Introduction

Once computer displays became widely available in the 1970s and 1980s, computer scientists started pondering the pedagogical value of visual animations of algorithms. Now that computer displays are ubiquitous and computer science has become one of the core technical disciplines, the need for algorithmic animations is widely felt. Animations should help students learn new algorithms faster and with less effort, and should help programmers "see" their code running.

An algorithm is a set of well-defined steps, described using a formal language, which transforms input data into output data. Sometimes it is difficult to read the code used to implement a specific algorithm and test-runs can help to understand its mechanics. A bird's eye view of each step in an algorithm can improve learning and make it more enjoyable. It is one thing to read the sequence of instructions associated with, for example, Quicksort, and quite another to see the actual algorithm running, that is, to witness how successive computations gradually modify the order of the array of numbers. Humans are visual beings – psychologists know that a picture says more than thousand words [Larkin 87] and this remains true for animations: a good visualization of an algorithm can suddenly illuminate the whole subject.

However, producing good algorithmic animations is still more an art than a science [Hausner 01] and this is for good reasons. Superior algorithmic animations are based on a computational model of the problem to be solved, but also on a *cognitive model* of the intended audience. The best animations are those that match the level of problem abstraction with the level of knowledge and sophistication of the students [Gibson 94]. The best animation for a certain group of students (for example mathematicians) does not have to be the best for another group (for example physicists). This makes algorithmic animation actually a hard problem because it belongs to computer science but has to deal also with aspects of cognition, psychology and pedagogy [Petre 98].

This thesis deals with the general topic of algorithmic animation from different perspectives: first, we will go back to the basics, reviewing some older animation systems and the kind of graphical representations that they have used. We will see that separating the instrumentation of an algorithm from its rendering allows us to redirect the animation to different media, especially to an electronic chalkboard and to the Web.

In this thesis we study the classical chalkboard and explore what can be called good static explanations of algorithms, and how algorithms are "animated" by a lecturer using just a piece of chalk. These considerations can help us to distill the main principles behind good pedagogical expositions of algorithms. "Static animations" drawn with a piece of chalk are relevant for our digital era because new enlarged computer screens are making possible to use them as substitutes for the traditional chalkboard. The E-Chalk system, particularly, has been developed at the Free University of Berlin and provides an intelligent substitute for the blackboard. The screen can be used to start software agents in the background, making the blackboard come alive. The screen reacts immediately to the lecturers input and can, for example, perform a difficult computation (recognizing first the handwriting of the lecturer), or it can access a Web site and paste a picture of the latest weather report, or it can provide a translation of a word from German to Italian. The chalkboard of the future is not passive – it is a stage in which software agents act in the background as intelligent assistants for the lecturer. Good coordination and staging become as important as they are in a theater.

From this perspective -the teaching medium of the future- it is easier to understand the hierarchy of animation systems studied in this thesis. We go from very concrete and straightforward "static animations" of algorithms using some new features of E-Chalk to a whole library of primitives and data types that can be used to provide animations for the Web. Animations for the blackboard are produced using the Chalk Animator described in Chapters 4 and 5. For Web animations, I decided to use the Macromedia Flash animation engine, mainly because of its popularity and because it is able to export animations to other formats.[1] We will see later that the capability of exporting animations is very important to guarantee the longevity of a system. For our Flash rendering of animations, a new scripted language was defined. I gave it the name *Flashdance*, true to the algorithmic engineering tradition of naming animation systems after some kind of dance (as in the systems Tango, Samba, Pavane, etc.)

With this investigation of both the production of animations based on sketches (non-photorealistic rendering, as championed by Strothotte [97]) and high quality renderings for the Web, the thesis comes full circle. I start with the blackboard, but finish with the Web. Therefore, one of the messages of this thesis is that there is not a single unique way of "best" animating algorithms – there is a whole spectrum of alternatives that can be used by the lecturer. We consider some of them and provide examples of many algorithms which I have animated following different paradigms. This thesis is richly illustrated with screenshots of some of those animations.

[1] According to a survey of Internet users conducted by the NPD group for Macromedia, in June 2003, around 96% of Internet users in North America and Europe had installed the Flash 2 to Flash 5 players in their computers. This would make Flash more widely used than Java. See Section 6.1.

This thesis is organized as follows: Chapter 1 deals with the motivation for algorithmic animation systems and with reported results of its use in the classroom. Chapter 2 is a survey of algorithmic animation, from its humble beginnings with flow-charts up until the present days. This review will help us to extract some conclusions about the capabilities demanded of algorithmic animation systems by different communities. Chapter 3 contains a discussion about the general principles and standard views used by some animation systems and also about pedagogical guidelines to be followed when designing animations. Those guidelines focus our efforts in the subsequent chapters. Two hand-crafted productions were implemented to illustrate the general ideas. One deals with sorting algorithms, providing new visualizations thereof; the second is a whole unit for learning linked data structures in Java. Chapter 4 introduces the Chalk Animator, a system for the production of static or dynamic animations for the E-Chalk system. That chapter shows how to animate sketches of data using the macro facility of E-Chalk. The scripting language briefly introduced there, is the same used for the production of Flash animations. Chapter 5 shows how sketch animations can be produced using an interactive approach: the computer receives input data in handwritten form and the animation is then started. The handwritten input is animated by the computer using an animation protocol especially developed for E-Chalk. These sketch animations can be used directly for the Web. The result is a system which provides almost “on-the-fly” algorithmic animation. The chapter also shows that programming languages can be interpreted using handwritten code. A BASIC interpreter was especially written for the chapter. Chapter 6, finally, describes Flashdance a new scripting system for algorithmic animation that can be used to produce Flash animations. Flashdance provides a language for algorithmic animation which is imperative and reversible. Our scripting language lets the user move forward or backward in the animation without having to explicitly instrument reversibility in the code of the algorithm. Animations produced with Flashdance can make full use of the high-quality Flash animation engine. Chapter 7 provides the final conclusions, reviews my educational experiments with the system, and mentions future challenges. Chapter 8 is a summary of contributions. The list of references is comprehensive and includes the most important papers in the field, since the beginning of algorithmic animation in the 1960s, to the present – it should provide a useful source to the reader.