TOWARDS ANIMATING COMPUTER PROGRAMS: A FIRST PROGRESS REPORT

Ron Baecker
Assistant Professor
Computer Science
University of Toronto

ABSTRACT

Computer animation is the art of using the computer as a synthetic tool in the production of animated films. Striking successes have been achieved in visualizing complex dynamic phenomena of mathematics, physics, and chemistry. The paper describes the beginning stages of a major effort to animate the behavior of algorithmic processes. We are studying how best to represent computer processes with dynamic images, and attempting to test this understanding with short experimental film clips. We are also building tools to facilitate the creation of these images, and to enable efficient production of computer science teaching films.

RESUMÉ

L'animation par ordinateur est l'art de l'utilisation de cet ordinateur comme outil de synthèse dans la production de films animés. On a eu de "formidable" succès dans la visualisation de phénomènes cinétiques en mathématiques, en physique et en chimie. Dans cette communication, on décrit le commencement d'un effort majeur afin d'animer le fonctionnement des processus algorithmiques. Nous cherchons les meilleures méthodes pour représenter des processus de l'ordinateur à l'aide de dessins animés et nous essayons de vérifier ces représentations par des extraits courts de films expérimentaux. Nous construisons aussi les outils pour faciliter la création des images et pour permettre la production de films d'instruction scientifique par ordinateur.

Third NRC Man-Computer Communications Seminar -- 30 May 1973

TOWARDS ANIMATING COMPUTER PROGRAMS: A FIRST PROGRESS REPORT*

Ron Baecker
Assistant Professor
Computer Science
Univ. of Toronto

Since its inception, computer graphics has provided a tool for the description, analysis, and explication of the behavior of computer programs. Some investigators have experimented with graphic displays as program debugging aids.(1) Other have used them to monitor the behavior of computer systems.(2) Still others have built "flowchart compilers" or other graphical language processors which allow direct programming in a graphical language such as flowcharts.(3) Finally, the most successful and commercially useful efforts have produced systems capable of documenting existing programs with computer-drawn flowcharts.(4)

Concurrent with these developments there has emerged an art of computer animation.(5) Here the computer is used as a synthetic tool in the production of animated films. Striking successes have been achieved in visualizing dynamic phenomena of mathematics, science, and engineering.(6) The computer has proved particularly useful because of its ability to construct precise, mathematically determined images, because of its ability to simulate hypothetical worlds, because of its ability to expand or contract space and time, and because of its ability to portray complex spatial phenomena, particular those in three dimensions. Computer animators have appreciated the ability to "debug" a film that comes from the ability to redo it with incremental effort, and which also makes it economical to explore variations on a theme. Most importantly, computer animation has produced new animators, opening doors to many to whom the art had been inaccessible heretofore.

* This work was funded by grants from the National Research Council.

Why Animate Computer Programs?

While physicists, chemists, biologists, and mathematicians have seized upon this opportunity with great vigor(7), computer scientists have produced very little computer animation. In the 1960s, computer animation's first decade, the technique was used to produce only one major computer science teaching film.(8) There has also been no effort to develop systematic techniques for animating computer processes.

This is surprising and disappointing, since it seems natural to explicate the behavior of computer programs through animation because:

(1) Programs are inherently temporal, that is, they exist through time. We often describe a program's dynamic behavior with a hand simulation, manipulating symbolic information in the form of numbers and character strings. Animation can be regarded as the graphic simulation of a process, in which dynamics are expressed by moving images. The animator of a computer program must have flexible control of the temporal expansion from program time to movie time, since one second of the movie may represent microseconds or seconds of program time.

(2) The power of a single computation is magnified by repetition under controlled variation, either iterative or recursive. The "meaning" of a repetive calculation can often be induced from a sequence of examples. A motion picture is an economical technique for presenting a sequence of examples.

(3) Parallel, or quasi-parallel, activities occur in modern computations in an organized form. A motion picture can easily present several simultaneous activities. If this is done with structure and restraint, making use of principles of visual perception and cognition, it is an effective technique for displaying parallel computations. It can also be used to show simultaneously various "points of view".

(4) Structural relations, and relations of cause-and-effect, are central to the understanding of computer programs. Such relationships can be stated and explained explicitly... "A is connected to B"... "C and D share the common property F"... "F causes G". Alternatively, they may be introduced implicitly by showing them in action. If we could see programs in execution, we could induce structure and cause-and-effect from our observations.

