Programming for Students of Information Design

Taeko Ariga
Department of Information Design
Kyoto University of Art and Design
24 Kamihate-cho Kitashirakawa Sakyo-ku
Kyoto, Japan, 606-8252
ariga@acm.org

Hideki Tsuiki
Department of Integrated Human Studies
Kyoto University
Yoshida Nihonmatsu-cho Sakyo-ku
Kyoto, Japan, 606-8501
tsuiki@i.h.kyoto-u.ac.jp

Abstract
Programming is one of the subjects which students of information design need to study. Programming training will improve their multimedia representation, even when they use software packages. It will also enlarge the possibility of new interactive multimedia art over the Internet. We propose course materials to teach programming by Java, which have been applied to a programming course in the Department of Information Design.

Introduction
The Department of Information Design aims to educate students so that they can represent their ideas by combining visual and audio information. The representation has not only digital media form but also traditional art form, for example painting and photographs. Therefore, students in this department must have both artistic and computing skills to create and author their works. The computing skills that are required for creating multimedia works are as follows:

1 The skills to use computer and network
2 The understanding of basic computer science
3 The skills to use software packages to create multimedia works
4 The programming skills

Software packages to create multimedia works are so powerful that students could easily create and author a variety of media. As students learn how to use tools, they acquire ways to represent their idea. It might be adequate if students are only required to create something as an operator in the future. However, they are inclined to use only the methods that the tools provide. They do not even notice that they are confined by tool's limitations.

Programming skills liberate students from this limitation. Students can change their mind from using only functions provided by tools to finding new ways of representation. It will improve their works even when they use software packages: it stimulates their imagination and enhances their general ability to create multimedia works. In creating interactive works, whether they have programming skills or not make a difference.

Programming skills for students of information design are similar to those for computer science. But the goal of the course is different in design discipline and in computer science, as King and Barr[1] pointed out. This paper proposes course materials to teach programming to students of information design by Java.

Role of programming
One of the major objectives of teaching programming is to develop the ability to think algorithmically. Students could experiment to decompose problems, to find solutions of each of them, and to organize them by programming. Writing programs is the best way to learn these processes and develop solutions, not only for computer science majors but also for non-majors.

Non-majors in computer science, who usually use computers as a tool, do not intend to learn quite high-level of programming skills, compared with majors in computer science. Our concern is students majoring in information design. They would be situated in the middle of the two. They are required to learn the usage of

software packages for creating multimedia works, and in addition, they need the programming skills. Even for using software packages, basic understanding of programming concepts is essential in writing a script to expand a function of a multimedia tool. Additionally, practical skills and knowledge of programming are necessary for interactive and network programming. Therefore, we have two phases of the course, the conceptual phase and the practical phase.

- Conceptual phase
  The goal of this phase is to understand programming concepts like data types, control structures, array, modularization in addition to thinking. In this phase, the object-oriented paradigm is essential, since many authoring tools for creating multimedia works have adapted it. Lingo, which is a script language for Micromedia Director, is an example. If students do not understand the object-oriented paradigm, an application using Lingo would be mess. For example, when one creates an animation in which a mouth and eyes move in a face, if a mouth and each eye is defined as an independent object, flexibility is increased and an script can be written simply.
- Practical phase
  This phase prepares students to write their own program for an actual use on top of conceptual understanding. Knowledge and techniques for interactive and network programming should be learned. Students who use only a script language of an authoring tool may not be motivated to this phase. However it is strongly recommended to achieve this phase if one feels limitations of tools and wants to enlarge the possibility of new art works.

Which language is used?

Many of the programming courses specially designed for "multimedia students" use scripting languages, such as HyperTalk, Lingo(Director)[1] and JavaScript[2]. The reasons for this are that most students majoring in fields related to multimedia will not become engineers, and they therefore need only conceptual understanding of programming. However, we would like to emphasize that "multimedia students" need to learn programming with a "real" programming language. A "real" programming language means one which has enough functions to develop a program which works interactively and over the Internet.

A computer is not just a high-qualified drawing tool or authoring tool. It is a tool which one can create anything one wants. This statement may be too extreme. However, students should not be confined inside functionality provided by existing tools. They should find a possibility of creating new art representations with interactive and networking programming

Which language do we use to teach programming? We selected Java which meets these requirements. Java is an object-oriented language with rich libraries and becoming a popular language especially over the Internet. Additionally, one can write an application executed in web clients and recently an environment for developing application in web servers has matured.

Approach to teach Java programming

The approaches and courses to teach Java for CS1 have been proposed[3,4,5,6]. We reexamined these approaches from an information design point of view. As mentioned above, we have two phases of the goals. In order to achieve these goals, we took the following two policies.

