

University of Montana

## ScholarWorks at University of Montana

---

Graduate Student Theses, Dissertations, &  
Professional Papers

Graduate School

---

1996

### Task-centered user interface design of an algorithm animation program

Yolanda Jacobs Reimer  
*The University of Montana*

Follow this and additional works at: <https://scholarworks.umt.edu/etd>

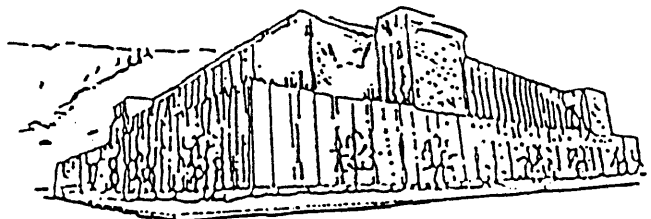
**Let us know how access to this document benefits you.**

---

#### Recommended Citation

Reimer, Yolanda Jacobs, "Task-centered user interface design of an algorithm animation program" (1996). *Graduate Student Theses, Dissertations, & Professional Papers*. 5120.  
<https://scholarworks.umt.edu/etd/5120>

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact [scholarworks@mso.umt.edu](mailto:scholarworks@mso.umt.edu).



Maureen and Mike  
**MANSFIELD LIBRARY**

The University of **MONTANA**

---

Permission is granted by the author to reproduce this material in its entirety, provided that this material is used for scholarly purposes and is properly cited in published works and reports.

*\*\* Please check "Yes" or "No" and provide signature \*\**

Yes, I grant permission

No, I do not grant permission

Author's Signature Yolanda J. Reimer

Date 7-25-96

Any copying for commercial purposes or financial gain may be undertaken only with the author's explicit consent.

TASK-CENTERED USER INTERFACE DESIGN

OF AN

ALGORITHM ANIMATION PROGRAM

by

Yolanda Jacobs Reimer

B.S. The Pennsylvania State University, 1989

presented in partial fulfillment of the requirements

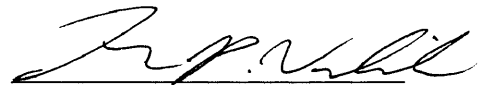
for the degree of

Master of Science

The University of Montana

1996

Approved by:



Chairperson



Dean, Graduate School

7-30-96

Date

UMI Number: EP40584

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI EP40584

Published by ProQuest LLC (2014). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code



ProQuest LLC.  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 - 1346

Task-Centered User Interface Design of an Algorithm Animation Program

Director: Nicholas P. Wilde 

Numerous algorithm animation systems designed and implemented in the last decade have been predicated on the belief that graphical animation tools will help students learn algorithms better than traditional methods alone. Many studies have been conducted to date, both informally and empirically, to test whether or not this initial belief is true. The results of these studies have been somewhat mixed. In general, existing algorithm animation systems have not proved as significantly effective in helping students to learn as was initially hypothesized. The question of how to make algorithm animation systems more effective as instructional tools continues to be studied today.

This research attempts to find more effective ways to create algorithm animation systems by using a task-centered user interface design approach to design and implement a prototypical algorithm animation program. Also incorporated into this prototype system are the results and conclusions derived from previous studies. Upon completion of the implementation of the algorithm animation program, user testing is conducted to see which elements of the design were most successful.

Although this research is informal and exploratory in nature, it nonetheless reaches important conclusions in the area of effectiveness of algorithm animation programs as instructional tools. Certain conclusions reached in this study are found to coincide with those of prior studies -- for example, all the users of the algorithm animation system in this study reported feeling excited and interested in using the program, a result that also occurred in previous studies. However, this study also reaches some new and important conclusions in its own right, such that algorithm animation systems are greatly enhanced when they are designed from the user's perspective and when they include visual and textual semantic redundancy.

# TABLE OF CONTENTS

1. Introduction
2. The Design Process
  - 2.1 *Task-Centered User Interface Design*
  - 2.2 *Choosing the Application Tool*
  - 2.3 *Choosing the Algorithm*
  - 2.4 *Design Details*
  - 2.5 *Common Features*
  - 2.6 *View 1 -- Visual / Overall Strategy (VO)*
  - 2.7 *View 2 -- Visual & Code / Overall Strategy (VCO)*
  - 2.8 *View 3 -- Code / Overall Strategy (CO)*
  - 2.9 *View 4 -- Visual & Code / Implementation Detail (VCI)*
  - 2.10 *View 5 -- Code / Implementation Detail (CI)*
3. Implementation
4. User Testing
  - 4.1 *Set-up*
  - 4.2 *Results*
  - 4.3 *Observations*
5. Conclusions
6. Future Work

## LIST OF TABLES

Table 1. Matrix of Procedural Understanding. p. 31

Table 2. Matrix of Conceptual Understanding. p. 32

## LIST OF ILLUSTRATIONS

Figure 1. Graham's scan algorithm pseudocode.	p. 11
Figure 2. Different views available to user.	p. 16
Figure 3. Navigational Buttons.	p. 18
Figure 4. View 1 -- Visual / Overall Strategy (VO).	p. 20
Figure 5. View 2 -- Visual & Code / Overall Strategy (VCO).	p. 22
Figure 6. View 3 -- Code / Overall Strategy (CO).	p. 23
Figure 7. View 4 -- Visual & Code / Implementation Detail (VCI).	p. 24
Figure 8. View 5 -- Code / Implementation Detail (CI).	p. 25



## 1. Introduction

Tango [25], Balsa [8], Zeus [5], and AACE [14] are just a few of the algorithm animation systems that have been created to date. The conceptual belief behind such systems is that students will learn algorithms faster and more thoroughly using graphical animation tools than using more traditional learning methods. Proponents, seeking to bring students beyond the passivity of learning through lecture alone, laud the active engagement and dynamic nature of these systems. Unknown at the time these systems were built, however, and still an open question, is the extent to which these animation programs assist the learning process, and the optimal ways to use them in conjunction with the more traditional methods of teaching. In their paper entitled *Do Algorithm Animations Assist Learning? An Empirical Study and Analysis* [23], Stasko, Badre and Lewis discuss the widely held belief that animations should help in the learning process, but then add that "...the viability of algorithm animations as instructional aids remains rooted in intuition. No substantive empirical evidence has ever been presented to support these claims." Also cited in this paper are several informal studies that have been conducted in an effort to explore the value of animations as instructional tools. These studies reveal a variety of conclusions, including that stronger students seem to benefit more than weaker students from using algorithm animations [28], and that any immediate advantages gained by students who have learned using animations diminish with the passage of time [21]. The empirical study conducted by Stasko et al. and described in their paper, involved two groups of student subjects: one was given only a textual description of the pairing heap algorithm, while the second group was presented with both an algorithm animation system (XTango) demonstrating pairing heaps, and the same textual description given to the first group. The results of this study affirmed that although the group that had access to XTango in addition to the text scored slightly higher on post-

tests than the alternate group, and completed the post-tests slightly faster, the competency gap was not as significant as had been expected. Another important result of this empirical study, as well as of many other informal studies, is that in each instance the students using animation tools reported feeling interested and excited about using them. While such benefits are perhaps non-quantifiable, they are no less important. In concluding this article, Stasko, et al. list some key findings that they feel should be incorporated in future animation systems, such as rewind/replay capabilities, and warn that their results "suggest that any general virtues that visual, animated presentations may possess are not powerful enough to produce good performance unless the presentations are keyed to specific learner needs."

In a more recent empirical study, Stasko, this time in conjunction with Lawrence and Badre [24], examined students who learned Kruskal's Minimum Spanning Tree algorithm using a variety of methods. This experiment studied the effects of students learning the algorithm through lecture, or through a lecture and lab combination. The lecture portion of this experiment was divided into two sub-groups: one with a teacher lecturing on the algorithm and using in-class examples generated by the Polka animation system, and another with a teacher lecturing on the algorithm and presenting in-class examples with overhead transparencies. The lecture/lab combination group was further sub-divided into the two categories of passive lab and active lab. In the passive lab, the students viewed pre-defined scripts of an animation illustrating the algorithm, while in the active lab, the students were able to create and view their own animation examples of the algorithm. The results of this study were that the students in the lecture/lab combination group showed an advantage over the students in the lecture group, and that those students using the active lab with user generated examples benefited the most.

In their as yet unpublished paper *Testing Effectiveness of Algorithm Animation*, Gurka and Citrin [18] look at some of the factors they believe are responsible for the ineffectiveness of past algorithm animation systems. One of the issues they present is that many of the existing algorithm animation systems rely too much on the visual, and often lack what they refer to as "semantic redundancy", the displaying of additional information, such as text, in an animation along with the visualization. Gurka and Citrin believe that the inclusion of semantic redundancy in an algorithm animation program will add to its overall effectiveness.

So while we are perhaps closer to understanding some of the effects and issues surrounding algorithm animation programs as instructional aids, the question of how to make these programs as effective as possible remains. The nature of the study described here, revolves around conducting exploratory research into the task-centered user interface (UI) design, implementation, and user testing of an algorithm animation program. The task-centered UI design process focuses on the statement by Stasko et al. quoted above: that an effective algorithm animation program must be "keyed to specific learner needs." This research also strives to incorporate some of the other techniques and hypotheses aimed at making an algorithm animation program more effective, such as rewind and replay capabilities, and semantic redundancy. The goal is to help clarify how algorithm animations may affect an individual's learning, and to work towards making algorithm animation systems as useful as possible in assisting the learning process.

Formal and empirical studies can be extremely difficult to accomplish, especially in a relatively limited time. This study admittedly lacks the necessary breadth and formality of data to make conclusive claims about algorithm animations as instructional tools. Additionally, the numerous factors that can bias such a study, and therefore must be

controlled, such as prior knowledge of the students/users, realistic post-testing conditions, representative testing body, statistically relevant numbers of testers, etc., are beyond the scope of this research. As exploratory research, this study seeks to gather further insight into this area of research, and perhaps to lay a foundation for future research in this area.

## **2. The Design Process**

### ***2.1 Task-Centered User Interface Design***

The user interface design methodology I am ascribing to in this research project is a modified version of task-centered user interface design, as described by Lewis and Rieman in [19]. The task-centered UI design methodology stresses that the key to the development of any good interface is user participation throughout the entire system design and development process, from inception to system testing and beyond. Since the goal of my algorithm animation system design was to design a program that proves most effective from the user point of view, this methodology was the most advantageous.

Lewis and Rieman cite the following key points to effective task-centered UI design:

1. Determine who the system users will be.
2. Address the specific tasks users expect to accomplish in using the system, developing system functionality in consultation with the user's actual needs.
3. Intelligently borrow from existing interfaces and interaction designs for the to-be-designed system.
4. Design the system by balancing user needs with system limitations.
5. Assess design usability early before the prototype implementation is too far along.
6. Test prototypes with the users.
7. Iterate parts of the process repeatedly until the optimal blend of user satisfaction and system functionality is achieved.

While addressing the user-related question raised by the first two of Lewis and Rieman's key points, I did not have the advantage of consulting with potential users. Although I had a general idea who my initial users would be, the group nevertheless remained uncertain, as did the users who might make use of the algorithm animation system in the future. Consequently, I was forced to rely on my own experience and research in designing user interfaces, as well as on the advice of my thesis advisor, in order to assemble a rough user assessment.

