Journal for Geometry and Graphics
Volume 11 (2007), No. 2, 237–247.

An Educational Non-Photorealistic Rendering System Using 2D Images by Java Programming

Kunio Kondo[1], Tomoyuki Nishita[2], Hisashi Sato[3], Koichi Matsuda[4]

[1]*School of Media Science, Tokyo University of Technology*
1404 Katakura, Hachioji, Tokyo 192-0982, Japan
email: kondo@media.teu.ac.jp

[2]*Dept. of Complexity Science and Engineering, Graduate School of Frontier Sciences*
The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa City 277-8561, Japan
email: nishita@k.u-tokyo.ac.jp

[3]*Dept. of Information Media, Kanagawa Institute of Technology*
1030 Shimo-Ogino, Atsugi, Kanagawa 243-0292, Japan
email: sato@ic.kanagawa-it.ac.jp

[4]*Iwate Prefectural University*
152-52 Sugo, Takizawa, Iwate 020-0173, Japan
email: matsuda@iwate-pu.ac.jp

Abstract. It is important to improve the teaching and tutoring materials available in undergraduate CG education. Exposure to such material will certainly help encourage CG research. However, new image generation algorithms proposed in recent research are not usually included in tutoring material at the undergraduate level. We, the authors, have developed teaching material designed for educating undergraduates in CG. This paper describes teaching materials that support the study of NPR, or "non-photorealistic processing", a painting-style image processing technique used in CG education. Our goal is to make a tutoring system for studying NPR through Java programming. First, we will introduce "Jimmy", an educational Java software program for NPR. Jimmy supports a new brush and filtering functions for painting-style images. Second, we introduce a "CG Experiment Course Syllabus" for using the proposed system, and we describe the outcomes of its exercises. Students can develop new brush, filtering, and drawing techniques for painting-style images by extending and modifying the algorithms. Through using our system we found the following results:
(1) The proposed educational system was very useful for students in comprehending an algorithm by extending the existing source of the algorithm, and the

ISSN 1433-8157/$ 2.50

students were able to comprehend various painting processing methods by developing new image filters.
(2) The proposed educational system spurred an increase in the number of students interested in non-photorealistic image generation techniques.

Key Words: Computer graphics, non-photorealistic rendering, educational systems, Java

MSC 2000: 68U05

1. Introduction

This paper describes a new system for CG education based on Java software that was designed for Non Photorealistic Rendering ("NPR", or "painting-style," also known as "artistic style"). We have carried out teaching and resource development for CG education at both the undergraduate and graduate levels. We have found that in most institutions, however, that new image generation algorithms proposed in recent research are usually not handled at the undergraduate level. We've also found from our work teaching at the graduate level that utilizing such algorithms in an educational system at the undergraduate level is invaluable for future research and training at the graduate level. To do this, we designed a system that supports new brush and filtering functions for painting-style images. Using our system, students can develop new brush functions, filtering functions, and drawing techniques to create new effects that mimic painting-style images.

There are many examples of CG education at Japanese universities [11, 2, 12, 9, 5, 6]. Authors have already developed tutoring material for CG education at the undergraduate level [3, 4, 13, 14, 10]. It is usually comprised of basic CG, and generally at this level, new image generation algorithms proposed in recent research are not handled. We are not aware of any CG educational programs that use new image generation algorithms. At our institution, we have a CG education course at the graduate level that incorporates an important field of new image synthesis methods, and students have carried out experiments on the basis of the results of those practices at the graduate school since 2004 [7, 8]. However, the standard CG textbook in Japan was revised in 2005 to include realistic rendering techniques, however the detailed algorithms of NPR have not been introduced by the CG-ARTS Society.

In this paper, we explain tutoring material that supports study of the NPR painting-style image processing technique and its application in an undergraduate experiments course. We call the Java software program that performs these techniques on image files, "Jimmy". In this paper we also describe our syllabus of CG exercises using the proposed system and the result of the experiments designated in the syllabus. Moreover, there are provisions for the use of the program through the union of the input, the display, and the parameter input of the image that allow for easy improvements of existing algorithms. Our educational system "Jimmy" was developed to accommodate the following areas: Students developing the brush function, the filtering function, and to create a drawing technique for new painting-style images based on educational Java software for NPR. As students develop the applet using Java themselves, the system, its design, and the design of the course facilitates an understanding of Jimmy's algorithms and various image filter painting processes quite well.

