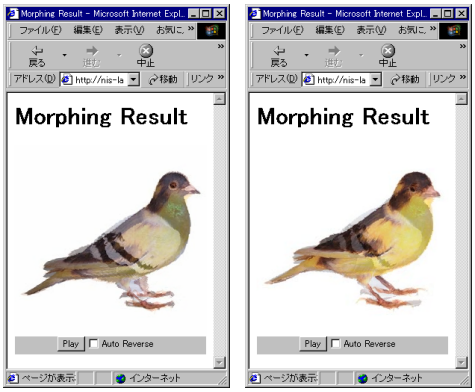
3.3.3 Animation Generation

The Web server which received correspondence-related information from the Web client starts the morphing application, interpolates the source image and the target image based on correspondence-related information, and generates several intermediate images. The generated intermediate images are then saved and are passed to the MPEG file encoding application. The MPEG file encoding application loads the stored images and generates an MPEG file. This MPEG file is sent to the Web client in the form of an embedded link in HTML so that it can download from the Web client side.

3.3.4 An Example of Output Image

The source image, the target image and parts of the intermediate images are shown in Figure 6. Sixteen intermediate images are generated when using the field morphing.

source image target image



intermediate images

Figure 6: An Example of Morphing

4 CONCLUSION

In this paper, a system that creates CG images via the Web has been proposed, and some examples of the system were shown. A computer that has a low processing capability can obtain a high quality image by performing the rendering on the Web server.

We intend to create various other examples of the system and to extend the fields of applications in the future.

ACKNOWLEDGMENTS

We would like to express our appreciation for the help given by Mr. Nanba of the Nishita Lab. for his assistance with a basic experiment of a lighting design, and to Mr. Koiso of the Nishita Lab. for his assistance with the morphing.

REFERENCES

Aihara, K., Tanabe, T. and Toyoda, M., 1990, "Chaotic neural networks", *Phys. Lett. A*, 144, 6, 7, pp.333-340.

Beier, T. and Neely, S., 1992, "Feature-based image metamorphosis", *Computer Graphics*, Vol.26, No.2, pp.35-42.

Cohen, M. F. and Greenberg, D. P., 1985, "The Hemi-Cube: A Radiosity Solution for Complex Environments", *Computer Graphics*, Vol.19, No.3, pp.23-30.

Dobashi, Y., Nakatani, H., Kaneda, K. and Yamashita, H., 1998, "An Interactive Lighting Design System Integrating Forward and Inverse Approach", *The Institute of Image Electronics Engineers of Japan*, Vol.27, No.4.

Hallam Oaks Pty. Ltd., 1995-2000, "POV-Ray - the Persistence of Vision Ray-tracer", http://www.povray.org/

Larson, G. W. and Shakespeare, R., 1998, "Rendering with Radiance", *Morgan Kaufmann Publishers Inc.*, ISBN 1-55860-499-5.

Nishita, T. and Nakamae, E., 1985, "Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection", *Computer Graphics*, Vol.19, No.3, pp.23-30.

Nishita, T. and Nakamae, E., 1994, "A Method for Displaying Metaballs by using Bezier Clipping", *Computer Graphics Forum*, Vol.13, No.3, pp.271-280.

NTT DoCoMo Inc., 2002, "Java Contents", http://www.nttdocomo.co.jp/english/p_s/i/java/

Takahashi, K., Kaneda, K., Yamanaka, T., Yamashita, H., Nakamae, E. and Nishita, T., 1993, "Lighting Design in Interreflective Environments Using Hopfield Neural Networks", *The Illuminating Engineering Institute of Japan*, Vol.17, No.2.

Whitted, T., 1980, "An Improved Illumination Model for Shaded Display", *Comm. ACM*, 23, 6, pp.343-349.

Wolberg, G., 1990, "Digital image warping", *IEEE Computer Society Press.*

ABOUT THE AUTHORS

Yoshinori Mochizuki, Dr., is a Research Associate of Computer Science Department at the University of Tokyo. His research interest is Computer Graphics. He can be reached by e-mail: mochi@is.s.u-tokyo.ac.jp, or trough postal address: 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-0033, Japan.

Tomoyuki Nishita, Dr., is a Professor of Complex Science and Engineering Department at the University of Tokyo. His research interests are Computer Graphics including Lighting Models, Hidden-surface Removal, and Antialiasing. He can be reached by e-mail: nis@is.s.u-tokyo.ac.jp, or trough postal address: 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-0033, Japan.