Mindful of Lewis and Rieman's third key point, I was able not only to borrow ideas from some of the better user interfaces I had experience with, but also to benefit from the growing number of articles covering this area of research. In *AACE - Algorithm Animation for Computer Science Education* [14], for example, Peter Gloor discusses his own experience in developing an algorithm animation system for educational purposes. In the article, Gloor cites numerous factors he deems critical to the design of an effective user interface component to such a system. Not only must the system be clear and concise, Gloor explains, but it must also have a historic component to it, must emphasize the important steps, retain the user's interest, and be interactive. The system described in this thesis was prototyped using Apple's HyperCard program -- a valuable resource for working with HyperCard is Apple Corporation's *HyperCard Stack Design Guidelines* [1]. This book focuses on techniques for developing good stacks in HyperCard, while also highlighting important general user interface issues to be mindful of while designing. Perhaps most important to my project, though, were the results of other algorithm animation systems research studies. One of the conclusions reached by the earlier Stasko et al. experiment [23] was that a good animation system should have rewind and/or replay capabilities, providing some sort of historical view of the algorithm animation. I kept all

of these points and results in mind when designing my algorithm animation program, and incorporated them where applicable and when possible.

Key point 4 of the design methodology was not an issue for this design and implementation project for two main reasons. First of all, I knew from the outset that my algorithm animation program would be relatively minor in scope: I only wanted to test one algorithm for this initial research, and early benchmarks of the both the size of my algorithm animation program and the speed at which I wanted it to run showed no problems. Thus I was confident of having sufficient machine resources to run my animation adequately without memory or processing speed becoming a major concern.

The algorithm animation program developed for this study can be viewed ultimately as being a prototype for the development of future algorithm animation systems. Upon completion of a skeletal version of the animation program, where enough of the program was implemented to illustrate the intended design direction, I discussed and reviewed this partial implementation with my advisor before continuing. While such a consultation may seem to be somewhat trivial as it did not include potential future users of the program, it proved to be useful in solidifying the direction of my design as well as in rectifying minor misdirections before the more substantial implementation of the program. Thus, key point 5 of the task-centered UI design was completed.

As for key point 6 of the methodology, after the design and implementation of the system was complete, I conducted user testing sessions, the preparation and results of which are described in section 7 of this paper. I was unable, however, to incorporate the iterative nature of key point 7 in this study, a step that would make for possible future work on this animation program.

## ***2.2 Choosing the Application Tool***

One of the first decisions I had to make before I could design my animation was what application tool I would use. Available resources did not limit me in any way, as I had access to a Unix network of RS/6000 machines, IBM compatible PCs and Macintosh PCs, all with a wide variety of software installed. I briefly considered creating my program using C, X/Motif or HyperStudio before settling on the Macintosh HyperCard application (version 2.3) with HyperTalk scripting language. Although I had programming experience with both C and X/Motif and not with HyperCard/HyperTalk at the time I made this decision, I was fairly convinced from the outset that to chose C or X/Motif would mean a longer implementation time with consequently less time devoted to actual design issues, something I expressly wanted to avoid. I was aware from the outset that a project like this could easily get side-tracked into becoming nothing more than a major programming project of yet another algorithm animation system. To counter this, my plan was to focus first on the task-centered UI design process, and then on the user testing and feedback stage, creating just enough of a program implementation in between to allow me to complete these two main tasks satisfactorily. From my preliminary investigation of the capabilities of HyperCard, I was persuaded that this application tool would be suitable for creating the major functionality and features that I wanted to include in my animation. I could only hope that any shortcomings I might find with the HyperCard application as I learned more about it would create only minor deviations of my animation program from the initial design.

Though initially I lacked specific HyperCard knowledge, I was generally familiar with the Macintosh environment, as well as with many of its application tools, making me confident that I would pick up the specifics of HyperCard and HyperTalk with relative

ease. In addition, I knew from experience that Apple's documentation was usually both abundant and well-written. For materials that I relied on in learning this tool, see [15][29].

Since user testing of my animation was to be a major part of my study, the final component in my decision to use HyperCard was the necessity of an installation location that would allow me to conduct my user testing privately. My access to C and X/Motif would have meant conducting testing in a public lab with up to a dozen other computer users; with the Macintosh and HyperCard application, however, I had access to a private office, one more than adequate for holding my user testing sessions. Additionally, I had to verify that the Macintosh PC on which I would be doing not only my user testing, but also the development of the animation program, had sufficient memory to hold my program, and was fast enough to run the animation smoothly and without any obvious time delays. Both of these factors were tested early in the development stage, and were found to be sufficient.

### ***2.3 Choosing the Algorithm***

Essential in choosing the algorithm to use in my animation was that the algorithm be one which my eventual users were not previously familiar with. Even though my research was mostly informal, one of my primary focuses was the user testing and feedback session. To be as accurate and unbiased as possible, I needed an algorithm that was new to all of my users, especially since only a limited number of users were available for testing. Since I was planning to use volunteers from the CS332 Algorithms course, I checked with the instructor to determine what types of algorithms he would and would



not be covering throughout the semester. Beyond this knowledge, I had to hope that my users did not have other prior exposure to the algorithm I would choose.

In addition to picking an algorithm that I believed would be new to my users, I also wanted one that was of medium complexity, and one fertile enough for animating in a variety of ways during my task-centered UI design. I also needed an algorithm that I was confident could be thoroughly completed in the given time frame, again realizing that the focus of my research was not to be the implementation, but the task-centered UI design and user testing process. I was concerned that if my algorithm were too simple, it would prove futile in trying to test the reception of key parts of the algorithm using various tactics. Conversely, if I chose something too involved, I might only have been able to achieve fragmented and incomplete results, again creating difficulties when trying to reach conclusions from the overall effort.

Keeping all of these factors in mind, I settled on Graham's scan version of the convex hull algorithm [9][22]. The convex hull algorithm falls under the category of computational geometry, an area I believed people were generally less familiar with than, for example, trees or heaps. The convex hull algorithm determines the smallest convex polygon of a given set of points such that all points end up either a vertex of the polygon, on one of the edges of the polygon, or in the interior of the polygon [9]. A common way of envisioning the concept of the convex hull algorithm is to picture the set of points as being nails sticking out of a board, and the smallest convex polygon (convex hull) of those points being formed by wrapping a tight rubber-band around the outside of all the nails. The convex hull algorithm is also used as an initial step in the computation of other geometric algorithms, such as the two-dimensional farthest-pair problem.

The convex hull algorithm has multiple methods of implementation: Graham's scan, Jarvis's march, the incremental method, the divide-and-conquer method, and the prune-and-search method, see [9] for further details of these methods. Because it has multiple interesting parts to illustrate, I chose to implement Graham's scan method, again in the interests of making the task-centered UI design a challenging process. See [16] for the original version of Graham's scan as given by Graham.

Graham's scan runs in  $O(n \lg n)$  time, and uses a method known as *rotational sweep* to process points in the order of the polar angle they form with a reference point. The process by which Graham's scan finds the convex hull of a set of points is first to label and order all the points in the set. This is done by establishing a base point,  $p_0$ , which is the lowest point on the y-axis (and leftmost point on the x-axis in case of multiple points having same minimum y values). The rest of the points are then labeled in order of least polar angle to greatest polar angle relative to  $p_0$ . Once the points have been labeled and ordered, the stack is initialized. From here, three points are compared (top of stack, next-to-top of stack, and next point in ordered list) to see if the angle they create forms a "left turn" or not. If they do form a left turn, the third point is pushed onto the stack; if not, the stack pops a point and continues analyzing the angles formed by the next three points. This continues for all points in the set, eventually yielding the stack that holds only the points in the smallest convex polygon. Figure 1 lists the pseudocode of Graham's scan convex hull algorithm used in my animation program. See [9] for more information on the details of the algorithm.

```

GRAHAM-SCAN(Q)
1  let  $p_0$  be the point in  $Q$  with the minimum
   y-coordinate, or the leftmost such point in
   case of a tie
2  let  $\langle p_1, p_2, \dots, p_m \rangle$  be the remaining points
   in  $Q$ , sorted by polar angle in counterclockwise
   order around  $p_0$ , (if more than one point has
   same angle, remove all but the one that is
   farthest from  $p_0$ )
3  top[S]  $\leftarrow$  0
4  PUSH( $p_0, S$ )
5  PUSH( $p_1, S$ )
6  PUSH( $p_2, S$ )
7  for  $i \leftarrow 3$  to  $m$ 
8      do while the angle formed by points
         NEXT-TO-TOP(S), TOP(S), and  $p_i$ 
         makes a non-left turn
9          do POP(S)
10     PUSH( $S, p_i$ )
11 return S

```

Figure 1. Graham's scan algorithm pseudocode

Although the convex hull algorithm, being geometric in nature and relatively easy to visualize, is a conducive one for a visual animation, this was not one of the reasons I chose it. Had I chosen a less graphical algorithm, I would simply have had to be more creative in deriving a visualization for it. For example, many algorithms animated by other systems, such as some of the sorting algorithms illustrated by XTango and Zeus, are created from the authors' visualization (or research) of that algorithm. Douglas, Hundhausen, and McKeown, in a University of Oregon research project detailed in *Toward Empirically-Based Software Visualization Languages* [13], used a visual storyboarding technique to study tendencies of human conceptualization of the bubble-sort algorithm. Their report illustrates common features discovered during this experiment, such as the fact that all of the participant groups used a different element (squares, stick figures) to represent the elements being sorted, different method to illustrate magnitude of elements (numbers, colors), and so on. Had I chosen a non-graphical algorithm to animate, I would have used a similar process (albeit with much

less formality) to derive my own visualization of that algorithm. One may also argue that nearly any algorithm will yield at least one obvious graphical representation. It is rare, for example, to find an algorithm in a traditional text without a commonly associated accompanying picture. As it is, I do not believe the geometric algorithm I chose adversely affects my overall goals, nor do I think it makes my tasks any easier or harder to accomplish than a non-graphical algorithm would have made them. Although Graham's scan convex hull algorithm has an overall natural graphical representation, its detailed steps do not suggest intuitive individual representations. I had to determine not only the interesting and salient characteristics of the algorithm, but also how to represent them graphically in the animation. For example, the process whereby three points are compared to determine if they create a left turn angle, as described above and seen in Figure 1, might be illustrated in a variety of ways. It was part of my task-centered UI design process to find the best manner of illustrating this step, as well as the other detailed steps.

## ***2.4 Design Details***

As indicated by the previous summary of the task-centered UI design process, one of the first and most important questions a designer needs to ask and understand is "Who are my users?" In this study, the answer was an easy one: students who want or need to learn about the algorithm. I am not making any assumptions about whether these users are using this animation tool in conjunction with other forms of learning, as this is not the focus of my research. Also beyond the range of my thesis are issues relating to how easy or difficult it may be to create additional animations using my method or application program, as this too, is not the focus of my thesis. Instead, I am interested in the

individual's learning process, and as such, how different views of an algorithm animation and different methods for accomplishing this view affect the learning process.