Some features of the Jimmy system are as follows:

(1) Jimmy is open-source, and it is possible to expand its system by defining new processes that refer to the filter by modifying and extending the algorithm within the Java applet

program.

(2) Jimmy is a union of the input, the display, and the parameter inputs of an image, allowing students to better understand various image filter picture processing techniques.

At our institutions, third year undergraduate students in the Department of Information Science are trained using the Jimmy system. An experiment based on Jimmy is comprised of 15 sessions delivered every three days. A student proceeds through understanding the system, the algorithm and the system's painting-style processing in about one month. Students then make a report that the instructor evaluates and returns to the student with comments. Students must then write improvements to the algorithm, add their own experimental samples, and then evaluate and submit another report based on their improvements.

The positioning of this CG course in this department is as follows. C programming is taught in the first and second years. Then, in the third year students are introduced to Image Processing and Computer Graphics. The Computer Graphics course uses a textbook that introduces not only basic algorithms but also introduces programming for understanding the algorithms, and that programming is done in Java. The course is accompanied by an experiment from one of the themes of our "Exercises on Information Science" course in the second semester of a student's third year. A team of about six students practice this theme for three weeks. And in total, there are about 60 students taking this course and exercises annually.

In Section 2 of this paper, we give an overview of Jimmy. In Section 3, we show how to make image filters, and in Section 4 we describe CG education using Jimmy, a syllabus of CG exercises using the proposed system, and the results of those exercises. Lastly, in Section 5, we give an evaluation of those exercises and we finish with a set of conclusions and recommendations for further research on CG education.

2. An overview of Jimmy

In this section, we describe the usage of the Jimmy system according to Fig. 1. Jimmy is a Java applet, and there is a User Interface section for image input, a filter selection section, and a parameter input section. By simply creating filter algorithms, students can convert photorealistic images to painting-style, non-photorealistic images. Currently images of 800×600 pixels can be displayed in the applet.

Jimmy's menu is displayed on the right side of the screen, and only appears when needed. In the upper part of the menu, there are pull-down menus to select the image and the desired image filter, and when both image and filter are selected, parameters for the selected filter are displayed. Parameters can be set using slide bars, or other means, and can control the effect of the image filter. It is also possible to directly input the parameter in proportion to the content. After properly setting the parameters, users select a painting, press the execution button and a NPR image is drawn, based on a combination of the original image and the image processing filter parameters. In addition, changes of the filter parameters can be confirmed even after converting the image.

Jimmy is useful at the following three stages of CG education.

(1) *Generation of NPR images:*
Once a student starts the NPR applet and uses various filters, the applet uses various NPR techniques to generate a new image. By changing various parameters, images can be changed to suit the student's artistic tastes. A student can take a picture

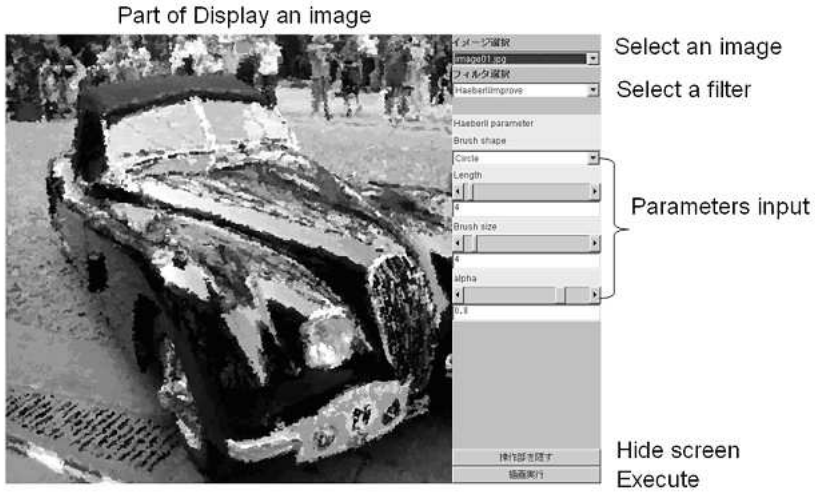

Figure 1: Proposed system Jimmy

and download the image data to the applet and convert it into another aesthetically appealing image.

