A Visual Simulator for Computer Education by Java Programming

Yoshiro Imai*1, Shinji Tomita*2, Hitoshi Inomo*1, Wataru Shiraki*1 and Hiroshi Ishikawa*1

*1 Faculty of Engineering, Kagawa University
*2 Graduate School of Informatics, Kyoto University

Faculty of Engineering, Kagawa University,
2217-20 Hayashi-cho, Takamatsu, 761-0396, JAPAN
Tel: +81-87-864-2244 FAX: +81-87-864-2244 E-mail: imai@eng.kagawa-u.ac.jp

Abstract
We have developed a visual simulator for computer education tool. It is designed to illustrate an internal behavior and structure of computer and explain graphically how a computer works. It is implemented in Java programming language to be executed on several computing environments such as Windows PC, Macintosh and many Unix workstations. It works as both stand alone application (Java appli) and Java applet so that we can distribute it from Web server by means of internet. Its GUI is not so rich but even beginners operate it very easily, and it employs Java-provided thread programming for interactive simulation. This report describes design concepts of our visual simulator, implementation detail with Java programming, and usages as an education tool of classroom lectures.

Ｊａｖａ アプレットによる計算機ビジュアルシミュレータの実現

今井慈郎*1, **富田眞治***2, **井面仁志***1, **白木 渡***1, **石川 浩** *1
*1 **香川大学工学部信頼性情報システム工学科** *2 **京都大学大学院情報学研究科**
〒761-0396 **香川県高松市林町** 2217-20 **香川大学工学部信頼性情報システム工学科**
Tel: 087-864-2244 FAX: 087-864-2244 E-mail: imai@eng.kagawa-u.ac.jp

あらまし
Java **プログラミングによる計算機内部の動作・構造を理解させるためのビジュアルシミュレータについて，その設計方針，実装方式**(GUI **やスレッドプログラミングなど)そして大学・高専及び企業の新人教育などにおける教材としての利用状況などについて，具体的に報告する．**

1. Introduction

The more frequently many people use computers, the higher-powered and more complicated actual computers grow. It is too difficult for beginners to comprehend how computers work precisely because of their complex properties like as a black box. Recently, however, several curricula of advanced educational institution, such as high school, university and new comers' training center of company, have information literacy courses and introductory lectures for computer science. In order to study computer more precisely, it is necessary to understand the internal structure and behavior of computer. Not an abstracted lecture on Computer System can provide visual and applicable understanding on structure and behavior of computer.

We have developed a visual simulator called "VisuSim (/vizim/)" as computer education tool. With this tool, it is convenient for beginners to understand internal structure and behavior of computer visually. VisuSim offers window-based graphical view to explain how computer works, interprets sample assembly programs stored in memory and demonstrates data transfer mechanism between registers of CPU and memory. It is designed to fulfill its function not only as stand-alone application on several platforms but also as Java applet on major browsers.

In this paper, we describe basic design concepts of our visual simulator, its system configuration implemented with Java programming language, characteristics of its GUI for actual manipulation, and its application to practical lectures of computer system and/or computer literacy.

2. Basic Concepts of Visual Simulator Design

Our visual simulator: VisuSim has been designed and developed for the following major objectives:

1) A demonstration tool for assistance to complete oral explanation how computer works in ordinary classroom

lectures,
2) Self-learning software, which can be obtained from web server, to review and confirm already-lectured contents after school.

These design objectives may be suitable for almost education tools or self-learning software as well as our visual simulator. It is important that education tools will be available as widely as possible and can be manipulated easily. If such a tool is designed only for a handful of specific users, it cannot enjoy periodic improvement for lack of feedback and request from other users.

Now we will mention the basic concepts to design our visual simulator. At first, basic design concepts of our simulator are enumerated, and secondly commentated for some items of concept as follows:

- Education-aid demonstration tool
- Self-learning assistant software
- Enhancement of visual facility
- Graphical user interface
- Easy improvement and quick distribution
- Employment of Java programming language
- Browser-based execution environment

Distribution of software may utilize computer network connectivity. Web sites provide downloadable data and binaries on their home pages, accept access from anyone through network, and transfer data or binaries according to request. So it is quite natural that we choose a design policy to utilize software distribution style based on network and web data transfer service. A famous methodology of software design, "Stepwise Refinement/ Enhancement" advocated by Niklaus Wirth[2], teach us that it may be effective for many types of software to be designed and implemented from a simple prototype into more complicated products through several versions. So we have employed the following design procedure of visual simulator: designing lower version of product, opening the implemented version to the public on our web site, obtaining some reports on using and evaluating it by students and designing higher version with error decrease and enhancement based on usage reports.