Also worth repeated a mention is the fact that I am designing this algorithm animation program from the task-centered *user* interface approach. In other words, my design and implementation is centered around the user, who is in this case the learner, not the teacher or the animation designer. This is an important distinction to make, as both are valid views with quite different approaches and results. One observation often present in the literature describing past algorithm animation programs is that the users have had to possess expert knowledge about the algorithm before using the animation in order to benefit from it substantially. "For a student to benefit from the animation, the student must understand this mapping (from the abstract computational algorithm domain to the animated computer graphics domain) and the underlying algorithm upon which the mapping is based" [23]. This implies that the wrong approach might have been taken in the design process of these animations, perhaps having focused on the teacher's view and understanding of the algorithm instead of the student's (user's) and his or her probable inexperience.

The next critical question in the task-centered UI design method to ask is, "What tasks do the users hope to accomplish in using the animation program?" There are different ways to answer this question, ranging from a general answer to more specific learning tasks. In general, the users hope to learn as much about Graham's scan convex hull algorithm as possible using the animation program. This generality, however, must be broken down and analyzed more specifically in order to be of practical use in the design process. I settled on the following basic characteristics, or user learning tasks, of this algorithm that

I felt necessitated illustration in some way, if possible, in the animation program that I was to develop:

1. Labeling (in order) all points in set Q.
2. How the points/angles are compared.
3. How the stack is initialized and works throughout algorithm.

Closely related to the question of what are the learning tasks of the users in using this system, is, “How am I going to design a system that will help the users accomplish these learning tasks?” This question really drove the rest of the design process. For experimental purposes, and in an effort to learn as much about how an algorithm animation system best assists in the learning process, I settled on designing five different views of the algorithm in my animation program. The five views developed are associated with two varying levels of information provided, labeled *Overall Strategy* and *Implementation Detail*, and with three different methods of illustrating the algorithm animation, *Visual*, *Visual & Code*, and *Code*.

The two different levels of information provided, Overall Strategy and Implementation Detail, differ in the amount of information provided to the user. For example, in the views displaying the algorithm’s code (pseudocode), the Overall Strategy level has a summary version of the pseudocode shown in Figure 1, whereas the Implementation Detail level has the lower-level code shown exactly as in Figure 1. Similarly, in these same views that illustrate the algorithm’s code, the accompanying data structures in the Overall Strategy level show just the key stack, and in the Implementation Detail view, other variables are shown in addition to the stack.

The three methods designed and studied for accomplishing the learning tasks, *Visual*, *Visual & Code*, and *Code*, work in conjunction with the different levels of information provided. The *Visual* method uses only a visualization of the algorithm to try and help the user accomplish the learning tasks, and has only the Overall Strategy level of information corresponding to it (the *Visual* method has no Implementation Detail counterpart). The *Visual & Code* method, which has views for both Overall Strategy and Implementation Detail levels of information, uses a combination of visualization and algorithm pseudocode to accomplish the learning tasks. An important note to make is that the *Code* method, whether used by itself or in conjunction with the *Visual* method, also contains graphical representation of the data structures of the algorithm. So as not to confuse the reader, the data structures component of the *Code* method is usually spelled out throughout the remainder of this document; however, where it is not, the reader must remember that the *Code* method always contains the data structures as well. Finally, the *Code* method, which also has views for both levels of information, uses a combination of pseudocode and data structures to help the user accomplish the learning tasks. Figure 2, which displays an actual screen from the animation program, illustrates the five different views available to the user as just discussed.

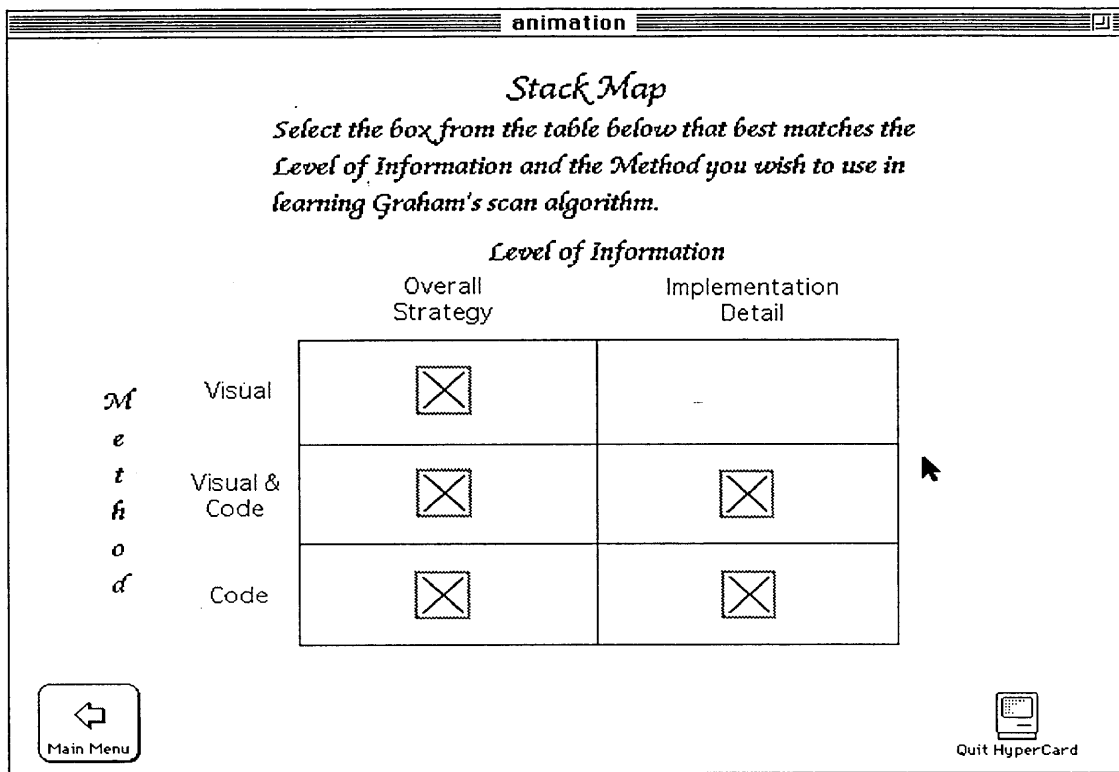


Figure 2. Different views available to user

To summarize, the five views provided by the animation program are listed below, along with their respective identification codes (in capital letters) utilized throughout the remainder of this document.

1. Visual / Overall Strategy (VO)
2. Visual & Code / Overall Strategy (VCO)
3. Code / Overall Strategy (CO)
4. Visual & Code / Implementation Detail (VCI)
5. Code / Implementation Detail (CI)



## ***2.5 Common Features***

Before detailing each of the five unique views found in the animation program, it may be worthwhile to discuss the common features designed for the animation program in general. At the start of the algorithm animation program, each user has the option of looking at a preliminary description of the convex hull algorithm. This preliminary description is the same for all users, regardless of the view he/she will ultimately look at. Included in this initial description is a textual description of the convex hull algorithm as well as a simple animation of the algorithm, a list of other algorithms (in addition to Graham's scan) that compute the convex hull, and the run time of Graham's scan algorithm for computing the convex hull.

Other common features of the program, which deal primarily with navigation through the animation, were incorporated into all views of the animation whenever applicable, and are summarized in Figure 3. The user has the basic option of viewing the animation frame-by-frame, or playing multiple steps of the algorithm at once. The frame-by-frame option allows the user to go either forward or backward, the backward option providing the animation's historic component, something often missing from other algorithm animation systems. For longer steps, the user can often choose between fast or slow play speeds. There is a stop button for all play options that allows the user to stop and eventually resume (using any of the other options/buttons available) the animation. Once a step of the animation consisting of multiple moves has completed playing, or is at the last frame, the user can reset the entire step from the beginning. This allows the user the ability to review different steps of the animation, possibly viewing them in a different manner than before. At any time during the course of viewing the algorithm animation, the user can choose to start the animation over from the beginning, or return to the stack map, where they can then choose a different view of the algorithm, or exit HyperCard

altogether. This variety and combination of navigational features provides a flexible viewing environment for the user, allowing him/her to choose his/her individual pace throughout different parts of the animation.

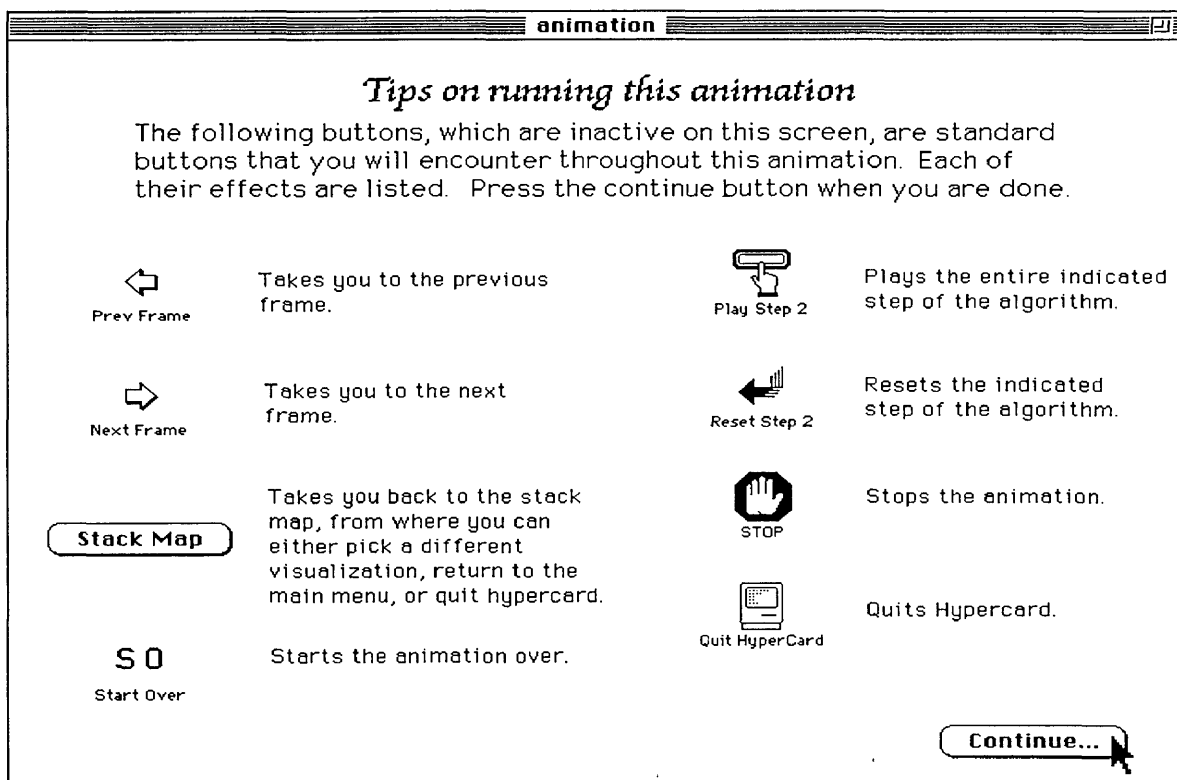


Figure 3. Navigational Buttons

This algorithm animation program uses pre-determined datasets only, meaning that the user is not able to choose his/her own data with which to run the algorithm. While allowing the user to create individualized exercises might suggest a more effective algorithm animation program, see [23], to incorporate such a feature into this program was beyond the scope of the current research. This feature is, however, a topic discussed in the future work section of this paper.