(2) *Understanding art, style, and aesthetics through algorithms:*
Jimmy helps users understand the use of algorithms to generate and stylize images by allowing them to change various parameters, and view the results. Because students can see the Java source program, they can analyze the algorithms behind various artistic filters, processes, and effects.

(3) *The development of NPR algorithms:*
To encourage the development of new NPR algorithms, Jimmy's Java source program has been provided as open source. Using this source program, students can develop new filter processes. If each student develops even only a section of a new art filter, students can add to the library of available filters, techniques, and knowledge available for other students very quickly.

3. How to make image filters

In this section, we will describe how to make image filters. First, we describe the materials necessary for making an image filter, and then we explain the method for making the filter.

3.1. Materials for making an image filter

Below we describe a folder of materials necessary to execute the applet and to make an image filter.

First we place five classes in a folder, which we will name *imageFiltering* (see Fig. 2). Then, we create another nested folder inside the imageFiltering folder titled *images*, and we place either jpg or gif image files in that images folder. Students can organize their images and make them easier to find in Jimmy by adding the filenames of each image to a text file called *List.txt*, and placing that text file in the images folder. This will then add the names of the filters to the *"Filter Name List"* within Jimmy. Students can also add a class titled *Filter-name.class* in order to add names to List.txt. Currently image size must be 800×600 resolution.

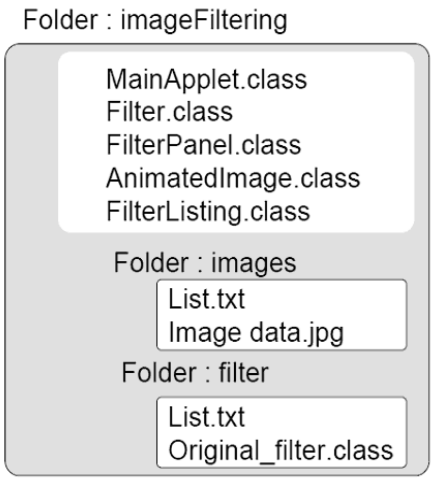

Figure 2: Contents of the NPR folder

3.2. The structure of classes

In this section, we describe the Java classes necessary to draw NPR images. Fig. 3 shows the structure of those classes.

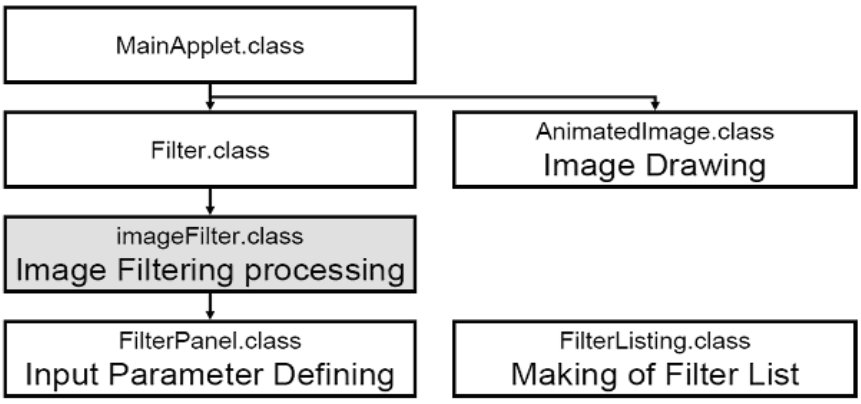

Figure 3: The structure of classes

MainApplet.class
Other classes are called from this applet.

AnimatedImage.class
The function of this class is to draw images. The image is operated through this class.

Filter.class
This is an abstract class for filters. Only the existence of an external specification will make use of this class necessary.

FilterPanel.class
This class specifies the parameters of a filter. It is displayed in the applet, and the parameters are specified and set in the filter.

FilterListing.class
This class updates the filter list. It is independent of other classes, and it can be used as a standalone Java program.

Filter.class
Filter.class and FilterPanel.class were inherited as inner classes.

