USE OF ANIMATION DIRECTOR MOVIES TO TEACH CS1

PROGRAMMING CONCEPTS

Herve Franceschi
Capitol College
Laurel, Maryland

ABSTRACT

We have developed a set of multimedia, Director-based movies to enhance and complement the teaching of CS1 using Java. From the beginning, we have decided to privilege simplicity and ease of use. There is one movie per concept and the movie is a simple, interactive, easy to use tutorial focused on a single concept, as opposed to a complex, multi-use, flexible tool. There is essentially no learning curve to use the movie. In this paper, two examples using Director are presented illustrating two typical programming concepts: looping and recursion. Benefits of using such an approach are discussed. A very brief overview of Macromedia Director is presented: Director is a software program to make movies, or dynamic animations, independent or part of a web page.

INTRODUCTION/ BACKGROUND

Over the last ten years, a lot of research has been done on the use of animation and visualization techniques for data structures and algorithms. Most of these are targeted at CS2 topics such as recursion, linked lists, other data structures, or sorting and searching. Typically, they are implemented as Java applets, are interactive, and let the user control the animation; most of them are designed to address a significant number of concepts with the same tool, possibly making the tool harder to use for the lower level students.

Our approach is different: we are focusing on CS1 using the Java programming language and we are using Shockwave movies designed with Macromedia Director. Our objective is to provide a movie illustrating the main topics of each chapter in the CS1
curriculum, in order to help the students learn. Our target audience is the CS1 student. We originally wanted to capitalize on the tweening animation capabilities of Director and add an element of fun in the movies. We decided instead to focus on code animation and execution, in order to avoid any potential distraction from the concept at hand. We kept the framework simple, with a menu of the chapter's main concepts, and step by step, user-controlled animation of the code with visual feedback on the values of the various variables used in the code. Thus, the focus is more at a low level rather than at a high level of abstraction. We feel that for CS1, a lower level of abstraction is appropriate.

MULTIMEDIA AND MACROMEDIA DIRECTOR

Director is one of the 2 industry standards, along with Macromedia Flash, for making movies: these movies can include animation and dynamic data manipulation. They can be converted to an executable program, or can be part of a web page. While Director is a fairly expensive development software program, the Shockwave Movie Player, the reader program for Director movies on the web, is available free of charge and easy to download from the Macromedia Internet web site.

Director is based on the metaphor of theater production: its basic components are the stage, where the action takes place, the score, which represents the timeline of the play or movie, and the cast, a user-defined collection of cast members, or actors. Cast members can be of various types: text, graphic images, animated gifs, sound, buttons, ...; cast members are reusable all along the movie. Furthermore, Director includes a scripting language called Lingo, which we will best described here as a programming language for non-programmers. Lingo features include basic programming tools such as local and global variables, arrays, looping, and functions.

ORIGINATION OF THE IDEA, AND HOW THE MOVIES WORK

Teaching both multimedia and programming classes has led to the simple idea of using multimedia as a tool to complement the traditional way of teaching programming.

The worldwide web, graphical animations, and user interactivity, have become standard features in the 21st century's student environment. These tools and features can be used independently or as a complement to more traditional teaching techniques. Whereas the main vehicle for teaching computer science is still the classroom, the use of interactive and dynamic movies can be appealing to web-savvy students and visual learners. Furthermore, they can be used not only in the classroom but also as tutorials outside the classroom.

An important feature of the movies is that there are very easy to use for both the instructor and the students: there is virtually no learning curve in order to use them. By keeping the movies easy to use, we wanted to maximize the chances of instructors and students actually using the movies.

In designing visualization tools, there is a tradeoff between how much flexibility is given to the user and how easy it is to use the tool. More complex tools with steep learning curves discourage their use, which defeats the purpose. The flexibility offered in the movies is sufficient to illustrate the concept at hand; for instance, when working
with arrays, the number of elements in the array and the array values are typically hard
coded. When retrieving an array element, the value of the array index is a user input.
When searching for a value inside an array, that value to search for is also a user input.
The user can then test several of these values to illustrate how the code works in various
cases. This combines easiness to use, and the flexibility required to explain the concept
at hand.

The movies start with a menu offering a choice of 2 to 6 topics: for instance, the
movie on looping will offer the user the following choices: for loop, while loop using a
counter, while loop using a sentinel value, while loop using a condition, do . . while loop
using either a sentinel or a condition. Once the user chooses a topic, a code sequence is
typically presented illustrating the topic at hand. The movie provides step by step
animation of the Java code, with the animation and feedback controlled by the user via
a "Next Step" button. A "Back to Menu" button is also provided as a way for the user to
move from one code sequence to another quickly.

The typical movie is dynamic and let the user enter the "main" parameter of the Java
code sequence, typically an integer; some of the variables are hard coded in the interest
of keeping the use of the movie simple and efficient for the user. However, the movie is
sufficiently dynamic to let the user test all possible cases, for instance get inside a loop
or exit it, or test the general case of a recursive method and the base case as well. The
"Next Step" button enables the user to move from one line of code to the next at his or
her own pace; each time the user clicks the button, the variables in play are updated to
show the user the effect of the statement just executed.