Although HyperCard version 2.3 has both sound and color capability, I chose not to use sound in my animation, and to use color only minimally. Both of these features, if not used carefully, can be a distraction in a user interface and can often adversely impact its effectiveness [19]. Additionally, the nature of my animation is such that sound would be an extraneous feature. I might have chosen to use color more prominently than I did if this feature of HyperCard was easier to implement. As it was, I found the color tools difficult to use in this version of HyperCard, to the extent that I generally avoided them. The difficulties I had in using color are discussed in greater detail in the Implementation section of this paper, along with some other HyperCard shortcomings I experienced.

An attempt was made to maintain consistency between the various views of the animation, although this was not a critical issue in my initial research. Because of the specific nature of my user testing, and because I had a limited number of users to test with, it seemed likely that the same user would not look at multiple views of the algorithm animation. However, in addition to being good design practice to add the extended functionality, this animation system may someday be expanded for future work. It therefore made sense to design the animation so that all views were consistent in their overall layout, structure, point set  $Q$ , convex hull and button availability.

## ***2.6 View 1 -- Visual / Overall Strategy (VO)***

The VO view of this algorithm animation program is the simplest, shortest and most concise of all the views. It consists of a visualization of the set of points  $Q$ , and the animation of that point set as the convex hull is determined. Initially, only the set of points appear on the screen. When the user chooses either play or next frame buttons, the

points are first labeled as a ray sweeps over the set of points in a counter-clockwise rotation. Once the points are labeled, the appropriate angles are compared to determine if they form a non-left turn or not (the words left and non-left appear above the angle to notify the user which type of angle it is), and therefore, whether they currently remain part of the convex hull or not. Although not noticeable in Figure 4, which shows a step in this view of the animation, the labels of the three points whose angle is currently being analyzed appear in the color red to help the user to distinguish them more easily. Eventually, after all the points in the set are analyzed, the convex hull is displayed.

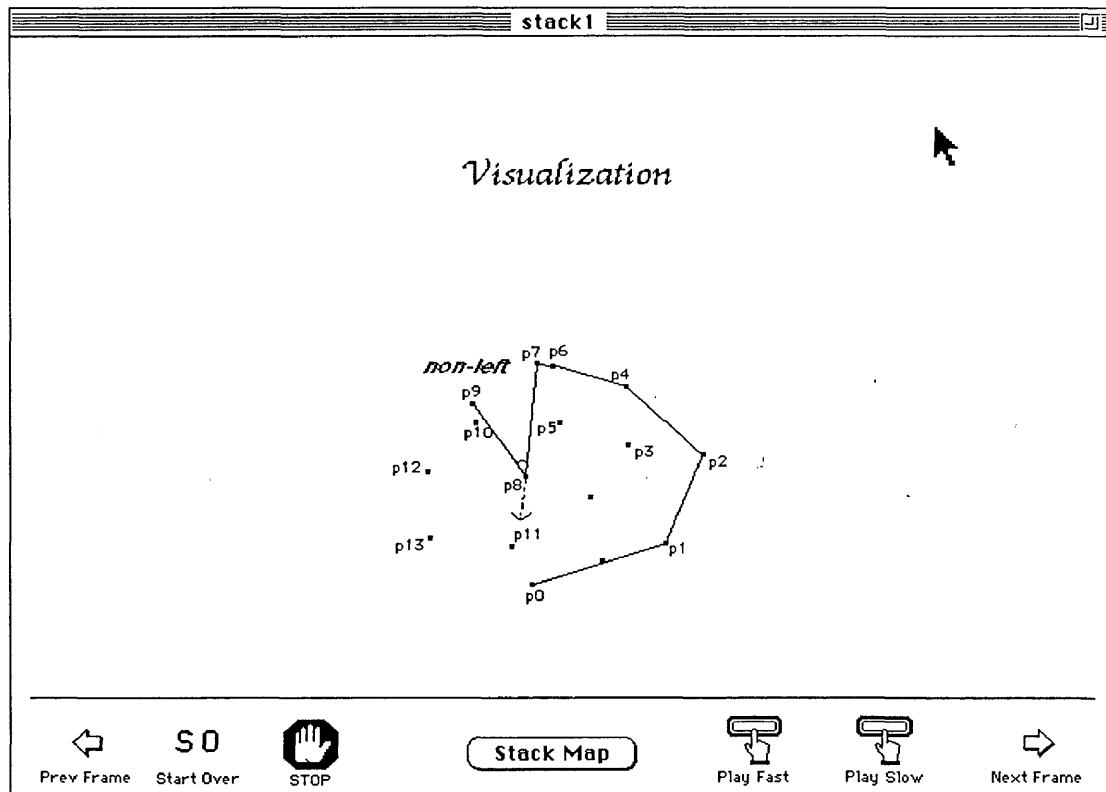


Figure 4. View 1 -- Visual / Overall Strategy (VO)

## ***2.7 View 2 -- Visual & Code / Overall Strategy (VCO)***

The VCO view of the animation, as depicted in Figure 5, has three distinct partitions: the pseudocode, the visualization, and the key data structures. The pseudocode is high-level; the original pseudocode as seen in Figure 1 has been summarized into four main steps and displayed as our high-level pseudocode. The current displayed step of the algorithm is highlighted in bold and is boxed within the pseudocode partition. The visualization partition has the same picture as in view 1, but with additional text explicitly stating which points and angle are being evaluated. These point labels are also in color at this point, as described in view 1. Finally, the data structures partition shows a visualization of the key data structure in this algorithm, a stack. Pointers show the TOP element of the stack as well as the NEXT-TO-TOP element. The motion of an element getting either pushed onto or popped from the stack is also animated. As the user navigates forward through this view of the animation, appropriate and meaningful elements in each of the partitions are animated in conjunction with one another and relative to the active step of the algorithm. While this motion or animation is difficult to illustrate in this static document, the same type of animation exists for the remainder of the views in the program (all views consisting of two or more partitions).

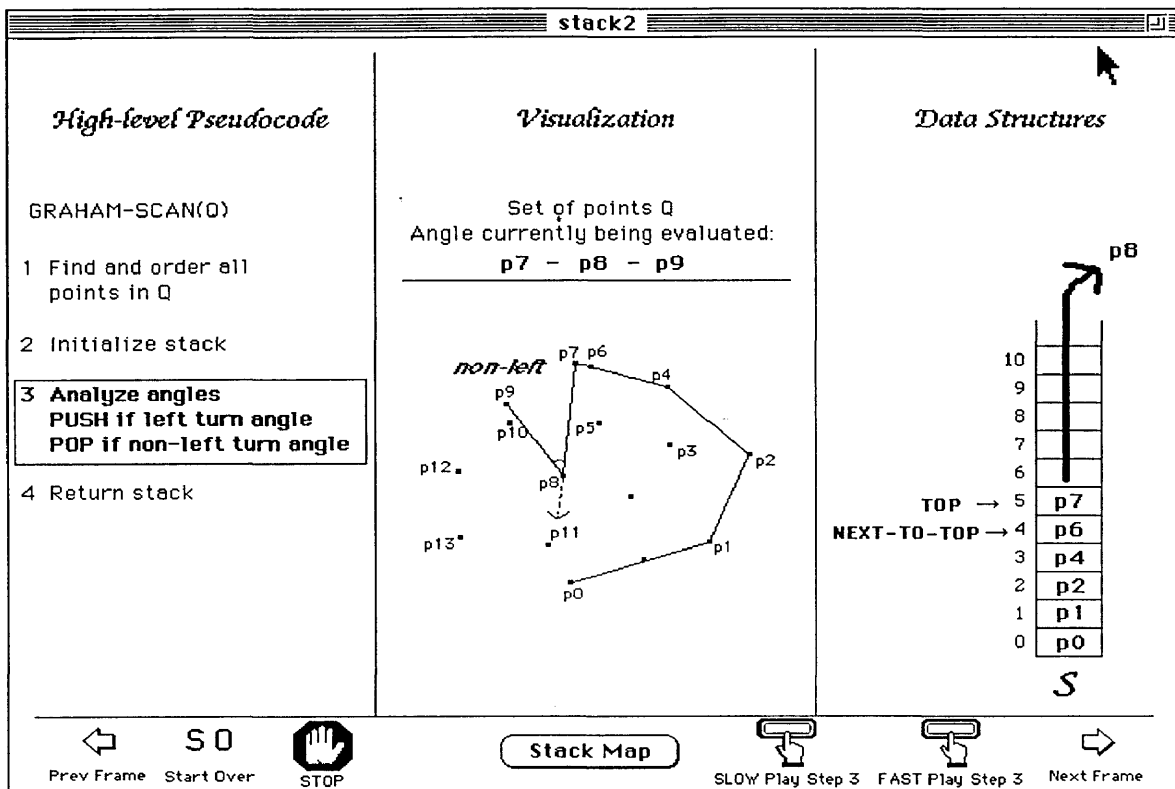


Figure 5. View 2 -- Visual & Code / Overall Strategy (VCO)

## 2.8 View 3 - Code / Overall Strategy (CO)

The CO view of this animation system, as shown in Figure 6, has two basic partitions, the pseudocode partition and the data structures partition. Since this view corresponds to the Overall Strategy level of information provided, the pseudocode partition displays the higher-level pseudocode, as discussed in view 2. Within the pseudocode partition, there is a sub-partition showing only the set of points in  $Q$ , along with their labels after step 1 of the pseudocode runs. There is no visualization of the algorithm on this set of points however, as this is not a visual view. The points are shown just to give the user a sense of perspective so that he/she might better relate the pseudocode to the data structure movements. The active step of the pseudocode is highlighted as in the other views. The

stack, being the key data structure, is displayed in the data structures partition, along with the added text informing the user which points and angle are currently being evaluated, and if that angle is a left-turn angle or not. The stack features TOP and NEXT-TO-TOP pointers, and also illustrates movements of elements being pushed and popped.