On the other hand, if may be difficult to construct effective animation sequences because:

(1) Masses of detail exist in the description of many computer programs. To be effective, program animation must abstract or highlight the essential while leaving the detail "potentially accessible". It seems likely that the best visual explanation of computer programs will consist of integrated motion picture

sequences and still graphics. The former will communicate the structure; the latter, the detail.

(2) The understanding of the "meaning" of a computer program depends upon the comprehension of a variety of abstractions, for example, literal, variable, name, value, assignment, evaluation, binding, floating point number, string, array, index, pointer, procedure, procedure call, iteration, and recursion. Graphical conventions and means of representation must be developed for each such abstraction. These techniques must be meaningful to those who understand the abstractions. They must be suggestive to those who are still trying to comprehend the abstractions. They must also be aesthetically pleasing.

With these incentives and challenges in mind, we have launched a substantial effort in program animation, the use of computer animation to visualize the dynamic abstractions of computer science, and to explain the behavior of complex algorithmic processes. There are two major aspects to this work. We are studying how best to represent computer processes with dynamic images, and testing this understanding with short experimental film clips. We are also building tools to facilitate the creation of these images, and to enable efficient production of computer science teaching films. These two phases are mutually reinforcing and overlapping.

How Should Computer Programs be Animated?

Huggins has coined the phrase iconic communications to describe the art of constructing pedagogically effective and aesthetically pleasing visual images.(9) We have begun to study issues in the iconic communication of computer programs.

Our initial concern has been with the display of abstractions such as those listed above. This includes data, such as numbers, strings, arrays, and pointers, and processes, such as assignment, evaluation, iteration, and recursion. We are investigating how optimally to use elements of line, shape, texture, color, movement, and timing to portray and communicate these abstractions.

Here are two examples. Margie Alcorn is studying techniques for graphically representing a bubble sort. Stefan Sullivan is studying techniques for graphically representing a complex tournament replacement tape sort.

Our methology includes continual discussion and evaluation of sketches and film samples, and occasional empirical studies on "subjects" to validate our intuition.

Results of this work will be presented at the meeting. We may also show if time permits some previous examples of effective program animation: Jim Wheeler's portrayal of the switchyard algorithm for converting infix expressions into Polish postfix form; and Carolyn Dutky Romano's portrayal of the Warnock non-deterministic hidden line elimination algorithm.

What Tools Will Aid the Animation of Computer Programs?

How does one animate an algorithm? It is possible to animate a specific algorithm expressed in a standard programming language by augmenting it, in an ad-hoc manner, with calls to a display package. We currently are working in this way with BATCH ART, a set of FORTRAN and PL/I callable subroutines which generate and manipulate two-dimensional pictures and draw them on a line printer, a paper plotter, or a microfilm recorder.(10) The package includes routines that produce common geometric figures, that allow input and output of arbitrary images, that apply precise or randomly controlled geometric transformations to existing pictures, and that combine existing still pictures into new ones, or into image sequences, that is, movies. One of the most useful routines is a key frame animation program that generates "in-betweens" by interpolating images between selected key frames.

Our graduate students have animated numerous algorithms with BATCH ART. We find it adequate but awkward and unsatisfying. Images must be described with too much detail at a low level of abstraction. Tools for describing movement and dynamics are very limited. Parallel activities can be produced only with great difficulty. All aspects of a picture must be specified in each frame; one cannot establish global drawing commands to apply to all frames for all time.

Hence we would like to build some new tools to facilitate our work. It is possible to imagine, and even to build, a system which could animate any program in any language running on a specific machine. However, we believe that it is impossible to build such a system so that it would produce good animation.

We are steering an intermediate course between the two extremes of accepting the limitations of existing tools, or of attempting to build new ones of unrealistic or absurd generality. We are rather beginning to build a variety of powerful special-purpose tools, each suited to the animation of a particular class of program.

In designing these tools, we first asked ourselves the question: What parameters of computer programs most significantly affect their animation? These parameters include the application domain of a class of programs, for example, syntax analysis, graph theory, or matrix calculations; the language in which a program is expressed, for example, ALGOL, LISP, or 360 assembly code; the nature of the implementation, for example, compiled, interpreted, or conversational; and the environment for producing the movie, for example, off-line, interactive written, or interactive graphical.