Colors are used to help the student follow the code and its execution: most of the
code is shown in gray, and the current statement is shown in red. Variable values and
outputs are shown in blue. Again, the focus is on simplicity, easiness to use, and low
level explanations.

ILLUSTRATING LOOPING TECHNIQUES WITH A STEP BY STEP MOVIE

Our first example illustrates the concept of looping in a step by step animation. In
a counting for or while loop, the user is asked to input a number, and the movie will step
through the code one line at a time and display the values of the various variables as it
moves from one line of code to another.

Here are three screenshots showing how the animation is rendered.

**Figure 1:** The user was earlier prompted for a number and entered 4 as input (the for
loop header is generated by the movie dynamically); this code sequence computes the
sum of the integers from 1 to the value input by the user, here 4. This screen picks up the
code execution after a number of steps; the statement i++ is executed and shown in red;
the value of the variable i is incremented from 2 to 3. The movie keeps track of the value
of the variables i and sum, as well as the value of the boolean expression ( i < 4 ).
Figure 2: The statement $i < 4$ is executed and shown in red; the value of the boolean expression $i < 4$ is shown in blue (it is true here).

Figure 3: After the condition to enter the body of loop ($i < 4$) evaluated to false, the next statement executed is the first output statement, shown in red. Actual output (sum is 6) is shown in blue. The second output statement, shown in gray, will be executed next and shown in red when the user clicks on the "next Step" button; the corresponding output will then be shown in blue ($i$ is 4).
ILLUSTRATING RECURSION WITH A STEP BY STEP MOVIE

Our second example illustrates recursion with a step by step animation. This example shows the factorial function coded recursively.

```java
for loop (using a counter)

Add all positive integers less than 4

// Code
int sum = 0;
int i = 0;
for (i = 1; i < 4; i++)
{
    sum += i;
}
System.out.println("sum is "+ sum);
System.out.println("i is "+ i);
```

```java
Recursion
Computing the factorial of a number

Scanner scan = new Scanner(System.in);
int a;
do {
    System.out.println("Enter an int from 0 to 9");
a = scan.nextInt();
} while(a < 0 || a > 9);
int result = factorial(a);
System.out.println("result is "+ result);
// More code here - Done for now
public static int factorial(int a)
{
    // The value of a is 4
    if(a == 0)
        return 1;
    else
        return (a * factorial(a - 1));
}
```
**Figure 4:** The movie is dynamic: the user previously entered 4 when prompted for an integer; the choice is limited to integers between 0 and 9, due to speed and space considerations. The factorial method has just been called with argument 4, the user input. Inside the method, the comment, updated dynamically, is shown in red and shows the value of the variable n inside the method, currently 4.

```java
public static int factorial(int n) {
    // The value of n is 4
    if (n == 0) return 1;
    else return n * factorial(n - 1);
}
```

**Figure 5:** The return statement is executed and shown in red. In blue, we show the returned expression (4 * factorial(3)) of the method, which includes a recursive call with argument 3.

```java
4 * factorial(3)
```
Figure 6: The first recursive call is made with argument 3; inside the method, we show the value of n, this time 3, in the comment in red, which is updated dynamically.

```java
Recursion
Computing the factorial of a number
Scanner scan = scanner.create(System.in);
int a;
do {
    System.out.println("Enter an int from 0 to 9");
a = scan.nextInt();
} while(a < 0 || a > 9);
int result = factorial(a);
System.out.println("result is " + result);
// More code here - Done for now
public static int factorial(int n)
{
    // The value of n is 0
    if(n == 0)
        return 1;
    else
        return (n * factorial(n - 1));
} 4*(3*2*1*1)
```

Figure 7: Several steps later, the code reenters the factorial method with argument 0. The comment is updated dynamically to reflect that, showing the value of the local variable n inside the method.
**Figure 8:** The base case has been reached with the call to factorial with argument 0. After the test for n equal to 0 evaluates to true, the method returns 1 and that statement is shown in red. The overall expression representing the returned expression from the original call to factorial is shown in blue. In it, factorial( 0 ) has been replaced by 1. At this point, the recursive calls have ended and the value 24 is computed and returned by the method. The output statement, shown in gray, will be executed next and shown in red (the output will be "result is 24", and will be shown in blue).

**CURRENT STATUS AND EARLY STUDENT FEEDBACK**

At this stage, the movie portfolio is still under development and is not 100% complete. As a result, it has not been tested extensively in the classroom yet. However, some of the movies, such as the ones on recursion and linked lists, have been presented to students in the classroom in non-CS1 classes. Students have expressed great interest and have responded very positively to that style of teaching. Students like the idea of using Director to illustrate programming concepts. A lot of students have been exposed, through a class or personal interest, to some type of multimedia software such as Flash or Director. Furthermore, they also like the step by step user control so that they can move through the code and its result at their own pace. Finally, they appreciate the availability of the movies as a permanent resource and self-paced tutorial outside the classroom.

**FINAL COMMENTS/CONCLUSION**

At this stage, although no quantitative data has been assembled, qualitative student feedback suggest that students would benefit significantly from the use of Director or Flash based movies as a complement to traditional teaching of CS topics such as programming. We also feel that the use of Director movies can be very efficient for other topics besides programming and computer science.

**REFERENCES**