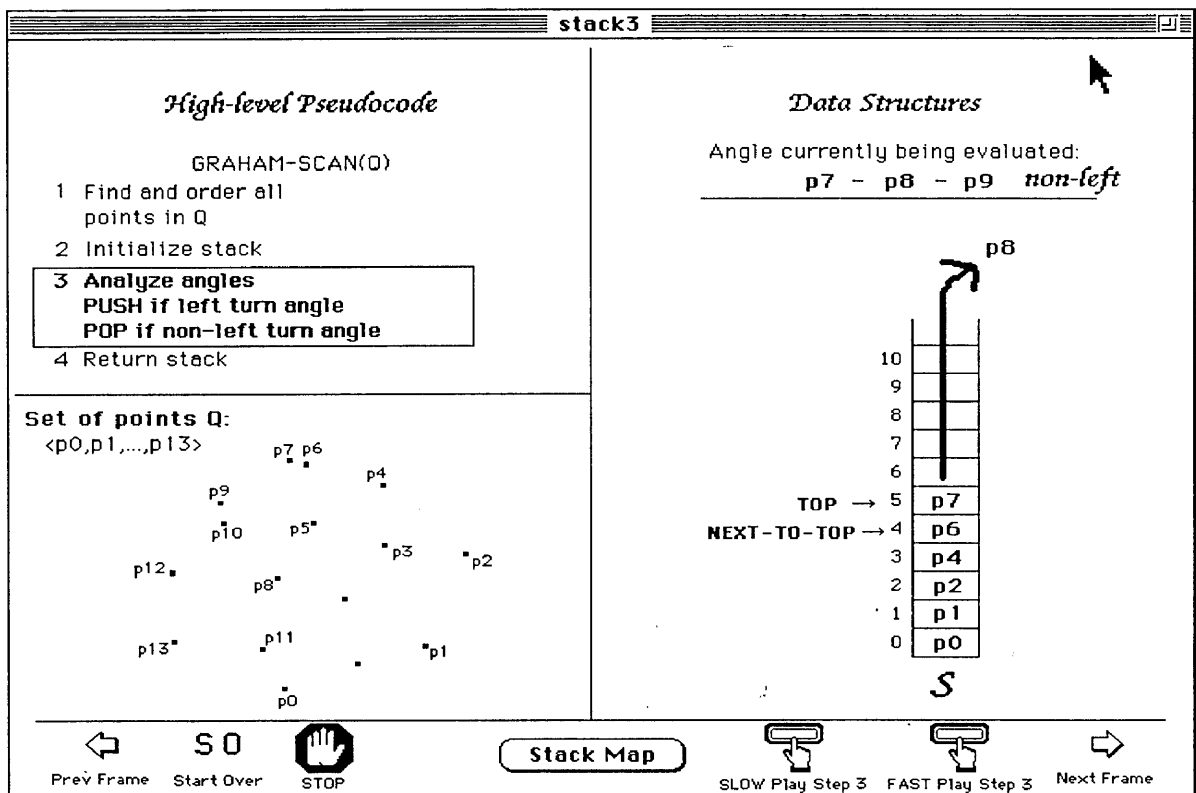


Figure 6. View 3 -- Code / Overall Strategy (CO)

## 2.9 View 4 -- Visual & Code / Implementation Detail (VCI)

The VCI view of the animation, shown in Figure 7, is similar to view 2. The same three basic partitions exist in each case, with more detail provided for this implementation level view. The pseudocode partition has the lower-level pseudocode displayed, exactly as

appears in Figure 1. The current step is highlighted the same way as in view 2. The visualization partition is identical to that in view 2. The data structures partition has the stack data structure, but also displays the data variables  $i$  and  $m$  which are used in the lower-level pseudocode of this view.

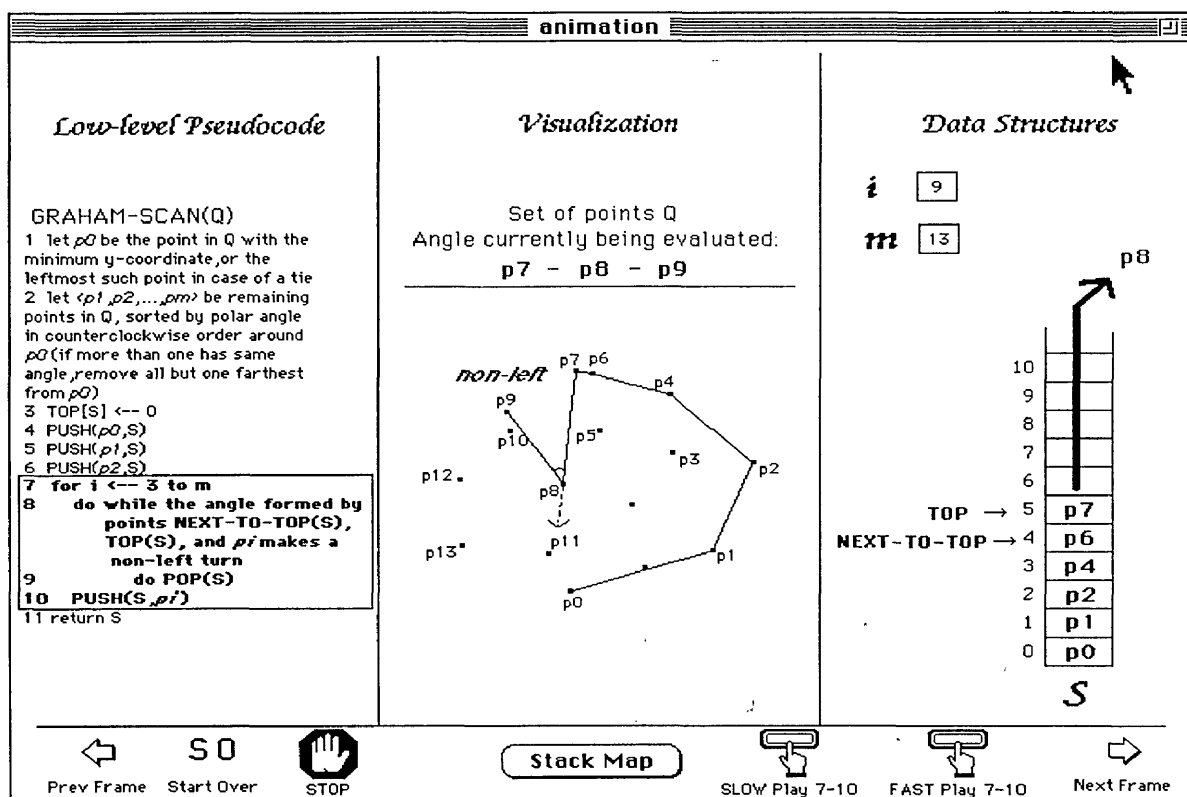


Figure 7. View 4 -- Visual & Code / Implementation Detail (VCI)

### 2.10 View 5 -- Code / Implementation Detail (CI)

Finally, the CI view, shown in Figure 8, is similar to view 3. The pseudocode partition displays the lower-level pseudocode (see Figure 1), and the active step is highlighted as in each of the other views. As in view 3, the set of points in  $Q$  are illustrated, but it



appears in the data structures partition instead of in the pseudocode partition, solely due to spatial reasons. The data structures partition is nearly identical to its appearance in view 3, with the exception that this view, being the Implementation Detail level, shows the active point  $p_i$ .

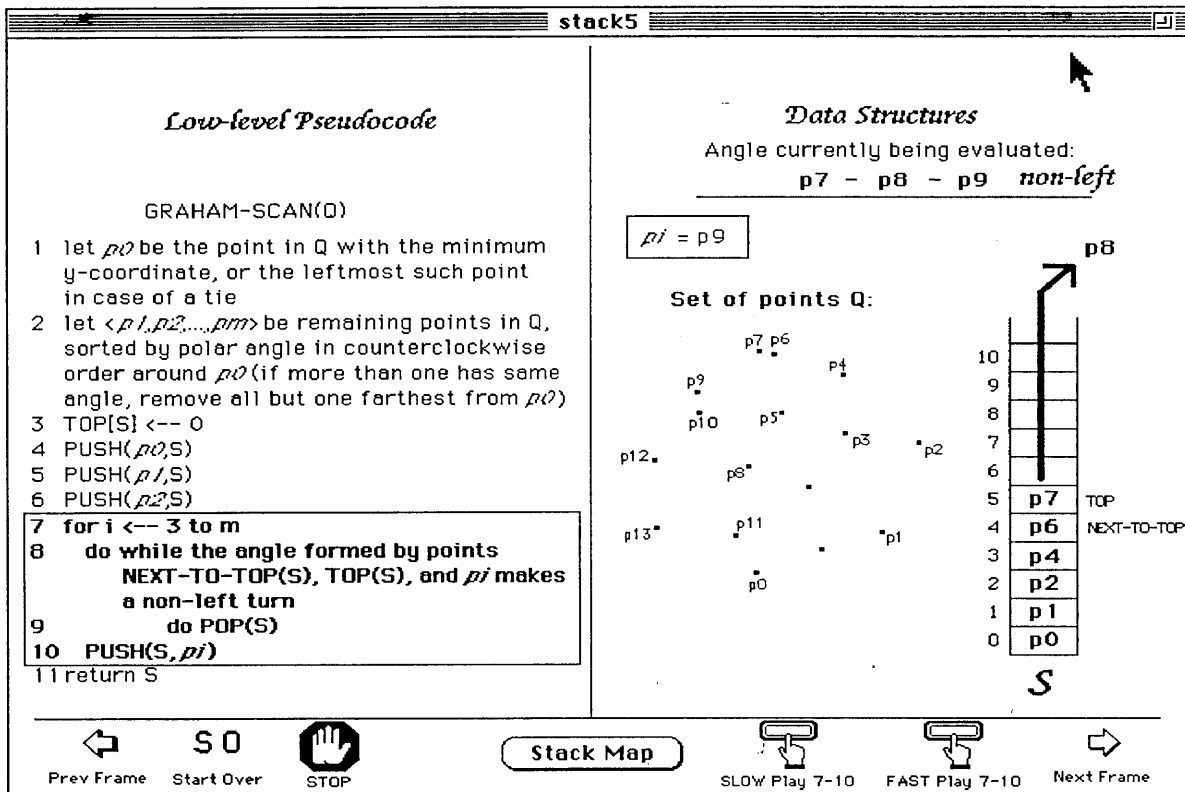


Figure 8. View 5 -- Code / Implementation Detail (CI)

### 3. Implementation

The final animation program consists of five different HyperCard stacks, one for each of the five unique views discussed in Section 2. The stacks range in memory required from 553K to 689K, and they range in size from 44 total cards to 84 total cards. As anticipated

during the design phase, neither machine memory nor processing speed was in any way detrimental to the performance of the implemented algorithm. I was able to create as many cards in each stack as needed to maintain a smooth and fluid appearance for the animated parts of the program. Additionally, the choice to use HyperCard as the programming application tool proved to be a sound one. HyperCard's functionality enabled me to implement nearly all of the design goals. HyperCard, and its scripting component HyperTalk, were not difficult application tools to learn, the learning process being made easier by the abundant available documentation and my prior Macintosh and programming experience.

The biggest shortcoming I found in using HyperCard was in its color and shading functionality. HyperCard does support color, but it was a difficult and time-consuming process to incorporate the color text or object into a stack card, and, further, to edit that color once it became a part of the card. The color feature of HyperCard is located within a "color tools" function window, which forces the user to open and close this function each time he/she wishes to create, edit, or delete any color aspect of the card. While this may sound like a typical user process when invoking features in application tools, it was an especially time-consuming and tedious one as implemented in HyperCard.

Additionally, color text does not copy when a card is duplicated. Since the creation of an animated sequence essentially means copying a card multiple times and changing each card slightly from the previous one, the lack of color copying capability makes this process very ineffective in dealing with color text or objects on the source card. I would have liked to have incorporated more color into the animation program than I did, but the overall process was too difficult and too time-consuming to make it practical.