- Teach the object-oriented paradigm at an early stage of a course
  It is recommended in CS1[4] that the concept of classes and objects should be taught first. This is also important for information design. Understanding the object-oriented paradigm at the beginning has a great effect to the total process of learning Java programming and also helps students to use a script language based on it.
- Teach graphics and interactive programming without a simplified package
  The main objective of the practical phase is to extend their potentiality for creating high-quality and original application. This phase provides graphics, interactive, and network programming that would help students

find new ways of representation in their design works. Pedagogical packages have been proposed to make API simple and to hide complexity to use AWT and Swing[3, 4]. Since grammatical matter of Java and usage of library classes are taught in the first phase of our course, simplified packages are not necessary.

Course Contents

The contents of this course consists of two parts(Table 1). The first part is to teach the object-oriented paradigm and the second part is to teach graphics and interactive programming. They correspond to the conceptual phases and the practical phase, respectively, which we mentioned in the section of the roles of programming. We published a textbook which consists of these two parts[7].

Table 1 Course contents

phases	Subjects	Materials
object-oriented paradigm	objects method variables control flow statements array defining classes inheritance overriding interface thread	Turtle Graphics
graphical and interactive programming	GUI writing graphics handling events applets reading/writing files networking	Java Packages awt swing io net

In the first part, we considered it important to present object-oriented concepts one by one through concrete programming examples. At the same time examples and exercises at each stage should be interesting in themselves, so that a student who have no experience in programming can start. In particular, we considered that the course should start with creating objects and sending messages instead of displaying “Hello World” on the screen. For this purpose, we prepared a library on which the whole first phase is based. It is a turtle graphics library, which consists of Turtle class and TurtleFrame class(Table 2) . A Turtle object has functions of Logo Turtle as its methods and a TurtleFrame object is a frame on which a turtle moves and draw lines. This library was produced by one of the authors based on Hagino's Java Turtle Graphics Program

The course starts with the example program listed in Figure1, through which object creation and message passing are taught. Since objects are displayed graphically, a student can visually understand notions such as object identity, control flow statements, array, multi-threading, and so on. Since the API documents in Table2 are presented with this program, he or she can enjoy controlling a turtle through programming. Just after this, the next example program treats how to access API documents of the Java standard classes, taking java.awt.Color used as the argument type of setColor() method as an example. We consider that referring to the API documents is important and should be taught form the beginning.

Table 2　API of TutleFame and Tutle class

TurtleFrame	Constructor	TurtleFrame()　Create TurtleFrame with the default size(400x400) TurtleFrame(int width, int height)　Create TurtleFrame with given size.
	Method	void add(Turtle t)　Add t to this frame. void remove(Turtle t)　Remove t from this frame. void clear()　Clear all drawing. void addmesh()　Draw grid lines.
Turtle	Constructor	Turtle()　Create Turtle at (200,200) and 0 degree(A turtle heads top). Turtle(int x, int y, int angle) Create Turtle at the given position(x, y) and orientation(angle).
	Method	void fd(int n)　Go forward n. void bk(int n)　Go backward n. void rt(int n)　Turn right n degrees. void lt(int n)　Turn left n degrees. void up()　Put a pen up. void down()　Put a pen down. boolean isDown()　Return true if a pen is down, otherwise return false. int moveTo(int x,int y, int angle) Move to (x, y) and turn to angle, and return a distance of movement. int moveTo(Turtle t) Move to the same position as t, and return a distance of movement. void setColor(java.awt.Color nc)　Set the color of the pen. int getX()　Return X coordination. int getY()　Return Y coordination. int getAngle()　Return theangle. void speed(int x) Set speed of movement. Smaller x is faster(the default is 20). static void speedAll(int x) Set speed of movement for all Turtle objects.
	Field	double kameScale　Size of the turtle. The initial value is 0.4. java.awt.Color kameColor　The color of the turtle. The initial value is green. static boolean withKameAll If true, draw lines with turtles. Otherwise draw only lines. The initial value is true.

Figure 1 The first sample program

```
public class Example1{
   public static void main(String args[]){
      TurtleFrame f;
      f = new TurtleFrame();
      Turtle m = new Turtle();
      Turtle m1 = new Turtle();
      f.add(m);
      f.add(m1);
      m.fd(100);
      m.rt(90);
      m1.fd(150);
   }
}
```

After teaching how to use a pre-defined class in this way, we treat class definition. We introduce this subject by extending the Turtle class as in Figure2, that is an example program which draws five hexagons by defining a MyTurtle. Other object-oriented concepts such as overriding, dynamic binding, interface, and multi-threading are also treated through Turtle programs. Note that students need not involve themselves in details of the standard class libraries like AWT and Swing in this phase.