3.3. How to Make an Image Filter

When a new filter is made, it establishes a class that inherits from Filter.class. The class that inherits getName() filter(), getFilterPanel(), and FilterPanel.class is described as an inner class. After students make the class of a new filter, they put it in directory */imageFiltering/filter/*. The following are classes that inherit from Filter.class:

getName()
returns the name of the filter without changing the name on the way.

Filter()
Operates the Animated Image class and performs new image filtering.

getFilterPanel()
Returns a FilterPanel class that operates a void filter. The inner class below inherits from FilterPanel. Its information is displayed on the right side of the Applet as a panel, and the parameter specifications for the filter are set in the receipt filter. (The parameter can be received in various ways: button, text field, etc.)

The setFilterParameter()
This class is used to set the void parameter for the filter.

3.4. How to Add Filters

All the class files written in "List.txt" in the */imageFiltering/filter/* directory are read each time a filter is used. Other program sources need not be changed. When a class file is added to the directory, "List.txt" is adjusted. When "image- Filtering/FileListing" is executed, the list file is made and all the class files are added to the list.

3.5. How to Add Images

The "use image" command reads all the picture files written in the "List.txt" file found in the directory /imageFiltering/images/. The user needs not change the program source; she must only add a jpg or gif image directly to the directory and correct the "List.txt" file accordingly. As long as "image-Filtering/FileListing" has been executed at least once, a list file will be present.

4. CG education using Jimmy

This section describes the contents and the results of experiments performed by third year students at the Department of Information Science at our schools. The theme of the exercise was to make an NPR filter using Jimmy, and to generate a non-photorealistic stylized image from a photo-realistic one. Each student designs an painting-style image processing technique by referring to and modifying Jimmy's source. In our exercise, students were asked to make an applet every three days for a grand total of 15 applets. The following is a list of discoveries made during the exercise process.

4.1. Tasks

The CG Educational exercise using Jimmy was done 15 times, each time over the course of three days (for a total of 45 days). During the exercise students were given the following tasks, and by fulfilling these tasks, they were able to create a painting-style filter each time.

Task 1: *Understanding the outline of the "Jimmy System"*

Task 2: *Understanding the Java behind "Imagefilter"*

Task 3: *Executing & programming Java for non-photorealistic rendering*

Task 4: *Designing an algorithm for artistic rendering*

Task 5: *Programming the Java behind "Imagefilter"*

Task 6: *Experimenting with drawing using Jimmy*

Task 7: *Writing an evaluative report.*

During the course, the following tools, items, and references were also shown to the students.

[The First Week]

(1) We introduce the students to an outline of Jimmy's system. Then, the students arrange a flow diagram to help and show that all of the members of their group understand the content of the Jimmy program.

(2) A reference program is compiled, and is added to the system.

[The Second Week]

(3) Together with the students, we analyze painting and drawing methods necessary for the generation and processing of a painting-style non-photorealistic image used in Task 3.

(4) Together with the students, we design a drawing algorithm and clarify the programming.

(5) Through this, we begin developing an imageFilter.

[The Third Week]

(6) We continue development of the Java behind the imageFilter.

(7) Students write a report on their work, findings, process, and evaluations.

Through this exercise, we can guide students in designing their own original non-photorealistic rendering method based on their own ideas, using Jimmy's programming as a reference.

To summarize the exercise:

(1) *The students set the problem.* Students analyze a photo, and then summarize the rules necessary to render the photo in a non-photorealistic (artistic) way.

(2) *The student creates the solution.* Based on the original photo and the rules students themselves delineated in the above step, students then write an algorithm for the drawing filter they imagined.

4.2. Sample results from the exercise

In this section, we show examples of the NPR images students were able to create with their own algorithms and Jimmy.

Fig. 4 is an example of a painting-style image generated using a square brush and the Haeberli method.

Fig. 5 is an example of a painting-style image generated using the oil painting method and a subtractive color mixture. The effect can be controlled through changing the weight of a CMY parameter. It is an example of an algorithm that was created by expanding on a previous one – the previous one being a form of pen-picture processing that used the colors within an image to perform it process.

Fig. 6 is an example of a picture that has been processed with a filter that causes diffuse edges after a Gaussian filter has been used. We can also change black edges to white by using this filter.