Other functionality within HyperCard that I found poorly implemented and difficult to use was its text and drawing capabilities. Once text is created and positioned on a card, it is not possible to select that text again as a group and to edit pieces of it. The user is forced either to erase the part he/she wishes to edit and refit the new text with the existing text, or to start the whole text over again from scratch. This also makes the interweaving of bold, italics, or super/subscripts into the same text passage nearly impossible. This feature of HyperCard was so ineffective that for the longer passages of text in my animation program, I was forced to create them in a different, more user-friendly Macintosh application tool, and then import them into the HyperCard stack. The drawing capabilities supported by HyperCard were also limited. HyperCard did not support some of the drawing tools that I needed in creating my program, such as arcs, curves, and arrows. This forced me to either choose different, less desirable graphics in their stead, or to draw the pictures I wanted using free-hand, a process that often left me editing these objects at the pixel level.

## **4. User Testing**

### ***4.1 Set-up***

The individual user testing sessions, which lasted between one and two hours apiece, were conducted over the course of a week. In total, six users tested the animation system; one of the five different animation views was randomly chosen for each user prior to testing, with the last user duplicating one of the previously tested views. Each user testing meeting began with a brief discussion of the research being conducted and the pending testing procedures. The user was reminded that participation was purely voluntary, and that he/she was free to stop the testing at any point. The user was likewise made aware of the informal nature of the testing, and assured that anonymity would be

preserved even if the results were published. Any concerns or questions, the users were told, could be directed to the Instructional Review Board (IRB) administrator, who reviewed and approved the user testing portion of this research [see Appendix 1]. A discussion of the testing procedures followed, with the user instructed to ...

- ... take as much time as necessary to learn as much about the algorithm as possible.
- ... use the "think aloud" methodology [19], or thought verbalization, while navigating through the program.
- ... use the resources provided by the animation program to resolve unexpected difficulties.

It was understood that I would remain in the room observing and taking notes, and that I would only be available to assist the user in the event of a seemingly unresolvable problem. The users were subsequently told that after using the animation program they would be given as much time as necessary to complete a post-test and a questionnaire, both to be taken without the animation program available to them. It was made explicitly clear to the users that they might not be able to answer all of the questions on the post-test, depending on which view of the animation they were exposed to, and that it was the animation program that was being evaluated, not them. In place of individual post-tests, the users were to be given the same post-test, with the stated assumption that some questions would be unanswerable for certain users. After the above material was fully discussed and understood, the user was asked to read and sign a consent form [see Appendix 2], and then to begin the animation program.

## 4.2 Results

None of the users reported having any knowledge of the convex hull algorithm prior to using the animation program. The average amount of time that the user looked at the animation program was approximately 14 minutes, while the average amount of time taken to complete the post-test was 31 minutes. There were a total of 14 questions on the post-test, and each of these questions were marked (after the testing) to be one of two types: *procedural* or *conceptual*. See [23] for further discussion.

The procedural questions are short-term memory type questions that were designed primarily to determine if the user paid close attention to the animation program. As such, the procedural questions did not necessarily test understanding of the algorithm presented, but rather exposure to the animation material. Although this hypothesis was not tested, one might surmise that since the procedural questions tested short-term memory and diligence of the users in using the animation program, many of the users might not be able to answer some of the same procedural questions on a delayed post-test that they were able to answer initially.

Conversely, the conceptual group of questions were mostly open-ended, essay type questions that tested the user's overall level of comprehension of the convex hull algorithm. It is this group of questions and answers that indicates how well the user understood the algorithm, and subsequently, how well the animation program succeeded in helping the user to learn the algorithm. Examples of two questions from the post-test, one procedural and one conceptual in nature, are listed below [see Appendix 3 for the full list of post-test questions and their type designations].

*Procedural:* What were the respective purposes of the variables  $i$  and  $m$ ?

*Conceptual:* How is it determined if a point is a vertex on the convex hull or not?

To facilitate the displaying of results, and in order to maintain anonymity, each user was referred to by the identification code assigned to each view. For example, the user who used the Visual / Overall Strategy animation view was VO, the user who used the Visual & Code / Overall Strategy view was VCO, etc. Since two users utilized the Code / Implementation Detail view of the animation, they are coded as CI#1 and CI#2 respectively.

The users' performance on the procedural questions are reflected by the chart in Table 1. The procedural questions were each assigned a point value [see Appendix 3], and these questions on the post-tests were graded much like a test would be graded, with the number of points awarded for any given question corresponding to how well the user answered the question. If a user was unable to answer a question because of the particular view he/she was exposed to, that question was deleted from the grading. For example, the user who saw the VO animation view would not be expected to answer the procedural question about specific variables listed above. This explains the discrepancies in the totals numbers for the procedural group of questions for each user. As illustrated by Table 1, all of the users scored sufficiently high on the procedural questions, indicating that they were engaging the animation program diligently and paying close attention. The fact that the users displayed adequate procedural proficiency lends greater authority to conclusions derived from the results of the post-test's conceptual questions.

	VO	VCO	CO	VCI	CI#1	CI#2
<b>Raw Score</b>	15/20	30/35	31/35	46/50	44/50	36/50
<b>Percentage</b>	75%	86%	89%	92%	88%	72%

Table 1. Matrix of Procedural Understanding

The answers to the post-test's conceptual questions were reviewed to assess how well the users understood the three main points that the animation attempted to illustrate: labeling the points in the correct order, analyzing the angles to determine if a point is part of the convex hull or not, and the relation to the stack to the overall algorithm. After studying the users' answers to the conceptual questions, I ranked their understanding of the algorithm for each of the three main points as being either HIGH, MEDIUM, or LOW (unlike the point based system used for the procedural questions). I chose to use a general ranking system for the conceptual questions rather than the more quantitative point approach because these questions were of an essay type, and did not necessarily have absolute answers. Most critical to the final conceptual ranking given was the user's success in answering the last conceptual question on the post-test, which called for illustrating, step by step, how the convex hull is determined for a given set of points. This last question was significant in that it quickly revealed any of the user's misconceptions about the overall working of the algorithm after using the animation program, regardless of how well other specific conceptual questions on the post-test were answered. For example, a user might have answered a limited conceptual question adequately, one that referred to one specific part of the overall algorithm, but then failed to demonstrate comprehensive understanding when faced with a question, like the last one, that required associating the various pieces of the algorithm. A user may also have

been able to answer a specific conceptual question adequately by writing down a suitable answer earlier on the post-test, but then was unable to illustrate his/her practical knowledge of that same answer using the exercise presented by the last conceptual question. Table 2 shows the results of the users' conceptual understanding, ranked from their answers to the conceptual questions on the post-test.

As I was the only person 'grading' the users' post-tests and ranking their knowledge of Graham's scan algorithm, I was conscious not to let any external biases interfere. I had no prior experience as to the abilities of any of the users, and as previously mentioned, each of the user / animation view pairs were chosen randomly prior to user testing. I believe I was successful in maintaining a fair and objective mind while evaluating the users' experience with the animation program, an effort that was made easier by my past teaching (and grading) experience.

Key Learning Tasks	VO	VCO	CO	VCI	CI#1	CI#2
Labeling points in correct order.	HIGH	HIGH	LOW	HIGH	MED	HIGH
Analyzing angles to determine if a point is part of the convex hull or not.	LOW	HIGH	HIGH	HIGH	LOW	LOW
Relation of stack to overall algorithm.	-----	MED	HIGH	HIGH	LOW	LOW

Table 2. Matrix of Conceptual Understanding

As seen in Table 2, those users provided with a visual animation, step-by-step textual descriptions of the algorithm in the form of pseudocode, and illustrations relating the key



data structures (VCO,VCI), had the highest overall conceptual ranking. Conversely, the users presented with a view of the algorithm that had either just the visualization alone (VO) or the code (and data structures) alone (CO, CI#1, CI#2) had the lowest overall conceptual understanding of the algorithm.

The question of whether the overall strategy level of information works better or worse than the implementation detail level of information, yields mixed and somewhat surprising results. In the absence of a visualization, higher-level information (Overall Strategy) appears to be more efficient than lower-level information (Implementation Detail), which is unexpected. It seems as though the users exposed to code (and data structures) but not to visualization (CO,CI#1,CI#2 in Table 2) were better prepared for the conceptual questions when presented with the higher-level pseudocode (CO) as opposed to the lower-level pseudocode (CI#1,CI#2). This is surprising in that, given the lack of an accompanying visualization for these views, one would expect that the more detailed information would be preferable for supplying missing links than the higher-level information. One possible explanation for the apparent advantages of higher-level information in the absence of an accompanying visualization, however, might be that without a visualization to make the steps of the algorithm clearer, the user may become mired in the details of the lower-level code and further confused. Comparing the levels of information may not be as relevant for VCO and VCI, since any missing information in these views might also be explained by the accompanying visual aspect of these views. However, if one does attempt any sort of comparison for the VCO and VCI views, the results appear to be more expected and not all that significant. The user exposed to the VCI view seemed to have a slightly better conceptual understanding of how the stack is used in the algorithm than the VCO user. This difference can easily be reconciled by the

fact that the VCI user had the lower-level pseudocode in his view, which explains the stack in more detail than the higher-level pseudocode shown to the VCO user.

There do not appear to be any tendencies indicated by the conceptual chart which suggest that the program's illustrations favored any one of the three key learning tasks over the others. However, it appears most conclusive that views with 'semantic redundancy' (a concept defined and referenced earlier in this paper) in the form of a visualization accompanied by code, portrayed the key aspects of the algorithm more thoroughly than the views that did not contain this redundancy.

### ***4.3 Observations***

The following notes are derived from my observations of the users as they viewed the animation system, from things they may have said in "thinking aloud" while using the system, and from the post-test questionnaires (see Appendix 4 for a complete listing of the questionnaire) filled out at the end of the user testing sessions:

- All of the users initially looked at the preliminary description screens and then the screen that provided navigational tips on running the animation (Figure 2) before proceeding on to his/her specific animation view.
- Some of the users who used a view of the animation that lacked certain information commented later that the inclusion of the view's withheld information would have helped them. For example, user VO, whose view did not have any textual descriptions or data structures, remarked after testing that the visual picture alone was not sufficient for learning the algorithm

thoroughly, and that he would have benefited from textual assistance accompanying the visual image. Similarly, user CI#2, whose view did not have the visual image animation, said that a picture revealing how the angles were compared would have aided her understanding.

- All of the users reported enjoying the animation program; all felt that having such a tool would greatly assist and often facilitate learning algorithms.
- The buttons used for navigating through the algorithm were used in a variety of ways. Some users used the *play* buttons (both fast and slow), while other users used the *next frame* buttons for forward movement in the program. It became obvious that different users preferred different methods of navigation; this might indicate that the optimal animation program would offer at least as many options for navigation as this program did. The flexibility of navigation methods in this program was cited in some of the questionnaires as being a beneficial characteristic.
- None of the users utilized the *prev frame* button for backwards, historical movement in the program. It may be that the nature of this algorithm does not demand the degree of historical perspective that other algorithms might (like a sorting algorithm, for instance, where it's critical to compare detailed information in a frame-by-frame manner).

## 5. Conclusions

So, are we any closer to defining key elements that make algorithm animation systems more effective, and can we say anything about using the task-centered UI design methodology as the basis for designing such systems? Despite the relatively ad-hoc

nature of this research, user testing analysis revealed definitive trends and positive results that can contribute to the dialogue about algorithm animation systems. Perhaps the most significant conclusions drawn from this research is the evidence that supports...