We must choose another design policy to build software based on windows-based operability with simple manipulation such as button pushing and so on, because all the PC's already adopt multi-windows system. Selecting programming language is one of the most serious problems whether system and/or software will be successful or not. In the general, C/C++, Visual BASIC or Java may be powerful and practical candidate of software description languages. With most regard to cumulative results till now, C/C++ must be chosen. Selection based on popularity will persuade us that Visual BASIC has obtained maximal numbers of users in the PC world. We decide to choose Java as description language, however, because Java can provide both of environment-independent executable binaries and window-based operability for multiple platforms[1].

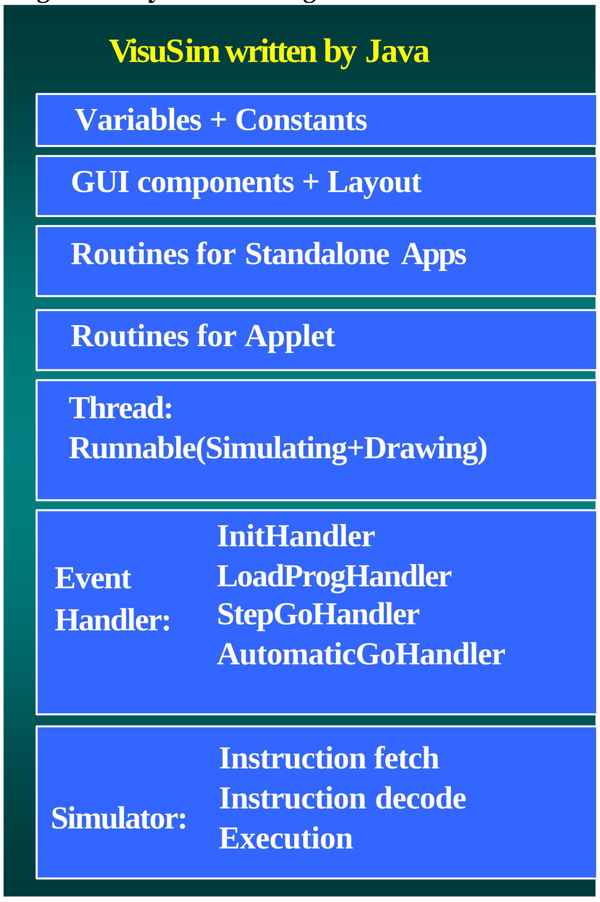
3. System Configuration of Visual Simulator

This section describes detail on system configuration of our visual simulator, namely, implementation with Java, simulation capability, operability with GUI and more effective features for enhancing educational demonstration. From now, we denote VisuSim as the abbreviation for our visual simulator. The pronunciation of VisuSim is /vizim/.

3.1 Implementation of VisuSim with Java

VisuSim is written in Java language, whose software codes has included about 2,000 line statements. And the current version of it has been organized with the following subsystems:
(1) GUI Components + Display Layout,
(2) Routines working for Standalone Applications,
(3) Routines working for Java Applet,
(4) Several Threads for Concurrent Operation,
(5) Event Handlers for Man-Machine Interaction, and
(6) Simulator for Interpreting Pseudo-Assembly Codes.

Figure 1 System Configuration of VisuSim



Subsystem (1) plays a role of user interface, whose components consists of bottoms, labels, and text fields for information interchanging between Java applet and user who operates the corresponding browser. Display Layout may be one of the most important factors to define practice of GUI and to determine whether that system is convenient to use or not. With Java programming style, it is easy to select suitable GUI components and put them in the very effective position.

Subsystem (2) is going to work whenever VisuSim is invoked as stand-alone application program with java interpreter, while subsystem (3) begins to operate after VisuSim is downloaded as Java applet into web browser. The both subsystems play almost same role of presetting variables and states of the whole system, making event handlers be ready to serve, and preparing for major functions of VisuSim, for example, system initialization, program loading, pseudo-assembly codes interpretation and so on. VisuSim includes both of the above two subsystems in its single source code, so that it can facilitate the two different activities. Namely, even one type of VisuSim can work correctly as either standalone application or Java applet. In addition, a single source code approach may give us another advantage, that is convenient to update and easy to maintain source code of VisuSim.