Fig. 7 is an example of an image filter that causes the photo to look like an image drawn with crayon on drawing paper. The student that designed the algorithm for this filter improved upon the Haeberli method.

Fig. 8 is an example of a similar filter, however instead of crayon, the filter creates an image with a colored pencil-like look. The strokes in this method are made by using the Bresenham algorithm. After the user inputs the number of colors, an approximation color is

Figure 4: Square brush

Figure 5: Oil painting method by CMY

Figure 6: Edges extraction method

Figure 7: Crayon method

Figure 8: Color pencil method

Figure 9: Oil painting method by strokes

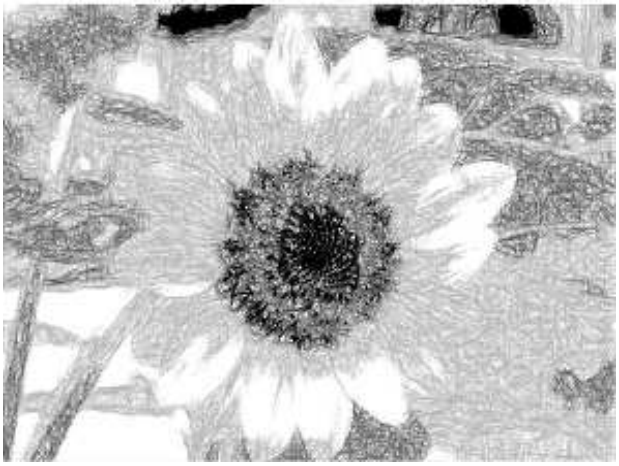

Figure 10: Pencil drawing method

Figure 11: Outline extraction by Sobel

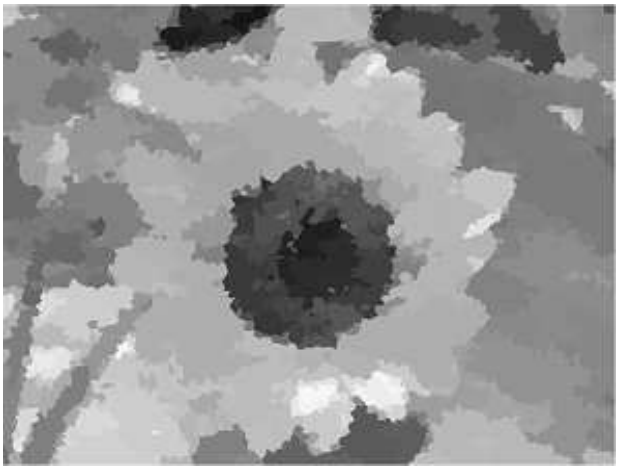

Figure 12: Area division method

calculated, and a new color for each stroke is decided.

Fig. 9 is an example of a filter that effects a look of an oil painter who uses a thick stroke. There are two parts in this method. The first part is to draw small areas of the image using thin brushes, and to paint large areas with thick brushes. The second part is to draw outlines around various areas of the image.

Fig. 10 is an example of a filter that uses an algorithm that is an expansion upon the one used in Fig. 7. Using Fig. 10's algorithm, we can effect upon the photo a look that feels like a pencil drawing.

Fig. 11 is an example of a filter that effects a fantastically bright look upon an image. The outline is extracted using the Sobel method and Brightness information on HSB.

Fig. 12 is an example of a filter that effects the look of an illustration by using the area division method. This method uses a Voronoi diagram to divide different color areas.

Before we began using Jimmy in our undergraduate student exercises, teachers were often caught explaining the answers to tasks, or simply giving problems with answers that the students already knew. Therefore, there was little creativity or motivation from the student side, and it was apparent that students often did not fully understand the problems or the solutions given to them. There were also several instances of reports and work that seemed as if it had been copied numerous times from other students.

Therefore, we have found very useful benefits to using the Jimmy system in our courses:

(1) Students are motivated to think for themselves about the target of the non-photorealistic rendering, discover the rules necessary to render the photo the way they want, and consequently their understanding of image processing deepens.

(2) Because students propose their own methods, filters, and algorithms, all work is completely original, adding not only to the knowledge of each student, and to the program's source, but to the knowledge of students to come.