We have begun work in an off-line environment on five special-purpose systems for animating classes of computer programs:

(1) Margie Alcorn is working on a system for the animation of searching, sorting and hashing;

(2) Larry Chan is working on a language for expressing graph-theoretical algorithms, and a system for animating them;

(3) Ricky Law is working on a language for writing and animating stochastic simulations;

(4) Kit-Mei Chan is working on a system for animating programs in mini-LOGO, focusing on the display of recursive string manipulations;

(5) Jim de Boer is working on a system for animating programs in micro-PL/I, focusing on the display of iterative numerical calculations.

In addition, Elinor Tezman is working on a program for the semi-automatic layout and display of trees and graphs, a program which will be used by the other systems. First results from these projects will be presented at the meeting.

Each system must provide the user with power, flexibility, and discrimination in specifying:

--The structure of interesting phenomena.
Which data (program states) and processes (program events) should be shown?

--Display formats and display organization.
How should these data and processes be displayed? What pictures and picture changes should be used?

--Observation points and conditions.
When, or under what conditions, should pictures be shown or changed?

--Modes of detail enhancement and suppression.
How can complexity be controlled in space and in time?

--Timing.
How is program time to be mapped into movie time?

Concluding Remarks

Our short term goal is to make it natural and easy to construct animation specifications, with "minimal" changes to the "original" program, for both quick-and-dirty, and polished, professional teaching films. We are interested in movies for the specific education of programmers and computer scientists, and for the general education of literate citizens. One long-range goal is to facilitate the development of integrated systems for graphically-mediated program preparation, editing, debugging, documentation, and explication. These systems will be laboratories for formulating, testing, and refining programs, and hypotheses about program behavior. Such laboratories will aid the increasingly difficult task of developing, maintaining, and controlling the large computer systems upon which so much of modern society depends.

References

(1) Stockham, Thomas G., "Some Methods of Graphical Debugging", Proceedings of the IBM Scientific Computing Symposium on Man-Machine Communication, Yorktown Heights, N.Y., 1965, pp.57-71.

Baecker, Ronald M., "Experiments in On-Line Graphical Debugging: The Interrogation of Complex Data Structures", Proceedings of the First Hawaii International Conference on the System Sciences, Honolulu, Hawaii, 1968.

(2) Grochow, Jerrold M., "Real-Time Graphic Display of Time-Sharing System Operating Characteristics", Proceedings of the 1969 Fall Joint Computer Conference, pp.379-386.

(3) Ohringer, Lee, "Flowcharts as a Computer Language", Scholarly Paper 37, Computer Science Department, Univ. of Maryland, College Park, Maryland, 1971.

(4) Abrams, Marshall D., "A Comparative Sampling of the Systems For Producing Computer-Drawn Flowcharts", Proceedings of the 1968 ACM National Conference, pp.743-750.

(5) Baecker, Ronald M., Interactive Computer Mediated-Animation, MIT Project MAC-TR-61, 1969.

Baecker, Ronald M., "From the Animated Student to the Animated Computer to the Animated Film to the Animated Student", Proceedings of the Purdue 1971 Symposium on Applications of Computers to Electrical Engineering Education, Lafayette, Indiana, pp.106-113.

Weiner, Donald D., "Computer Animation - An Exciting New Tool for Educators", IEEE Transactions on Education, Volume E-14, Number 4, November 1971, pp.202-209.

(6) King, Ellis, Producer, "Movies from Computers - An Interim Report", Color Sound film sampler, Education Development Center, 39 Chapel Street, Newton, Mass., U.S.A., 1967.

Knowlton, Kenneth C., "Computer-Animated Movies", from Emerging Concepts in Computer Graphics, Secrest and Nievergelt, Editors, W.A. Benjamin, 1968, pp.343-370.

Baecker, Ronald M., "Computer Animation: An Aid in Visualizing Complex Processes", Canadian Datasystems, Volume 5, Number 3, March 1973, pp.30-32.

(7) Wilson, Kent, Producer, "Patchwork 71", Color Sound film Sampler of Computer Animation in Chemistry, The Senses Bureau, Department of Chemistry, The University of California at San Diego, 1971.

(8) Knowlton, Kenneth C., "L6: Bell Telephone Laboratories Low-Level Linked List Language", 2 black-and-white sound films, Bell Telephone Laboratories, Murray Hill, N.J., 1966.

(9) Huggins, William H., "Iconic Communications", IEEE Transactions on Education, Volume E-14, Number 4, November 1971, pp.158-163.

(10) Mezei, Leslie and Zivian, Arthur, "ARTA, An Interactive Animation System", Proceedings of the 1971 International Federation of Information Processing Societies Conference, pp.429-434.

"BATCH ARTA Manual", Computer Systems Research Group, University of Toronto, 1973.