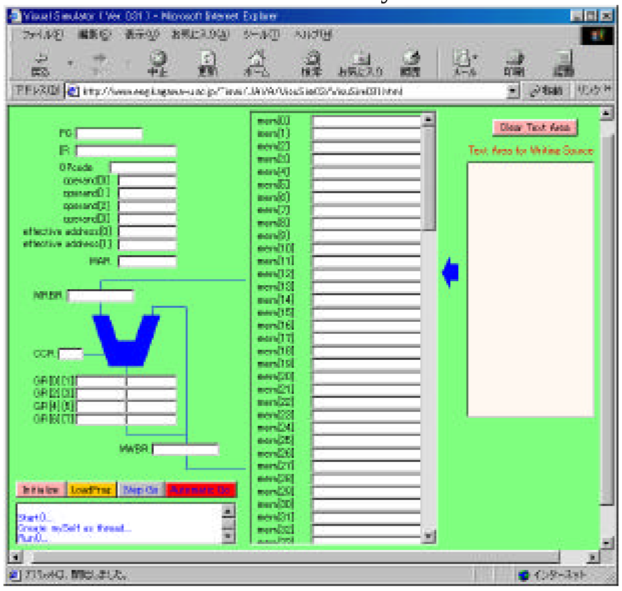
Subsystem (4) has some routines for thread, which are introduced for VisuSim to operate concurrently. This subsystem is designed and implemented according to formal thread description methodology of Java, because it is difficult for us to write entirely corrective program realizing concurrent operations with Java language. Both simulating and drawing are multitasking, therefore, they can work concurrently. Event-handling routines play essential roles of interaction, namely, they relate manipulation of GUI components by user to the corresponding inner routines of VisuSim, so subsystem (5) must be very much principal in VisuSim. This subsystem can be initialized by subsystem (2) as well as subsystem (3), may be invoked by user's operation for GUI components, and will be carried out to start some event handling services. Moreover a part of them are triggered to be changed into a multitasking thread to computing suitable processes, and then return the results in accordance with events.

Subsystem (6) is the main part of VisuSim. It consists of the following three major routines: instruction fetch routine, instruction decode one and execution one. All three routines and the following data area and text fields are combined to organize virtual computer hardware in a close relation of corresponding real hardware structure. At first, instruction fetch routine reads an instruction located by program counter (PC) out of memory array, and transfers it into instruction register (IR) built in control unit. Secondly, instruction decode routine investigates the content of IR, deciphers it into specification of operation code and operands, and throws those specified signals to the suitable units, which recognize what they must do.

Finally, execution routine indicates that every unit should operate correctly along the received order. The three addressing modes such as direct addressing, indirect one and immediate one, also interpreted by execution routine. Almost routines described above have own inner variables and states, and compute their proper processes with little dependence on other routines. Such routines can be tuned and updated by themselves without influence of others because of benefits from object-oriented programming. As a matter of fact, we wanted to write pure modular programming code, brush up every routine of VisuSim, and decrease a whole number of global variables in the source code of VisuSim. Frequently, each routine can hardly realize smooth co-relation with other parts of routines only by message passing mechanism, at the result of global variable decrement. Therefore, we must stop to decrease number of global variables and employ fairly conventional programming style with the suitable global variables for corresponding each other

3.2 GUI of VisuSim and its Operability

First of all, we show the overview of VisuSim in Figure 2. This figure demonstrates that VisuSim is working on the browsing window of Internet Explorer, which is a Japanese Edition bundled in Microsoft Windows 98 for almost Japanese Windows PC's. Needless to say, another major browser such as Netscape Composer can also provide an execution environment for VisuSim. In such a case, VisuSim is invoked as Java applet so that it has hidden some GUI components for file access service facilities in order to inhibit security violation.



```
Start()...
Create myself as thread...
Run()...
```

Figure 2 Overview of VisuSim's GUI

The GUI window of VisuSim consists of major three parts for computer hardware and some GUI components for interaction between user and VisuSim. As shown in figure2, the major three parts for computer hardware are represented with control unit, processor unit and memory unit. Control unit has some objects constructed with text fields which play roles of PC, IR and other registers respectively. Processor unit as well as control unit has some objects constructed with text fields which play roles of General-purpose registers (GR[0]-GR[7]), two types of memory registers (MRBR, MWBR), Condition-code register (CCR) and so on. Memory unit is nearly the same. It has been entirely implemented on slide-movable panel object so that it can partly show location of array-structured memory cells. And view of contents in memory unit can be slid as the occasion demands.