- the increased efficacy of an algorithm animation system designed from the user's perspective.
- incorporating an appropriate level of visual and textual semantic redundancy in the system, and designing this redundancy, again, from the user's perspective.

With the learning tasks of the user constituting the guiding issues of the entire design process, an approach consistent with the task-centered UI design methodology, the resulting algorithm animation system was concise and geared towards accomplishing specific goals. A natural complement to the user-oriented design process is determining and offering the optimal blend of information in the animation program to help the user accomplish the defined learning tasks. This 'optimal blend of information' was shown in our research to translate directly into semantic redundancy in the form of a visualization accompanied by pseudocode or other meaningful textual descriptions. The results of this study indicate that the views featuring this type of semantic redundancy (VCO,VCI) were more efficient in accomplishing the learning tasks than the views that did not (VO, CO, CI).

The results of this study also indicate that beyond simply including visual and textual information, an algorithm animation system's success depends on a careful assessment of what information is offered and how it is presented. That a user-oriented approach to designing and including this information is critical recalls, perhaps, those experiments

cited earlier in this paper in which results were not as positive as initially hypothesized. Some of these other experiments were conducted using a visual algorithm animation supported by lecture comments (audio) and/or textual descriptions, but these other forms of semantic redundancy were perhaps not incorporated into the learning process in a user-oriented manner, thus making them less efficient and adding to the disappointing results [23][24].

Additional evidence that supports the conclusion that the **visual** animation and the **textual** descriptions are the key components of the semantic redundancy is provided in the comparison between the VCO,VCI views and the CO,CI views of the animation program in this study. It could easily be argued that the CO and CI views of this animation did contain a form of semantic redundancy, as these views displayed not only pseudocode for the algorithm, but also the key data structures. However, it has been clearly stated from the *Results* in Section 4.2, that the views combining a visualization of the algorithm, the pseudocode, and the relating data structures (VCO,VCI) were more effective than those views that combined just the pseudocode and relating data structures (CO,CI). This strongly suggests that the visualization accompanied by the text is the key component in the mix of information presented to the user. Furthermore, within the views combining pseudocode and relating data structures, the manner in which the pseudocode is displayed (high-level, low-level) also appears to be significant.

Among the other conclusions that can be reached, it appears that an algorithm animation system that provides various methods of navigation and that allows for flexibility based on individual preferences is helpful to users, and ultimately contributes to a more effective learning tool than one that is less user-oriented. Sometimes, however, features that are expected to assist the user may also prove to be nearly irrelevant: considering

that the rewind function was never utilized, for example, it may be that historical snapshots are more critical to certain types of algorithms than to others. Ultimately, this study does not diverge from the finding of earlier experiments that users respond positively to using algorithm animation systems, an encouraging notion for the use of such systems in the future.

## **6. Future Work**

There are many possible branches of continued study in this area that might use this research project as a base or as a supplement. To begin with, a more formal, empirical version of this study might be conducted, one that more thoroughly considers the potential biases and that alters the controlling variables accordingly (includes a larger group of users, achieves a greater pre-testing knowledge of these users, establishes more realistic post-testing conditions, conducts delayed post-testing, etc.). It would also be intriguing to study to what extent an animation system would benefit from further iterations in the task-centered UI design process (key point 7, as defined in section 2.1), with intensified user interviews and perhaps greater integration of user ideas into the animation program. This animation program, or one like it, might also be expanded to include multiple algorithms, user-defined data sets, more color, and on-line quizzes to evaluate the impact of these additions on the conclusions reached. Additionally, remembering that none of the users in this study chose to look at the historical perspective provided by the system, further research might search for the conditions when historical access is more integral in an algorithm animation system.

Another possible focus for future research would be to continue to use the task-centered UI design process to target the visualization and textual comments of the animation to make them more efficient as viewed in conjunction with one another. The conclusion has already been made that the visual / textual pair makes for the most effective algorithm animation, but future work might endeavor to use the task-centered process to determine the best way of using or displaying each of these forms of information in relation to one another. The results of this study indicate that the level of information provided in an algorithm animation system (high-level, low-level) is important, but exactly how and to what extent remains to be further studied. Different levels of textual descriptions depending on the accompanying visualization, hidden versus visible text, and help text are just a few of the different paths that such further experimentation might take.

Future work might also consider the efficacy of teaching algorithms using an animation system, and how the efficiency of the accompanying textual descriptions might compare with audio (lecture) descriptions. This current study suggests that the semantic redundancy provided by textual comments (pseudocode) is clearly advantageous to the user, especially when the user has the ability to control the pace and the manner of viewing the comments during the animation. Also intriguing to consider would be the question of whether the more 'permanent' and 'reusable' textual comments have more utility than the fleeting words of a teacher who lectures as the animation runs. While the presentation of a lecturer is a more flexible and dynamic medium, the embedded textual comments of the animation system, although "stagnant," remain available if the user wishes to revisit and review any areas of the animation (provided the animation has rewind capabilities). A formal study analyzing the advantages and disadvantages of audio versus textual comments in an algorithm animation program might thus assist

efforts to create and use algorithm animation systems more effectively, and to better understand their value as instructional aids.

A more ambitious goal in the future of this research area would be to develop an algorithm animation system using the program developed in this research as the prototype, but one geared towards a wider user base. Such a system might be oriented towards both the algorithm animation creators (educators) as well as towards the animation viewers (students). The differences in targeted users (animation creators vs. animation viewers) might explain the discrepancies between previous algorithm animation systems and the prototype developed in this research. Whereas systems such as Balsa and XTango were developed with the intent of not only providing a solid user interface, but also of allowing for relatively quick and easy implementation of algorithm animations, the prototype created in this research focuses solely on the efficiency of the user interface. Ultimately, then, future work in the area of algorithm animation systems will yield a system that is efficient from both the educator's and the student's perspective, that allows for greater variety and more dynamic construction of algorithm representations, but that does not sacrifice the tangible benefits of the end-user (animation viewer) oriented approach that this study has favored.

The research area of algorithm animation is currently an active and dynamic one, which means that results of a study like this one point towards new studies as much as they suggest definitive conclusions. It is hoped, then, that this study will be an encouraging and useful aid in continued efforts to improve the effectiveness of algorithm animation systems as learning tools.



# Appendix 1. IRB Approval.

Form RA-10S  
(Rev. 2/95)

## THE UNIVERSITY OF MONTANA INSTITUTIONAL REVIEW BOARD (IRB) CHECKLIST

Submit one completed copy of this Checklist, including any required attachments, for each project involving human subjects. The IRB meets monthly to evaluate proposals, and approval is granted for one academic year. See *IRB Guidelines and Procedures* for details.

Project Director: YOLANDA REIMER Dept.: Comp Sci Phone: 721-5098  
Signature: Yolanda J. Reimer Date: 4-8-96

Co-Director(s): \_\_\_\_\_ Dept.: \_\_\_\_\_ Phone: \_\_\_\_\_

Project Title: USER TESTING OF ALGORITHM ANIMATION

Project Description: Testing the efficiency of an algorithm animation  
(in nontechnical language) program on learning. Goal is to analyze how test  
users interact with the animation program.

Please provide the dates requested below:

Date Submitted to IRB	Projected Start Date	Ending Date
	<u>5-15-96</u>	<u>6-30-96</u>

Students only:  
Faculty Supervisor: NICHOLAS P. WILKE Dept.: Comp Sci Phone: 293-4375  
Signature: [Signature]  
(My signature confirms that I have read the IRB Checklist and attachments and agree that it accurately and adequately represents the planned research and that I will supervise this research project.)

Project Director: Please complete page 2 of IRB Checklist, on back.

For IRB Use Only

### IRB Review and Determination:

Exempt from Review       Expedited/Administrative Review       Approved

Conditional approval: \_\_\_\_\_

Resubmit proposal: \_\_\_\_\_

Disapproved: \_\_\_\_\_

Signature/IRB Chair: James A. Walsh Date: 4-30-96

## Appendix 1. IRB Approval.

### Project Information

1. In your opinion, does this project meet the requirements for *Research Exempt from Review* as outlined in Section B of the IRB Guidelines and Procedures?

Yes (Complete information below and attach questionnaire/instrument)

No (Complete information below and attach IRB Summary, eleven items)

2. Human Subjects. Describe briefly: Volunteer students from undergraduate computer science course - CS344 CS342. CS332

Are any of the following included? Check all that apply.

Minors (under age 18)  If YES, specify age range(s): \_\_\_\_\_  
 Members of physically, psychologically, or socially vulnerable population?

3. How are subjects selected/recruited? Explain briefly: on a volunteer basis only for extra course credit.

4. Identification of subjects in data.

Anonymous, no identification  Identified by name and/or address or other

5. Subject matter or kind(s) of information to be compiled from/about subjects.

Describe briefly: How well certain aspects of the program worked and why.

Is information on any of the following included? Check all that apply.

Sexual behavior  Illegal conduct  
 Alcohol use/abuse  Drug use/abuse  
 Information about the subject that, if it became known outside the research, could reasonably place the subject at risk of criminal or civil liability or be damaging to the subject's financial standing or employability.

6. Means of obtaining the information. Check all that apply.

Field/Laboratory observation  Mail survey (Attach questionnaire/instrument)  
 Tissue/Blood sampling  On-site survey (Attach questionnaire/instrument)  
 Measurement of motions/actions  Examine public documents, records, data, etc.  
 In-person interviews/survey (Attach questionnaire/instrument)  Examine private documents, records, data, etc.  
 Telephone interviews/survey (Attach questionnaire/instrument)  Use of standard educational tests, etc.  
 Other means (specify): \_\_\_\_\_

7. Is a written consent form being used:  Yes (Attach copy)  No

*Don't have yet.*

8. Will subject(s) receive an explanation of the research before and/or after the project?

Yes (Attach copy)  No  
*Verbally*

9. Is this part of your thesis or dissertation?  Yes  No

If YES \_\_\_\_\_ date you successfully presented your proposal to your committee: \_\_\_\_\_

## Appendix 2. Consent Form.

### **Informed Consent Form for usability and effectiveness testing of an algorithm animation system**

You are being invited to participate in a research project conducted by Yolanda Reimer, a graduate student at the University of Montana's Department of Computer Science. This project is conducted under the direction of Professor Nick Wilde, Department of Computer Science.

You are invited to participate in a research study to evaluate the usability and effectiveness of an algorithm animation system. You will be asked to use a particular view of the algorithm animation system to learn as much as you can about the algorithm being presented. You may take as much time as you need to do so. When you are done, you will be asked to fill out a post-test to the best of your ability. You are not expected to be able to answer all the questions on the post-test, but you should do as well as you can. Again, no time limit will be given for the post-test. When you are done with the post-test, you will be asked to fill out a questionnaire describing the animation system. During this experiment, I will be taking notes and volunteering as little information as possible. You will be asked to verbalize your thoughts or "think aloud" while working. If you should get stuck at any point, try to use the materials given to get yourself moving again. If this doesn't work, I may help you.