5. Conclusions

It is important to improve the tutoring material used to help CG education at the undergraduate level. Exposure to such material will certainly help spark renewed interest and development in CG research. This report proposed a form of teaching material that supports the study of non-photorealistic image generation and processing techniques. These techniques support several new brush functions for use on photos to convert them to painting-style images. Those techniques are based on an educational Java software program designed for non-photorealistic rendering (NPR), filtering functions, and the study of image-drawing techniques. This software, which we dubbed Jimmy, is a CG training system for use in undergraduate CG courses.

From our study of Jimmy and its usage, the following findings were made:

(1) After implementing our proposal to use "Jimmy" in a way that supported the learning of artistic painting techniques, students' interest in the NPR technique increased.

(2) After introducing Jimmy into our course exercises, students were able to deepen their understanding of image generation processing by programming their own algorithms.

(3) Much of the content used in our Jimmy exercises was successfully used in other areas of CG training in all of our universities, proving that the open source arrangement of the system allows the material to be utilized in various ways and in various universities.

Acknowledgments

We would like to thank Mr. Kazuhide Hatsuyama, Mr. Toshiyuki Haga, Mr. Yasushi Tojo, and Mr. Hiroki Imahashi. This research was partially supported by the Matsushita Education Foundation, and the Japan Science and Technology Agency (CREST).

References

[1] P. Haeberli: *Paint By Numbers: Abstract Image Representations*. Computer Graphics **24** (4), 207–214 (1990).

[2] K. Kajiyama, K. Suzuki: *Report on the 26th Graphic Education Forum*. Journal of Graphic Science of Japan **35**/1, p. 21 (2001).

[3] K. Kondo, K. Ogata, H. Sato, S. Shimada: *Simple Graphic Tool Xgt on X-Window and Education of Computer Graphics for Beginners*. Proc. China-Japan Joint Conf. on Graphics Education 1993, pp. 169–173.

[4] K. Kondo: *The Integration of Computer Aided Visual Communication and Visual Thinking in Computer Science Education*. Proc. China-Japan Joint Conf. on Graphics Education 1995, pp. 131–136.

[5] K. Kondo: *"Survey" Education of Computer Graphics*. Special issue of Journal on Graphics Science, 73–74 (1997).

[6] K. Kondo: *Report on the 26th Graphic Education Forum, CG/CAD Education at Saitama University*. Journal of Graphic Science of Japan **35**/1, 22–23 (2001).

[7] K. Kondo, T. Nishita: *NPR Educational System by Java programming*. Proc. Japan Society on Graphics Society 2004, pp. 7–12.

[8] K. Kondo, T. Nishita: *"Jimmy" NPR Educational System using Java*. Proc. NICOGRAPH2004, pp. 21–22.

[9] S. Nagashima, H. Isoda: *An Attempt on Computer Graphics and CAD Education*. Proc. of ICECG, 1984, pp. 474–481.

[10] T. Nishita, K. Kondo, Y. Ohno: *Development of a Web Based Training system and Courseware for Advanced Computer Graphics Courses Enhanced by Interactive Java Applets*. Proc. Internat. Conf. on Geometry and Graphics 2002, vol. 2, pp. 123–128.

[11] K. Suzuki, K. Takeyama, S. Nagano: *Implementation of Computer graphics into Graphic Science Courses*. Journal of Graphic Science of Japan **44**, 5–12 (1988).

[12] K. Suzuki, H. Suzuki, Y. Yamaguchi, S. Nagashima, S. Nagano: *Integrated Descriptive Geometry and Computer Graphics Course at the Unversity of Tokyo, 1992 Update*. Proc. of the conference of Japan and China Graphics Education, 1993.

[13] M. Takahashi, H. Sato, K. Kondo, S. Shimada: *A manual to teach computer graphics by Java*. J. Geometry Graphics **2**/1, 101–108 (1996).

[14] M. Takahashi, H. Sato, K. Kondo: *A Remote Education System of Computer Graphics Education using Java*. Proc. 3rd China-Japan Joint Graphics Education 1997, pp. 228–233.

Received August 7, 2006; final form December 28, 2007