Figure 2 A sample program creating a subclass

```
public class MyTurtle extends Turtle{
    public void polygon(int n, int s){
        int   a=360/n;
        for(int j=0; j<6; j++){
           fd(s);
           rt(a);
        }
    }
}

public class Example{
    public static void main(String args[]){
        TurtleFrame f = new TurtleFrame(600,300);
        MyTurtle []hm = new MyTurtle[5];
        for(int i=0 ; i<5; i++){
            hm[i] = new MyTurtle(i*50+25, 150, 0);
            f.add(hm[i]);
            hm[I].polygon(6, 10);
        }
    }
}
```

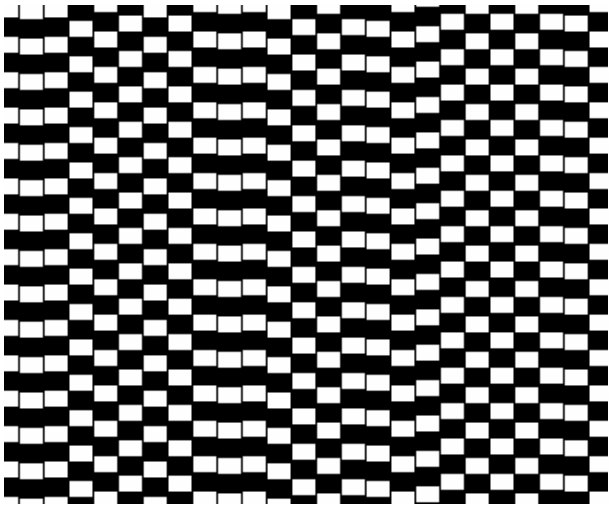
The second part is to teach graphics and interactive programming using Swing and AWT packages. In the first part, students have already got used to changing the visual response by coding. From our experience, students can easily start with naked Swing and AWT.

This phase instills programming in students and stimulates their multimedia representation through drawing graphics and event handlings. A familiar way for students to create an illustration or animation is the drag and drop operation, before they learn programming. Ideas that they have in their mind are implemented and visualized by moving a mouse and clicking on a drawing tool or an authoring tool. It is quite a straight process, ideas cross their minds, and then they click. In case of drawing graphics by a program, it is necessary to think how to direct a computer by coding instead of moving their hands.

At this stage, we adapt animation with geometrical movements and fractal graphics with interaction as examples. Animation is good to understand coordination, and fractal graphics are good to teach recursive process. Figure3 shows an graphical output of an animation in which small rectangles move downward. This animation is an example of visual illusions, in which vertical lines look to lean. When a user clicks on a column, a speed of movement on its column is changed. This example includes drawing graphics by indicating coordination, multi-threading, and interacting with a user. Additionally, example programs like this give students hints to represent their artistic image.

Network programming is also important to explore possibilities for finding new ways to communicate on multimedia works. We teach a simple server-client program with Java standard libraries.

Figure 3 An example of visual illusion



Summary

Important meanings exist in teaching programming to students of information design. Drawing graphics and interacting with a user by writing a program stimulates students to find and create new ways of representing multimedia works. We developed a programming course by Java for students of information design. The course's objectives are divided into two phases, the conceptual phase and the practical phase. Materials are proposed according to each phase.

One of the authors has used this approach for the past two years. It keeps motivation and sets clear objectives for students to learn programming. Students understand object-oriented concepts at an early stage of a semester, after struggling for 2,3 weeks. According to a turtle's movement, they enjoy coding from the beginning. After understanding the concept of programming and the basic programming techniques, students learn graphical and interactive programming with Java standard libraries.

Our effort to instill programming in students of information design brought certain results. Students could use script languages more easily than they did without programming knowledge. Furthermore, some of them created interactive programs over the Internet as their multimedia works.

Acknowledgements

The authors would like to thank Tatsuya Hagino at Keio University for providing his Turtle Graphics Program, on which our library is based.

Reference

[1] L.A. Smith King, John Barr, Computer Science for the Artist, SIGCSE Bulletin, Vol 29, no. 1, page 150, 1997

[2] Robert Ward, Martin Smith, JavaScript as a First Programming Language for Multimedia Students, Proceedings of the 6th annual conference on the teaching of computing/3rd annual conference on integrating technology into computer science education on Changing the delivery of computer science education, page 249, 1998

[3] Elliot Koffman, Ursula Wolz, CS1 Using Java Language Features Gently, Proceedings of the 4th annual SIGCE/SIGCUE on innovation technology in computer science education, page 40, 1999

[4] Michael Kolling, John Rosenberg, Tools and Techniques for Teaching Objects First In a Java Course Proceedings of the thirty SIGCSE technical symposium on Computer science education, page 368, 1999

[5] Stuart Reges, Conservatively Radical Java in CS1 Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, page 85, 2000

[6] Stephen Schaub, Teaching Java with Graphics in CS1, SIGCSE Bulletin, Vol 32, no. 2, page 71, 2000
[7] Hideki Tsuiki, Taeko Ariga, Java Programming for everyone(written in Japanese), Kyoritsu Publish, 2000