VisuSim has four bottom objects such as Initialization bottom, Program Load one, Step-wise Execution one, and Automatic Execution one. These are prepared to control VisuSim from user and accept external requests. For example, pushing Initialization bottom, the major three units of VisuSim are reset so that IR, eight General-purpose registers, all the memory cells has been clear and PC is set into zero. Program Load bottom is used for transferring an assembly program written in Program text field into memory unit. If VisuSim is invoked as standalone Java application, pushing of Program Load bottom can perform direct access to the file system, read a program stored in file, and transfer it into memory unit.

The two execution bottoms are used for start and stop simulator subsystem of VisuSim directly. One is for step-wise execution instruction by instruction, and another is for automatically executing a series of assembly codes until halt instruction in the codes is fetched and decoded. A message field below four bottoms is placed to display current internal state of VisuSim. After pushing bottom, a request from user is accepted through event handler and processed by the according routines of VisuSim. The internal state of VisuSim is updated and the corresponding message is output at text field for monitoring. Such a message is related to its current internal state like “LoadProgFromTextArea: finished.”

3.3 Simulation Capability of VisuSim

This section describes detail of simulation capability concerning VisuSim. Its capability may be partly evaluated by means of instruction repertory which an objective simulator can interpret. Namely, the instruction set of target virtual computer system will be one of measurements for simulation capability. Table 1 shows the repertory of instruction set for VisuSim. Classification of instruction set can be formally divided into following four groups: 1) Control instructions including "halt" and "noop" (noop means 'No Operation'), 2) Jump instructions including subroutine-related, conditional-jump and unconditional-jump operations such as "call", "ret", "jpgt", "jpge", "jplt", "jple", "jpeq", "jpne", and "jump", 3) Unary operation instructions including arithmetic and stack-related operations such as "neg", "push" and "pop", and 4) Binary operation instructions including arithmetic operations such as "add", "sub", "move" and logical operations such as "and", "or", "xor".

Each operation works simultaneously together with condition-code register(CCR), which consists of Negative flag and Zero flag. CCR of VisuSim is represented as anyone of (N, Z)=(- , -), (- , Z) or (N, -). All the instructions listed in table 1 may take maximum two operands, which can be specified as the following four ways of operand specification: direct address, register identification, indirect address with register modification and immediate value.

Direct address is the most normal addressing, which uses only operand field and specifies target location of memory. Register identification is another addressing which means using register as either source or destination of operation. “RegId” or “Reg” is defined to represent the specific register from GR0 to GR7 in our case. Indirect address with register modification is more powerful addressing so that it is indispensable to utilization of indexed addressing, for example. Immediate value is a very convenient addressing and let us describe brief assembly programs. Without this mode, we must secure location of memory for all the data to be manipulated in programs.

In general, iteration processing structure can be realized with combination of conditional jump, indirect addressing and so on. Therefore, relatively complicated iteration processing can be realized with such an addressing and interpreted by VisuSim. Program including iteration is one of the difficult subject matters of computer system. Beginners sometimes suffer from lack of suitable education tools which assist illustrating how computer works. With VisuSim as visual education tool, it is available for beginners to understand visually a mechanism that program with iteration is processed by computer.

3.4 More Effective Features of VisuSim

This section describes actual features which make VisuSim more effective to be used as an education tool for computer system. With simulator, an overview for processing sequence of program can be obtained through iteration of instruction fetching, instruction decoding, and executing. It is very useful to specify the flow of data transferring, namely, to indicate when and where data is read out or written in. Therefore, VisuSim has two types of indicating function for memory access to make easy understanding of computer's behavior. Color of the memory location turns to "blue" when the content of memory is fetched by CPU, that includes both of instruction fetching and data fetching. On the other hand, color of memory cell has been changed into "yellow" on storing data at that location.

In the same manner, when data is transferred into a general-purpose register, color of that register is changed into "yellow". The above indication can make VisuSim a more effective education tool for computer's behavior and assist user to comprehend how data is transferred in the computer's inner space. There is another idea which simulator has multiple execution modes for processing program. One is "non-stop" execution, which is suitable to explain what the target program does, while other is "step-by-step" execution which is convenient to examine every part of the target program closely. From these viewpoints, we consider that simulator must prepare two different types of execution modes. Actually, execution modes of VisuSim are twofold: automatic type for "non-stop" execution and stepwise one for "step-by-step" are summarized as follows;