Your participation in this project is entirely voluntary and you have the right to withdraw consent or discontinue participation at any time. You have the right to refuse to answer any questions for any reason. While you are participating in this study, please remember it is the animation program that is being evaluated, not you.

In addition, your individual privacy will be maintained in all published and written data resulting from this study. Your post-test and questionnaire will be identified by code # only.

If you have questions regarding your rights as a subject, concerns regarding this project, or any dissatisfaction with any aspect of this study, you may report them (confidentially, if you wish) to Dr. Walsh, Investigative Review Board, Psychology Department, University of Montana. Copies of the University of Montana Assurance of Compliance to the federal government regarding human subject research are available upon request from the Dr. Walsh, listed above.

I understand the above information and voluntarily consent to participate in the research project entitled "Usability and effectiveness testing of an algorithm animation system."

Signed \_\_\_\_\_

Date \_\_\_\_\_

### Appendix 3. List of post-test questions.

#### Graham's scan convex hull post-test

Thank you for participating in this algorithm animation research. Please answer the following questions to the best of your ability. Note: It is our intention that you will not necessarily be able to answer all the questions, but do what you can. This is NOT a test of you or your intelligence. Instead it is a tool that we hope will help us gain insight into how algorithm animations can be most effectively used. Again, thanks for all your help.

1. Why was it necessary to keep track of the NEXT-TO-TOP element of the stack?

*Conceptual*

2. Do all points in the set Q eventually get labeled?                      Yes            No

*Procedural                      5 points*

If no, why not?

*Procedural                      5 points*

3. How is the first point,  $p_0$ , determined?

*Procedural                      5 points*

4. What does the stack get initialized with?

*Procedural                      5 points*

5. What is the average run-time of Graham's scan convex hull algorithm?

*Procedural                      5 points*

6. How is it determined if a point is a vertex on the convex hull or not?

*Conceptual*

7. When do points get PUSHED onto the stack?

*Conceptual*

8. Explain how the points in set Q are labeled as specifically as you can.

*Conceptual*

9. What were the purpose of the variables  $i$  and  $m$ ?

*Procedural*            *5 points*

10. If you were trying to explain the convex hull algorithm to a friend in non-technical terms (not mentioning pseudocode or data structures), how would you describe it?

*Conceptual*

11. Is there a minimum number of points in set  $Q$  required to run this algorithm?

Yes                      No

*Conceptual*

12. Write down your understanding of the main steps of the algorithm, in the order that they occur, in either pseudocode or plain English.

*Procedural for users VCO, CO, VCI, CI#1, CI#2*    *10 points*

*Conceptual for user VO*

13. Is Graham's scan the only algorithm used to determine the convex hull of a set of points?

Yes                      No

*Procedural*            *5 points*

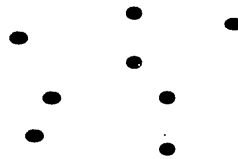
If not, can you name any of the others?

*Procedural*            *5 points*

14. Use as many of the following pictures as needed in order to find the convex hull of the set of points below. Try to illustrate this process as clearly as you can by using a different picture to show the main steps of the algorithm and by describing briefly what you're doing at each step.

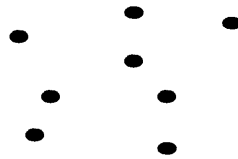
*Conceptual*

1



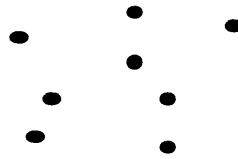
Description:

2



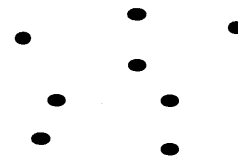
Description:

3



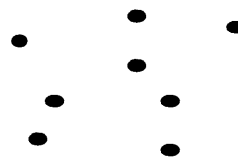
Description:

4



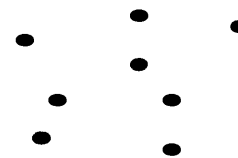
Description:

5



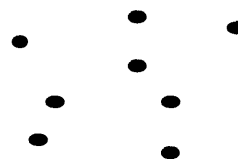
Description:

6



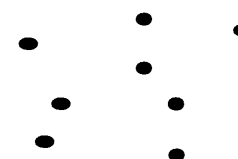
Description:

7



Description:

8



Description:

## Appendix 4: Post-test questionnaire.

### *Questionnaire*

1. Do you feel that this animation helped you in learning Graham's scan convex hull algorithm? If so, how and why. Please be as specific as possible.
2. Did you have any knowledge of the convex hull algorithm or Graham's scan convex hull algorithm prior to using this tool?

Yes

No

If yes, please explain how much and where acquired.

3. Are there any features or functionality that you felt would have helped you to better learn this algorithm that were not part of the animation? If so, what were they (specifically).
4. Did you feel, in general, there was too much, too little, or just the right amount of information in the animation?
5. What did the animation do particularly well, if anything?
6. Where could improvements be made in this animation?
7. What aspect of this algorithm took you longer to understand (check all that apply):
  - \_\_\_\_\_ comparison of points to determine if non-left angle
  - \_\_\_\_\_ how things got pushed and popped from stack
  - \_\_\_\_\_ how stack was initialized
  - \_\_\_\_\_ how points were initially labeled
  - \_\_\_\_\_ what angle was currently being analyzed
8. How did you feel in using this animation tool? (excited, bored, etc.)
9. Rank the following ways in which this tool would be best utilized to help you in learning this and other algorithms? (5-best use, 1-worst use)

- \_\_\_\_\_ teacher using this as lecture support (to show examples in class, etc.)
- \_\_\_\_\_ teacher using this as lecture support and having required lab sessions and/or homework with this type of animation program
- \_\_\_\_\_ teacher giving traditional lectures, and having required lab sessions and/or homework with this type of animation program
- \_\_\_\_\_ using the animation program only as lab and homework tool, not in conjunction with any lectures
- \_\_\_\_\_ not at all



## BIBLIOGRAPHY

- [1 ] Apple Computer, Inc. (1989). *HyperCard Stack Design Guidelines*. Addison-Wesley Publishing Company, Inc. Cupertino, CA.
- [2] James E. Baker, Isabel F. Cruz, Giuseppe Liotta, and Roberto Tamassia. "A New Model for Algorithm Animation Over the WWW." *ACM Computing Surveys*, 27(4): 568-572, December 1995.
- [3] Thomas Ball and Stephen G. Eick. "Software Visualization in the Large." *Computer*, 29(4): 33-43, April 1996.
- [4] Marc H. Brown. *Algorithm Animation*. MIT Press, Cambridge MA. 1988.
- [5] Marc H. Brown. Video: "An Anthology of Algorithm Animations using Zeus". Digital Systems Research Center, 1991.
- [6] Marc H. Brown. "Exploring algorithms using Balsa-II." *Computer*, 21(5):14-36, May 1988.
- [7] Marc H. Brown. "The 1992 SRC Algorithm Animation Festival." <http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-098.html>, March 1993.
- [8] Marc H. Brown and Robert Sedgewick. "Techniques for Algorithm Animation." *IEEE Software*, 2(1):28-39, January 1985.
- [9] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990, pages 898-907.
- [10] Kenneth C. Cox and Gruia-Catalin Roman. "Abstraction in Algorithm Animation." In *Proceedings of the 1992 IEEE Workshop on Visual Languages* (Seattle, WA), pp. 18-23, 1992.
- [11] Kenneth C. Cox and Gruia-Catalin Roman. "A Taxonomy of Program Visualization Systems." Technical report WUCS-93-22, Department of Computer Science, Washington University, Saint Louis, MO, 1993.
- [12] Sarah Douglas, Christopher Hundhausen, and Donna McKeown. "Exploring Human Visualization of Computer Algorithms." Technical report CIS-TR-94-27, Department of Computer and Information Science, University of Oregon, Eugene, 1994.
- [13] Sarah Douglas, Christopher Hundhausen, and Donna McKeown. "Toward Empirically-Based Software Visualization Languages." Technical report CIS-TR-95-12, Department of Computer and Information Science, University of Oregon, Eugene, 1995.

- [14] Peter A. Gloor. "AACE - Algorithm Animation for Computer Science Education." In *Proceedings of the 1992 IEEE Workshop on Visual Languages*, pages 25-31, Seattle, WA, September 1992.
- [15] Danny Goodman. *The Complete HyperCard Handbook, 2nd edition*. Bantam Books, 1988.
- [16] R.L. Graham. "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set." *Information Processing Letters*, 1:132-133, 1972.
- [17] Judith S. Gurka. "Algorithm Animation in the Classroom??. or If It's So Nifty, How Come We Can't Prove it?" Birds of a Feather Session. SIGCSE '96, Philadelphia PA.
- [18] Judith S. Gurka and Wayne Citrin. "Testing Effectiveness of Algorithm Animation." <http://soglio.Colorado.EDU/Web/Home.html#publications>, to appear in *IEEE Visual Languages Symposium*, 1996.
- [19] Clayton Lewis and John Rieman. *Task-Centered User Interface Design, A Practical Introduction*. University of Colorado @ Boulder. Shareware, 1993, 1994.
- [20] Richard E. Mayer and Richard B. Anderson. "The Instructive Animation: Helping Students Build Connections Between Words and Pictures in Multimedia Learning." *Journal of Educational Psychology*, 84(4):444-452, 1992.
- [21] Susan Palmiteer and Jay Elkerton. "An Evaluation of Animated Demonstrations for Learning Computer-Based Tasks," in *Proceedings of the ACM SIGCHI '91 Conference on Human Factors in Computing Systems*, pages 257-263, New Orleans, LA, May 1991.
- [22] Robert Sedgewick. *Algorithms, 2nd edition*. Addison-Wesley Publishing Co, Inc., 1988, pages 359-372.
- [23] John Stasko, Albert Badre, and Clayton Lewis. "Do Algorithm Animations Assist Learning? An Empirical Study and Analysis," in *Proceedings of the ACM INTERCHI '93 Conference on Human Factors in Computing Systems*, pages 61-66, Amsterdam, Netherlands, April 1993.
- [24] John T. Stasko, Andrea W. Lawrence, and Albert M. Badre, . "Empirically Evaluating the Use of Animations to Teach Algorithms," in *Proceedings of the 1994 IEEE Workshop on Visual Languages*, pages 48-54.
- [25] John Stasko. "TANGO: A Framework and System for Algorithm Animation." *Computer*, 23(9):39-44, 1990.

- [26] John Stasko and Sougata Mukherjea. "Toward Visual Debugging: Integrating Algorithm Animation Capabilities within a Source-Level Debugger," in *ACM Trans. on Computer-Human Interaction* 1(3), pages 215-244, 1994.
- [27] John T. Stasko and Charles Patterson. "Understanding and Characterizing Software Visualization Systems." In *Proceedings of the 1992 IEEE Workshop on Visual Languages*. IEEE, New York, 3-10.
- [28] Roger E. Whitney and N. Scott Urquhart. "Microcomputers in the Mathematical Sciences: Effects of Courses, Students, and Instructors." *Academic Computing*, 4(6): 14-18, 49-53, March 1990.
- [29] Dan Winkler and Scot Kamins. *HyperTalk 2.0: The Book*. Bantam Books, 1990.