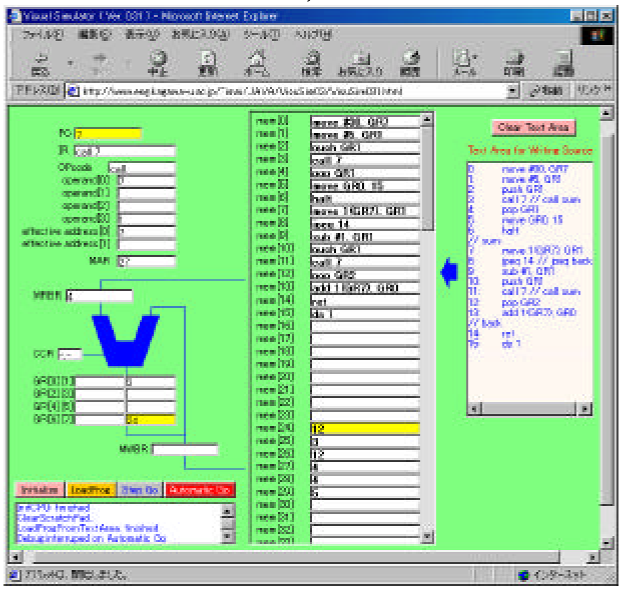


Figure 3. Automatic Execution Mode Using "Automatic Go" bottom

1) Automatic Execution (shown in Figure 3):

When the bottom for "Automatic Go" is clicked, the internal simulator of VisuSim starts on continuous repetition of machine cycle execution until decoding "halt" instruction or occurrence of interruption. This mode is much convenient for relatively long-term demonstration where VisuSim must interpret a program including several iterations, recursive procedure calls or other kind of complicated processes, for example. Because this mode of execution looks like playing slide show, it is useful to trace a whole target program sequentially and verify whether program is computed accurately or not. While simulator repeats its machine cycles in the automatic execution mode, pushing twice the bottom for "Automatic Go" triggers off occurrence of interruption.

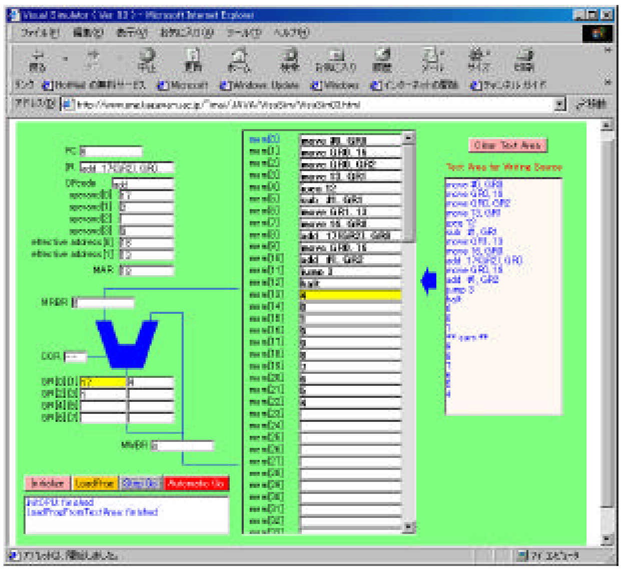


Figure 4. Step-wise Execution Mode Using "Step Go" bottom

2) Stepwise Execution (shown in Figure 4):

When the bottom for "Step Go" is clicked, the internal simulator of VisuSim activates only one machine cycle execution. A machine cycle of this mode strictly consists of instruction fetch, instruction decoding and execution. so sequential execution of the specific part of program needs consecutive operations of clicking the bottom for "Step Go". This execution mode is also convenient for interactive modification of registers and/or memory. It is easy to stop execution of program at any point, change contents of memory and/or CPU (including several registers), and restart from such a broken point.

This mode could make VisuSim compute only the limited part of long size program code, if program counter is reset to the specific location and the bottom for "Step Go" is being clicked during the suitable times. Such a process can point out the illegal description of programs, offer a better modification for improving, and finally prove code consistency of those examined programs. A detail testing of program can be realized in this mode.

Consequently, combination of two execution modes may provide a powerful debugging facility. In the class of assembly programming exercise described below, for example, these debugging facility has been employed to verify whether programs written by students work correctly or not. With both of stepwise and automatic execution mechanism, it is not only effective to debug program through some exhaustive method of investigating every check point, but also educational to demonstrate the relation between numbers of repetition and the corresponding content of loop control variable in the specific general-purpose register (GR[i]).

4. Evaluation for VisuSim as Education Tool

This section describes some applications of VisuSim into educational field, and brief evaluation of VisuSim as Web-based education tool by means of its actual usage.

4.1 Education Tool for Lecture

With VisuSim, it is efficient for teachers to explain the internal mechanism of computer and illustrate how computer works at their classroom lectures. For Example, a practical example of program such as addressing mechanism can provide useful images to students who have begun to study computer science. They will think of the reason why computer has some kinds of addressing through actual programs with direct addressing, indirect addressing and immediate value addressing, and find that program status of CPU is very much important to utilize several conditional branch operations.

Two kinds of execution mode provide a visual tracing function to investigate how program is processed in computer. Changing the value of loop counter makes control of iteration, which is realized with the following operation such as stopping execution, modifying and restoring data, and restarting from the location of interruption. Dynamic demonstration tool enables interactive communication between teacher and students concerning their common theme, realtime check and verification of anyone's question, quick application to the related problems, and then practical decision based on visual examination. Figure 4 illustrates image of lecture with VisuSim as education tool.

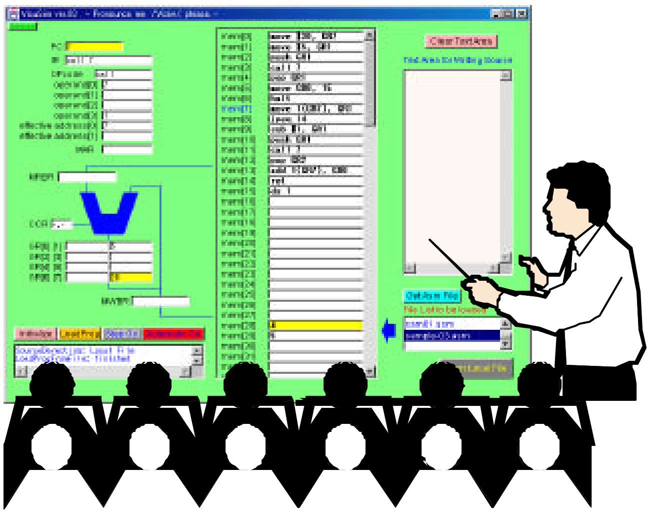


Figure 5 Image of Lecture with VisuSim

4.2 Self-Learning with Web-based Education Tool

VisuSim can be downloaded from our Web server at any time so that it is easy for students to obtain it at any place wherever it is available to connect with internet, even if they didn't attend lecture or couldn't understand that content at their class.

With usage of VisuSim as self-learning software tool, students have benefits to utilize VisuSim to review the content of lecture, investigate whether their programs work correctly or not, and verify their reports for homework by themselves. Figure 5 shows schematic diagram to explain that VisuSim can be downloaded from Web server at any place with connectivity to internet. Mobile computing facility enhances some students to utilize VisuSim outside of school and/or home, so they can study and demonstrate the contents of computer system, information literacy and/or assembly programming at any time and any place if they want.

Moreover, from the viewpoint of software developers , Web-based software distribution has some following benefits; namely, quick and low-cost release of the latest version for products, direct response of positive / negative estimation form users, brief improvement and modification of products into update version, and acquisition of kind and powerful cooperation from excellent users through the internet.

5. Conclusion

We can conclude our study as follows:

(1) It is recognized that our visual simulator, VisuSim is one of effective education tools for computer-related education and training by means of its practical usage.

(2) With use of VisuSim, graphical demonstration can be available in classroom lecture on computer system and information literacy, so that it is efficient for even beginners of computer to understand precisely the internal structure and behavior of computer.

(3) VisuSim can be easily obtained from the web-site, sufficiently executed platform-independently, and simply operated through its GUI and interactive facilities.

(4) VisuSim may be a good example for Web-based education tool. Execution environment is limited to Web browser such as MS-IE and Netscape Communicator so that no other software will be required in order to utilize VisuSim on any computing environment.

Acknowledgments

The authors would like to express special thanks to Prof. Haruo Niimi, Kyoto Sangyo Univ., and Prof. Kazunori Yamaguchi, Univ. of Tokyo for their constructive advices. This study is partly supported by grand 12040107 in aid for scientific research from the Ministry of Education, Culture, Sports, Science and Technology.

References

[1] David Flanagan, "JAVA EXAMPLES IN A NUTSHELL, A Tutorial Companion to Java in a Nutshell", O'Reilly & Associates, 1997

[2] Niklaus Wirth, "Algorithms + Data Structures = Programs", Prentice-Hall, 1976